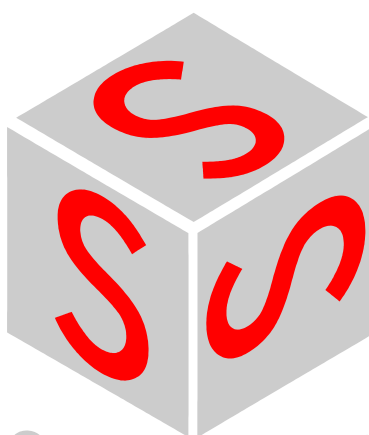


Manuel

CoDeSys SoftMotion V2.3

**Complément de manuel du
Système de programmation CoDeSys**



**S m a r t
Software
Solutions**

Copyright © 2007 by 3S - Smart Software Solutions GmbH
Tous droits réservés.

Toutes les mesures existantes ont été prises afin de garantir l'exactitude et l'intégralité de la documentation présente. Etant donné que des fautes restent toujours possible, malgré toutes les précautions qui sont prises, nous vous serions reconnaissants de bien vouloir nous faire part de vos remarques et de vos suggestions.

Editeur:

3S - Smart Software Solutions GmbH
Memminger Straße 151
D-87439 Kempten
Tél. +49/ 831/ 5 40 31 - 0
Fax +49/ 831/ 5 40 31 - 50

Edition: **09.02.2007 (CoDeSys 2.3.7.3)**
Document Version: **1.0**

Contenu

1	CoDeSys SoftMotion	1-1
1.1	Aperçu.....	1-1
2	Drive Interface SoftMotion	2-1
2.1	Configuration de commande pour SoftMotion	2-2
2.1.1	Dialogue de configuration BusInterface	2-2
2.1.2	Dialogue de configuration AxisGroup	2-2
2.1.3	Dialogue de configuration d'entraînement	2-4
2.1.4	Dialogue de configuration de l'encodeur.....	2-7
2.2	SM_DriveBasic.lib et génération automatique de code	2-7
2.2.1	Modules auxiliaires mathématiques.....	2-8
2.2.2	Modules de groupes d'axes	2-8
2.2.3	Modules de configuration.....	2-10
2.2.4	Modules de mode de régulation.....	2-11
2.2.5	Saisie directe de valeurs de consigne	2-11
2.2.6	Axe temps virtuel	2-16
2.2.7	Référencement via entrées numériques de matériel hardware	2-17
2.2.8	Modules de diagnostic	2-19
2.2.9	SMC_Encoder.....	2-20
2.2.10	Modèles de visualisation.....	2-20
2.3	Pilotes d'entraînement <DésignationBusInterface>Drive.lib	2-21
2.3.1	SercosDrive.lib.....	2-21
2.3.2	SM_CAN.lib	2-21
2.4	Variables de la structure AXIS_REF.....	2-22
2.5	Paramétrage de l'entraînement	2-27
3	L'éditeur CNC dans CoDeSys	3-1
3.1	Éditeur CNC - Aperçu	3-1
3.2	Éléments supportés et étendus du langage CNC DIN 66025.....	3-2
3.2.1	Fonction de point de commutation, Fonction H	3-4
3.2.2	Fonction auxiliaire, Fonction M	3-5
3.2.3	Utilisation de variables	3-5
3.2.4	Interpolation circulaire.....	3-5
3.2.5	Interpolation d'ellipse	3-6
3.2.6	Interpolation de spline.....	3-6
3.2.7	Sauts conditionnels.....	3-7
3.2.8	Modifier valeurs de variables	3-7
3.3	Démarrage de l'éditeur, insertion et gestion des programmes CNC	3-7
3.4	Éditeur CNC littéral	3-12
3.5	Éditeur CNC graphique.....	3-12
3.6	Commandes et options.....	3-13
3.7	Remplissage automatique de structure	3-16
4	L'éditeur CAM dans CoDeSys	4-1
4.1	Aperçu.....	4-1
4.2	Définition d'une CAM pour SoftMotion	4-1
4.3	Démarrage e inserer nouvelle CAM.....	4-1
4.4	Éditer CAM.....	4-3

4.4.1	Configuration générale de l'éditeur	4-3
4.4.2	Édition des propriétés des éléments de courbe	4-4
4.4.3	Fonctions des menus ,Extras' et ,Insérer'	4-6
4.5	L'éditeur CAM en mode en ligne	4-9
4.6	Utilisation des CAM	4-10
4.6.1	Effets des paramètres de module	4-10
4.6.2	Commutation entre CAM	4-12
4.7	Structure de données CAM	4-13
4.7.1	Exemple de CAM générée manuellement	4-15
5	Bibliothèque SM PLCopen.lib	5-1
5.1	Aperçu	5-1
5.2	Spécification PLCopen "Function blocks for motion control, Version 1.0"	5-1
5.3	Commande d'axes individuels	5-2
5.4	Commande de mouvements synchronisés	5-18
5.5	Modules supplémentaires	5-23
6	Bibliothèque SM CNC.lib	6-1
6.1	Aperçu	6-1
6.2	Modules	6-1
6.2.1	SMC_NCDecoder	6-1
6.2.2	SMC_GCodeViewer	6-3
6.2.3	SMC_ToolCorr	6-4
6.2.4	SMC_AvoidLoop	6-6
6.2.5	SMC_SmoothPath	6-7
6.2.6	SMC_RoundPath	6-9
6.2.7	SMC_CheckVelocities	6-10
6.2.8	SMC_LimitCircularVelocities	6-11
6.2.9	SMC_Interpolator	6-12
6.2.10	SMC_GetMPParameters	6-16
6.2.11	SMC_Interpolator2Dir	6-17
6.3	Fonctions et blocs fonctionnels auxiliaires	6-18
6.4	Paramétrage via variables globales	6-18
6.5	Structures de SM_CNC.lib	6-19
6.6	Trajectoire CAM avec SMC_XInterpolator	6-26
7	Bibliothèque SM CNCDiagnostic.lib	7-1
7.1	Modules pour l'analyse des données SMC_CNC_REF	7-1
7.1.1	SMC_ShowCNCREF	7-1
7.2	Modules pour l'analyse des données SMC_Outqueue	7-1
7.2.1	SMC_ShowQueue	7-1
8	Bibliothèque SM Trafo.lib	8-1
8.1	Aperçu	8-1
8.2	Blocs fonctionnels de transformation	8-1
8.2.1	Systèmes à portique	8-1
8.2.2	Systèmes à portique avec décalage d'outil	8-4
8.2.3	Système de portiques H avec entraînements stationnaires	8-8
8.2.4	Systèmes Scara à double articulation	8-9
8.2.5	Systèmes Scara à triple articulation	8-11
8.2.6	Cinématique parallèle	8-13
8.3	Transformation dans l'espace	8-15

9	Bibliothèque SM Error.lib	9-1
9.1	Aperçu.....	9-1
9.2	Blocs fonctionnels	9-1
9.2.1	SMC_ErrorString.....	9-1
9.3	L'énumération SMC_Error	9-1
10	Bibliothèque SM FileFBs.lib	10-1
10.1	Aperçu de la bibliothèque SM_FileFBs.lib	10-1
10.2	Blocs fonctionnels CNC	10-1
10.3	Blocs fonctionnels CAM	10-3
10.4	Blocs fonctionnels de diagnostic.....	10-4
11	Exemples de programmation	11-1
11.1	Aperçu.....	11-1
11.2	Drive Interface : Créer une configuration de commande pour des entraînements 11-1	
11.3	Commande d'axes individuels (single-axis).....	11-4
11.4	Commande d'axe unique en CFC avec modèle de visualisation (single-axis) ..	11-5
11.5	Commande d'entraînement CAM via axe de temps virtuel.....	11-7
11.6	CAM alternées	11-8
11.7	Commande à l'aide de l'éditeur CNC.....	11-8
11.7.1	Exemple CNC 1 : Création directe de OutQueue	11-9
11.7.2	Exemple CNC 2 : Décodage en ligne à l'aide de variables	11-11
11.7.3	Exemple CNC 3 : Préparation de trajectoire en ligne	11-14
11.8	Programmation dynamique SoftMotion.....	11-15
12	Index	I

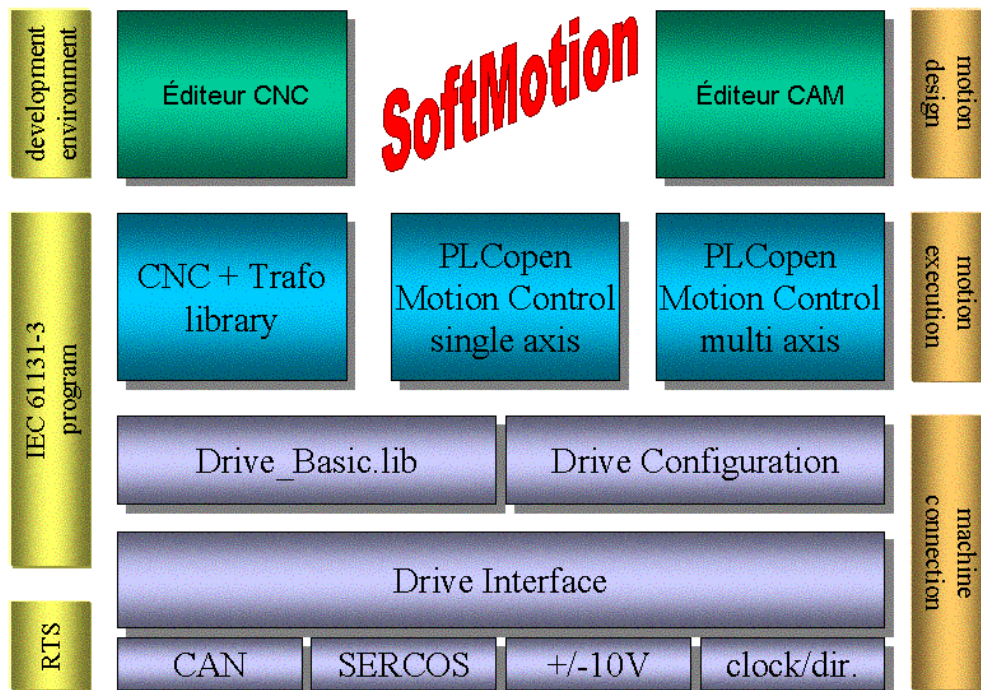
1 CoDeSys SoftMotion

1.1 Aperçu

SoftMotion permet de réaliser des mouvements - des mouvements simples à un seul axe aux mouvements complexes multi-dimensionnels, en passant par les mouvements de came, cela dans l'environnement de développement de *CoDeSys*. Surtout applications pour lesquelles les fonctions de déplacement ne sont pas seules à l'avant-plan ; une grande partie de ces applications est également prise par la commande du déroulement ou du processus pou par des fonctions auxiliaires. De telles applications sont idéales pour *SoftMotion*.

SoftMotion est une sorte de coffre à outils de base à l'aide duquel on peut influencer sur le mouvement de nombreuses manières - même en cours d'exécution selon le processus -, cela sans grands frais ni savoir-faire détaillé pour l'exécution des mouvements souhaités.

Toute la logique du programme est traitée dans le programme API, et seules les informations touchant exclusivement aux mouvements sont exécutées dans les modules de bibliothèque.



SoftMotion comporte les composants ci-dessous :

- DriveInterface**
 Ce composant *SoftMotion* est responsable de la communication avec les entraînements. Il se compose de la bibliothèque *Drive_Basic.lib* ainsi que des bibliothèques et pilotes spécifiques au système d'entraînement et de bus.
- L'éditeur de configuration** dans CoDeSys donne au programmeur une représentation de la structure et du paramétrage du matériel hardware de l'entraînement. À partir de cela, CoDeSys crée à l'aide des **bibliothèques DriveInterface** des structures de données IEC qui représentent les entraînements de manière abstraite. La Drive Interface communique automatiquement (et donc sans frais supplémentaires pour le programmeur IEC) avec les entraînements et garantit ainsi la mise à jour permanente des données structurelles de l'entraînement et la transmission des fichiers qui y ont été modifiés. Afin de commander le matériel hardware de l'entraînement, le programme IEC peut accéder à ces structures de données soit par le biais de modules standard provenant des bibliothèques *SoftMotion* (SM_CNC.lib, SM_PLCOpen.lib), soit à l'aide de programmes créés de manière spécifique par le programmeur IEC.

- La saisie des valeurs de consigne se fait toujours de manière cyclique, ce qui signifie que par cycle de tâche IEC, les valeurs de consigne (position, vitesse, accélération etc.) sont calculées et transmises par la Drive Interface aux entraînements. Il n'est pas prévu de "donner des ordres" aux entraînements, comme p.ex. la saisie d'une position cible de sorte que l'entraînement effectue de lui-même le déplacement correspondant et communique l'exécution correcte de l'ordre. Parmi les raisons, on peut citer le fait que les mouvements coordonnés de plusieurs axes seraient impossibles et que la commande centrale n'aurait aucune influence sur les entraînements lors de l'exécution de l'ordre.
- **Éditeur CNC**
L'éditeur CNC de CoDeSys sert à la programmation des mouvements des entraînements qui peuvent ensuite être communiqués par la Drive Interface au matériel hardware des entraînements et y être commandés. Il est possible de programmer à la fois graphiquement et textuellement des mouvements pluridimensionnels, conformément au langage CNC DIN66025. Il est en principe possible de réaliser des mouvements à max. 9 dimensions, et seulement deux dimensions peuvent être interpolées de manière autre que linéaire. Cela signifie que sur deux dimensions, il est possible de programmer des lignes droites, des cercles, des arcs de cercle, des paraboles, des ellipses et des splines ; les autres directions ne sont interpolées que de manière linéaire. Pour chaque trajectoire créée, CoDeSys génère automatiquement une structure de données globales (données CNC) qui sont à disposition dans le programme IEC.
- **Éditeur CAM**
L'éditeur CAM à commande graphique intégré à CoDeSys sert à la programmation de cames (CAM) pour la commande d'entraînements à plusieurs axes. Pour chaque trajectoire programmée, CoDeSys crée automatiquement une structure de données globales (données CAM) qui est à disposition dans le programme IEC.
- **Bibliothèque CNC**
Les bibliothèques "SM_CNC.lib", "SM_CNCDiagnostic.lib" et "SM_Trafo.lib" mettent à la disposition du programmeur IEC des modules à l'aide desquels il peut éditer, représenter et exécuter les mouvements créés au sein de l'éditeur CNC ou planifiés en cours d'exécution.
- **Bibliothèque PLCopen**
La bibliothèque PLCopen MotionControl Éditeur CNC.lib" se compose entre autres de modules permettant de programmer et exécuter en toute simplicité non seulement la commande d'axes individuels, mais également de mouvements synchronisés de deux axes. Outre des modules de scrutation du statut, de paramétrage et d'utilisation générale, elle contient des blocs fonctionnels qui sont à même de déplacer de plusieurs façons un axe selon un comportement prédéfini de vitesse et d'accélération. Si deux axes doivent être synchronisés, l'un d'entre eux est le maître qui commande selon certaines prescriptions le second (esclave). Ces prescriptions peuvent p.ex. être une CAM générée au sein de l'éditeur CAM et qui accouple via les modules disponibles l'axe esclave à l'axe maître. En outre, il existe des blocs fonctionnels permettant la programmation d'entraînements électroniques ou de décalages de phases.
- Bibliothèque de fichiers **SM_FileFBs.lib** La bibliothèque "SM_FileFBs.lib" repose sur la bibliothèque du système "SysLibFile.lib" et ne peut de ce fait être utilisée qu'avec des commande qui supportent cette bibliothèque.
- Bibliothèque d'erreur **SM_Error.lib**
?La bibliothèque "SM_Error.lib" contient tous les messages d'erreur qui ont pu être générés par les modules des autres bibliothèques. En outre, elle permet de créer à partir des variables numériques d'erreur des textes d'erreur en allemand et en anglais.

Portabilité

À l'exception de quelques pilotes appartenant à DriveInterface et qui peuvent directement commander les composants du matériel hardware, tous les composants d'exécution SoftMotion sont programmés selon IEC1131-3. Ceci permet d'atteindre une indépendance maximum par rapport aux plates-formes.

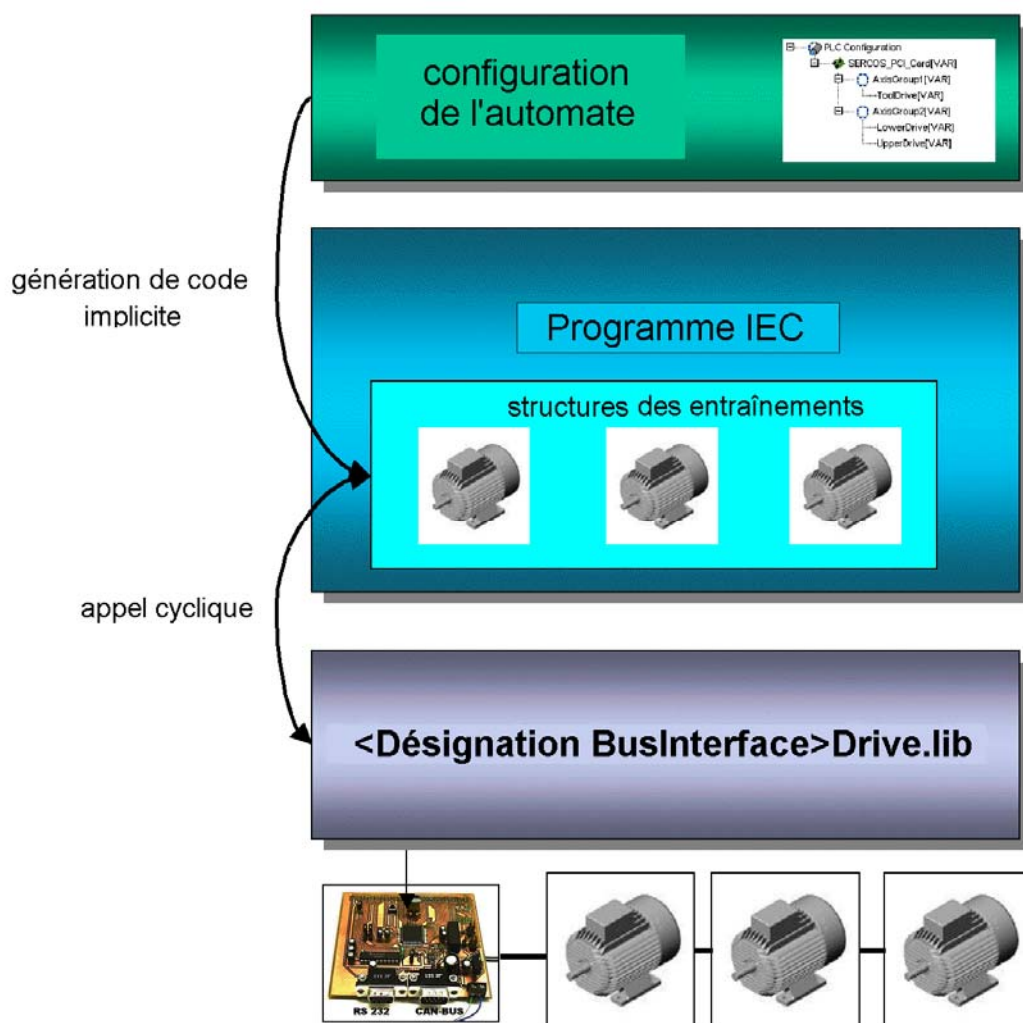
Pour une compréhension rapide des composants SoftMotion, on recommande au lecteur d'étudier les exemples s'y rapportant.

2 Drive Interface SoftMotion

La Drive Interface SoftMotion est une interface normalisée qui offre au programmeur la possibilité d'incorporer l'image abstraite d'un entraînement dans le programme IEC, de l'y configurer et d'y accéder. Cette interface veille automatiquement à la mise à jour et la transmission des données qui sont nécessaires à la commande du matériel hardware de l'entraînement. Ainsi, les entraînements peuvent non seulement être aisément échangés et les programmes IEC réutilisés, mais l'utilisateur évite également les difficultés et les problèmes liés à l'incorporation d'un entraînement.





La Drive Interface travaille avec les composants suivants :

- Le **configurateur de l'automate** CoDeSys : Sur base d'un fichier de configuration approprié, on doit ici représenter et paramétrer la structure des entraînements à exploiter. Cette structure est ensuite rendue accessible à l'aide des bibliothèques Drive Interface, via des variables (de système) créées implicitement pour et affectées à l'application (programme IEC).
- La bibliothèque interne **SM_Drive_Basic.lib** propose des structures de données IEC et des variables globales qui correspondent aux entraînements, groupes d'axes et interfaces de bus représentés au sein du configurateur de commande.
- Le pilote d'entraînement, c.-à-d. la **bibliothèque spécifique <Désignation BusInterface>Drive.lib** devant être fournie par le fabricant de la commande pour les entraînements et le système de bus (p.ex. SercosDrive.lib ou Sercos<nom de client>Drive.lib) propose des fonctions spéciales qui sont nécessaires pour l'échange de données entre les structures et le matériel hardware concerné.



2.1 Configuration de commande pour SoftMotion

Afin de représenter la structure des entraînements, la configuration de l'automate CoDeSys propose les éléments suivants :

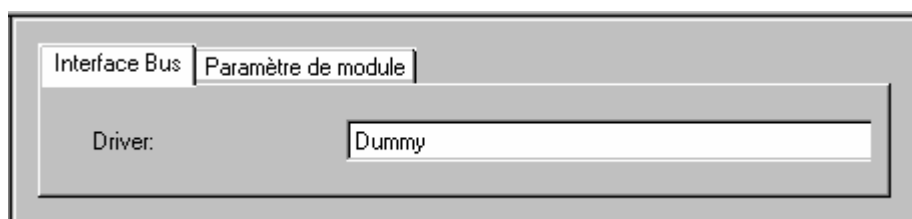
-  **BusInterface** : interface de bus de terrain permettant de communiquer avec les entraînements
-  **AxisGroup** : un groupe d'entraînements physiquement cohérent
-  **Drive** : entraînement
-  **Encoder** : transmetteur

Les interfaces de bus, groupes d'axes et entraînements peuvent être désignés de manière univoque par des identificateurs IEC 61131-3 :

Chacun de ces objets de configuration peut être configuré via un dialogue, pour autant que la description de la configuration de la cible le permette. Outre leur configuration aisée via un dialogue, les paramètres peuvent être réglés au sein d'une liste de configuration ("paramètres de module"). On y trouve également des paramètres spécifiques à la cible qui commencent avec "MS".

2.1.1 Dialogue de configuration BusInterface

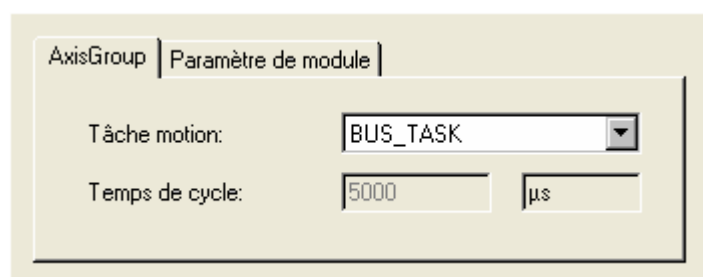
On ne choisit en général ici que le pilote de communication.



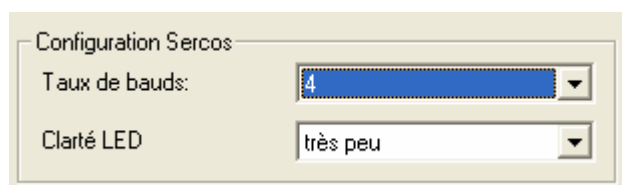
2.1.2 Dialogue de configuration AxisGroup

Ce dialogue indique la tâche qui voit la communication avec les entraînements ainsi que leur temps de cycle - s'il s'agit non pas d'une tâche cyclique, mais bien d'une tâche déclenchée par événement.

Dans le cas de systèmes sans configuration de tâche, ce champ reste vide.



Avec des interfaces **Sercos**, il faut procéder à d'autres réglages spécifiques, en l'occurrence le taux de baud en MBaud et l'intensité des LED.



Pour les **groupes d'axes CAN** également, il faut procéder à des réglages spécifiques :

Configuration spécifique

No. de contrôleur: 4

Taux de bauds: 500 kBaud

Générateur SYNC: PLC

Outre le taux de Baud, on définit avec Controller n° le numéro du contrôleur CAN utilisé (remarque : si vous utilisez une commande à 2 canaux CAN avec la bibliothèque 3S-CANopen.lib, cette dernière prend par défaut Controller 0, vous devez donc sélectionner le canal 1 pour les entraînements).

Avec un générateur SYNC, on peut choisir entre trois méthodes différentes pour synchroniser entraînements et API :

- API: l'API est ici utilisé comme maître de synchronisation. En général, l'utilisateur définit la tâche Motion comme étant une tâche cyclique. Cette tâche appelle le pilote qui envoie directement un télégramme SYNC. Cette méthode est certes la plus simple mais elle peut cependant entraîner des problèmes avec une commande à gigue élevée et avec des entraînements qui exigent un télégramme SYNC extrêmement précis.
- 1er entraînement : s'il supporte cette propriété, le premier entraînement génère ici le télégramme SYNC. La tâche Motion du SPS est en général définie sur base de l'événement <AxisGroup>.bSync et attend donc de recevoir un télégramme SYNC pour procéder ensuite à l'exécution de la tâche.
- SYNC_Device : cette méthode est utilisée si les deux méthodes ci-dessus s'avèrent impossibles. Un périphérique supplémentaire capable de générer des télégrammes SYNC minutieusement chronométrés est installé dans le bus avec CAN ID 127 (Index : 1005h, Bit30).

Tous ces réglages peuvent également être consultés et modifiés via l'onglet "Paramètres de module".

sTask	Cordon portant le nom de la tâche dans laquelle la transmission des données de ce groupe d'axes doit avoir lieu.
dwCycle	Temps de cycle (en µs) de la tâche définie sous "sTask" (ne doit être indiqué que si la commande ne supporte aucune tâche et appelle automatiquement PLC_PRG (tâche par défaut))
wParam1 ... wParam4	Paramètres de type WORD spécifiques à la carte / l'entraînement
dwParam1 .. dwParam4	Paramètres de type DWORD spécifiques à la carte / l'entraînement

2.1.3 Dialogue de configuration d'entraînement

Ce dialogue permet de définir l'identité de l'entraînement (**ID lecteur**). En outre, on peut y choisir le **type** d'entraînement, linéaire et rotatif (modulo).

Le champ **Facteur de conversion** permet de définir la conversion entre les valeurs complètes de positionnement reçues de l'entraînement et les unités techniques utilisées dans le programme UIC. De plus, on peut encore tenir compte d'un entraînement supplémentaire. dans l'exemple ci-dessus, un entraînement qui génère 3600000 pour une rotation a été étalonné de telle sorte que les unités techniques sont données en angles.

Dans le champ **Configuration pour commande linéaire**, il est possible de définir des logiciels pour limiter le déplacement en fonction du type d'**entraînement linéaire** choisi ; pour des **entraînements rotatifs**, il faut définir la plage Modulo.


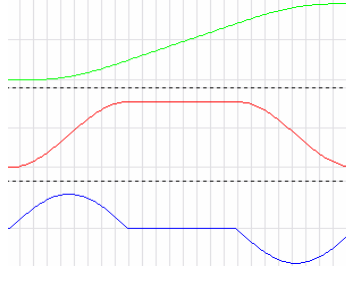
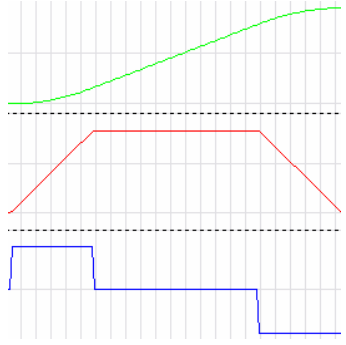
Le champ **Données de communication cycliques** permet à l'utilisateur de sélectionner les valeurs de consignes et réelles qui doivent être transmises à l'entraînement de manière cyclique.

Valeurs max. définit les valeurs limites utilisées par les modules SMC_ControlBy, afin de constater si on est en présence d'un saut (voir description de SMC_ControlAxisByPos).

Type de rampe de vitesse permet à l'utilisateur de spécifier (si supporté par les bibliothèques intégrées) le type de profil de vitesse pour les modules d'axes individuels et maître/esclave générant des mouvements. "Trapézoïdal" désigne un profil de vitesse trapézoïdal (accélération constante pas par pas), "sigmoïdal" un profil de vitesse en \sin^2 (accélération constante), "parabolique" un profil d'accélération constant trapézoïdal ayant pour conséquence un profil de vitesse parabolique.

Pour les modes sigmoïdal et parabolique, il faut en outre définir le jerk.

Les graphiques ci-dessous montrent comment les différents modes de vitesse fonctionnent lors d'un positionnement. La position est donnée en vert, la vitesse en rouge et l'accélération en bleu.

<p>Avec le mode de vitesse trapézoïdal, on constate que l'accélération est composée de sauts.</p>	
<p>Avec le mode de vitesse sigmoïdal, ces sauts sont supprimés. Le déroulement du mouvement est clairement défini lors de la phase de mouvement, ce qui ne permet pas de limiter le jerk. Ce dernier n'est constaté que lorsque l'entraînement présente au départ une valeur d'accélération qui n'est pas égale à 0. L'accélération est en l'occurrence ramenée à zéro avant que le mouvement lui-même ne soit entamé. Comparé au profil de vitesse trapézoïdal, ce mouvement dure plus longtemps.</p>	
<p>Avec le mode de vitesse parabolique, l'accélération prend un profil constant et trapézoïdal, l'inclinaison de la rampe étant limitée par le jerk. Pour la vitesse, cela donne un profil constant parabolique. c'est uniquement avec ce profil que l'on peut réellement limiter le jerk.</p>	

Tous ces réglages peuvent également être consultés et modifiés via l'onglet "Paramètres de module".

wld	ID de l'entraînement dans AxisGroup (WORD)
wControlType	<p>Types de commande et de réponse prédéfinis (WORD) ; possibilité de choix entre :</p> <p>(Données d'envoi -> données de réponse)</p> <ol style="list-style-type: none"> 1. TOR -> --- Couple -> --- 2. VEL -> VEL Vitesse -> vitesse 3. VEL -> POS Vitesse -> position 4. POS -> POS Position -> position 5. POS,VEL ->POS,VEL Position, vitesse ->vitesse, position 6. VEL -> --- Vitesse -> --- 7. CONFIGURABLE Configuration manuelle via wCyclicDataS1, ..S2, ..S3 et wCyclicDataR1, ..R2, ..R3 (voir ci-dessous)
wCyclicDataS1 wCyclicDataS2 wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	<p>Définition des données d'envoi (..S..) et de réponse (..R..), pour autant que wControlType soit réglé sur "CONFIGURABLE" ; les possibilités de choix dépendent du lecteur d'entraînement. Sont en principe possibles :</p> <p>Act/SetPosition Position Act/SetVelocity Vitesse Act/SetTorque Couple (Torque) Act/SetCurrent Courant Act/SetUserDef Définitions spécifiques à l'utilisateur</p>

dwRatioTechUnitsDenom iRatioTechUnitsNum	Dénominateur et numérateur pour facteur de conversion de données de bus en unités techniques [u]; (DWORD ou INT)
iMovementType	Type de mouvement ; possibilités de sélection : linéaire ("linear") ou rotatif ("rotary")
fPositionPeriod	Période pour axes rotatifs en unités techniques ; Remarque : convertie en incréments (par "dwRatioTechUnitsDenom" et "iRatioTechUnitsNum"), cette période ne peut pas être supérieure à 16#80000000.
fSWMaxVelocity	Vitesse maximale pour contrôle de software
fSWMaxAcceleration	Accélération maximale pour contrôle de software
fSWMaxDeceleration	Décélération maximale pour contrôle de software
bSWLimitEnable	Actionner un contrôle de position logicielle (uniquement pour entraînements linéaires) qui fait que l'axe soit en statut d'erreur s'il quitte la fenêtre de positionnement.
fSWLimitNegative	Limite de positionnement négative (uniquement pour entraînements linéaires)
fSWLimitPositive	Limite de positionnement positive (uniquement pour entraînements linéaires)
bHWLimitEnable	Actionner un contrôle de position matérielle (uniquement pour entraînements linéaires) qui fait que l'axe soit en statut d'erreur s'il quitte la fenêtre de positionnement.

Un dialogue supplémentaire est disponible pour les entraînements Sercos.

Pour le type de périphérique, on peut choisir entre Entraînement et Périphérique E/S car il n'y a pas de support standard CoDeSys pour les E/S Sercos. Si on choisit Périphérique E/S, on ignore certains paramètres qui sont normalement transmis par le maître.

Par ailleurs, il est également possible de transmettre cycliquement des données supplémentaires de communication (en plus des valeurs standard POS, VEL, ACC, TOR, CUR). Pour ce faire, il faut saisir les numéros et longueurs des paramètres Sercos.

Contrôle **PackProfile** permet de vérifier si les réglages effectués l'ont été selon les normes PackProfile. On distingue pour ce faire les profils BasicA, BasicB et BasicC. Il faut noter que ce contrôle peut être effectué sur les données hors ligne. Si le mécanisme supplémentaire de configuration possible avec Sercos (lecture d'un fichier ASCII sur la commande) doit être utilisé, cela peut influencer sur le résultat. C'est la raison pour laquelle la bibliothèque SercosDrive.lib implémente un contrôle supplémentaire en ligne (voir Sercosdrive.pdf). Des fichiers XML sont fournis avec Sercosdrive.lib et peuvent être importés au sein de ce dialogue : ils contiennent tous les paramètres PackProfile admissibles.

En outre, il est possible de définir des paramètres qui doivent être écrits pour l'entraînement lors du lancement. Pour ce faire, il existe à chaque fois une liste de paramètres qui doit être envoyée à l'entraînement en Phase2, Phase3 et au début de Phase4. Avec ces listes correctes de paramètres, on peut ainsi veiller à ce que l'entraînement soit entièrement initialisé au début de l'application, par exemple dans le cas où il doit être échangé (voir également 2).

La fonction "bloqué" permet d'empêcher que des paramètres individuels soient automatiquement transmis par le pilote.

Tous ces réglages peuvent être enregistrés dans un fichier XML (bouton "enregistrer") ou être lus dans un fichier XML (bouton "charger").

Il existe également un dialogue spécial pour les entraînements CAN dans lequel on peut saisir des paramètres qui doivent être écrits pour l'entraînement lors du lancement. Ces dialogues peuvent également être enregistrés et chargés sous la forme de fichiers XML.

2.1.4 Dialogue de configuration de l'encodeur

Lors de la compilation, une structure de données de type SMC_ENCODER_REF est créée pour chaque encodeur configuré; cette structure est configurée avec les valeurs suivantes. Cette structure de données sert d'entrée pour le module SMC_Encoder qui génère un axe virtuel à partir de la configuration et des valeurs d'encodeur en temps réel. Selon les types d'interface BusInterface, la position actuelle de l'encodeur doit être saisie manuellement (entraînement fictif) dans le module SMC_Encoder, ou elle est lue automatiquement via le bus, en même temps que les valeurs réelles de l'entraînement.

wEncoderId	ID de l'encodeur (WORD)
dwRatioTechUnitsDenom iRatioTechUnitsNum	Dénominateur et numérateur pour facteur de conversion de données de bus (incréments d'entraînement) en unités techniques (unités utilisées dans l'application, unités SoftMotion) [u]; (DWORD ou INT)
iMovementType	Type de convertisseur ; possibilités de sélection : linéaire („linear“) ou rotatif („rotary“)
fPositionPeriod	Période pour axes rotatifs ; dépend des facteurs de conversion „dwRatioTechUnitsDenom“ et „iRatioTechUnitsNum“
bSWLimitEnable	Actionner un contrôle de position logicielle (uniquement pour entraînements linéaires) qui fait que l'axe lié ultérieurement via SMC_Encoder soit en statut d'erreur s'il quitte la fenêtre de positionnement.
fSWLimitNegative	Limite de positionnement négative (uniquement pour encodeur linéaire)
fSWLimitPositive	Limite de positionnement positive (uniquement pour encodeur linéaire)
dwEncoderModulo	Valeur à laquelle la mise en page de l'encodeur a lieu, largeur de bit de l'encodeur : (0: 32 Bit, 16#10000: 16 Bit, 16#1000: 12 Bit) etc.

En outre, d'autres paramètres spécifiques aux fabricants (ces paramètres commencent avec „P:“) peuvent également être présents, ils sont enregistrés dans les paramètres dwParam1..dwParam8 de SMC_ENCODER_REF.

2.2 SM_DriveBasic.lib et génération automatique de code

Si la bibliothèque „SM_DriveBasic.lib“ dans son application IEC1131 est intégrée à CoDeSys, des objets structurels sont automatiquement créés à partir de la représentation de l'entraînement saisie dans la configuration de la commande avec paramétrage approprié ; le programme IEC peut accéder à ces objets structurels.

Dans l'application IEC1131, il **faut** également intégrer la bibliothèque compatible avec la commande utilisée et spécifique au fabricant, avec le nom **<Désignation BusInterface>Drive.lib**. Celle-ci supporte les fonctions d'interface de pilote rapporté au matériel hardware. La désignation BusInterface résulte de la saisie sélectionnée dans la configuration de la commande, paramètres du module d'interface de bus, à „InterfaceType“. Elle est obtenue à partir de la désignation de sa chaîne, à savoir la partie gauche jusqu'au premier espace (exemple : „CAN (Peak)“ -> „CAN“ -> la bibliothèque spécifique au fabricant se dénomme : „CANDrive.lib“).

Au démarrage de l'application, l'appel implicite des fonctions <Désignation BusInterface>DriveExecute_Start et <Désignation BusInterface>DriveInit en début de tâche et <Désignation BusInterface>DriveExecute_End en fin de tâche assure la transmission et le contrôle des variables structurelles AXIS_REF. En cas de survenance d'erreur lors de l'initialisation des entraînements, la variable globale g_strBootupError contient une description de l'erreur créée par la bibliothèque<Désignation BusInterface>drive.lib.

Pour l'exécution de sa fonction principale, à savoir la représentation des entraînements dans le programme IEC, la bibliothèque SM_DriveBasic.lib contient quelques modules auxiliaires:

- Modules auxiliaires mathématiques
- Modules de groupes d'axes
- Modules de configuration
- Modules de mode de regulation
- Axe temps virtuel
- Modules de référencement via entrées numériques de matériel hardware
- Modules de diagnostic
- Modèles de visualisation

2.2.1 Modules auxiliaires mathématiques

La fonction **SMC_sgn** indique la valeur du signe de l'entrée, -1 pour une saisie négative, +1 pour une positive et 0 pour une saisie nulle.

La fonction **SMC_fmod** calcule la valeur Modulo de l'entrée x sur la période m. La valeur de réponse se situe toujours dans l'intervalle [0, m[.

La fonction **SMC_atan2** calcule et indique l'angle qui permet de résoudre l'équation suivante:

$\sin(\alpha) * f = \text{Sinus}$ et $\cos(\alpha) * f = \text{Cosinus}$.

À l'inverse des fonctions ATAN habituelles, la plage de valeur couvre ici tout l'intervalle [0; 2pi].

2.2.2 Modules de groupes d'axes

Modules auxiliaires de groupes d'axes

Les modules auxiliaires de groupes d'axes ci-dessous sont disponibles dans SM_DriveBasic.lib :

SMC_IsAxisGroupREady

SMC_GetAxisGroupState

SMC_ResetAxisGroup

SMC_DisableAllDrives

SMC_EnableAllDrives

Voir également ICI pour le chronométrage de cycle.

Chronométrage de cycle

Il est souvent essentiel à des fins de diagnostic, de pouvoir mesurer le temps de cycle exact d'un groupe d'axes, en particulier si la tâche commande activement la communication (p.ex. si le PLC intervient tant que générateur CAN-SYNC). La structure de données du groupe d'axes contient la variable `<AxisGroup>.bDebug` (BOOL) qui permet de commander cette mesure du temps. Si elle est TRUE, `<AxisGroup>.udiTaskMinTime` et `<AxisGroup>.udiTaskMaxTime` indiquent la durée minimale ou maximale écoulée entre deux appels de tâche. `<AxisGroup>.udiTaskRunTime` et `<AxisGroup>.udiTaskMaxRunTime` indiquent le temps actuel de cycle et le temps maximal de cycle de la tâche.

SMC_IsAxisGroupREady

Cette fonction de groupes d'axes de la bibliothèque SM_DriveBasic.lib indique par le biais d'une variable BOOLEenne si le lancement qui est exécuté de manière implicite pour chaque groupe d'axes au démarrage du programme IEC, est bien terminé et ledit groupe avec ses axes est prêt au service (TRUE), ou si le lancement n'est pas encore terminé ou si une erreur est survenue (FALSE).

SMC_GetAxisGroupState

Ce bloc fonctionnel donne des informations sur le statut du groupe d'axes :

Entrées (VAR_INPUT) du module :

bEnable : BOOL

Si cette entrée est TRUE, le module fournit des informations sur le statut du groupe d'axes.

Entrées / sorties (VAR_IN_OUT) du module :

AxisGroup : SMC_AXISGROUP_REF

Groupe d'axes pour lequel des informations sont nécessaires.

Sorties (VAR_OUTPUT) du module :

bDone : BOOL

TRUE dès que les données valides sont dans les sorties.

wState : WORD

Variable interne de statut des axes.

bStartingUp : BOOL

Le groupe d'axes est lancé, et les entraînements sont configurés. ($0 \leq wState \leq 99$)

bNormalOperation : BOOL

Groupe d'axes en fonctionnement normal. ($wState = 100$)

bResetting : BOOL

Le groupe d'axes est en cours de réinitialisation. ($200 \leq wState \leq 210$)

bErrorDuringStartUp : BOOL

Une erreur est survenue lors du lancement. ($wState \geq 1000$)

pErrorDrive : POINTER TO AXIS_REF

Le pointeur est sur l'axe qui a causé l'erreur. Valable uniquement si $bErrorDuringStartUp = TRUE$.

Cette sortie permet de supprimer l'axe défectueux dans le groupe d'axes en activant la variable $bDisableDriveInAxisGroup$ en cours d'exécution, de réinitialiser le groupe via $SMC_ResetAxisGroup$ et de continuer à le parcourir avec les autres axes en fonctionnement, pour autant que les redondances nécessaires existent au sein de la machine.

SMC_ResetAxisGroup

Ce bloc fonctionnel permet de réinitialiser complètement un groupe d'axes.

Entrées (VAR_INPUT) du module :

bExecute : BOOL

Si cette entrée est TRUE, le module entame la réinitialisation du groupe d'axes.

bKeepRatioSettings : BOOL

Si cette entrée est TRUE, les positions des entraînements ($dwRatioTechunitsDenom$ et $iRatioTechUnitsNum$), la valeur Modulo ($fPositionPeriod$) et le type d'axe ($iMovementType$, linear/rotatorisch) sont maintenus et ne sont pas remplacées par les valeurs réglées au sein de la configuration de la commande.

Entrées / sorties (VAR_IN_OUT) du module :

AxisGroup : SMC_AXISGROUP_REF

Groupe d'axes qui doit être réinitialisé.

Sorties (VAR OUTPUT) du module :

bDone : BOOL

TRUE dès que le processus est terminé.

bError : BOOL

Une erreur est survenue.

nErrorID : SMC_ERROR

Description de l'erreur.

SMC_DisableAllDrives

Cette fonction permet d'activer la variable bDisableDriveInAxisGroup de tous les entraînements. Ceci a pour conséquence qu'à l'exécution suivante de la fonction SMC_ResetAxisGroup, il n'y a plus d'entraînement disponible, mais bien que les mécanismes de communication de base du groupe d'axes sont seuls exécutés.

Entrées (VAR INPUT) de la fonction :

pAG : POINTER TO SMC_AXIS_GROUP

Pointeur sur le groupe d'axes.

Valeur de réponse de la fonction :

TRUE.

SMC_EnableAllDrives

À l'inverse de SMC_DisableAllDrives, tous les entraînements sont activés via cette fonction et ils sont lancés via l'appel de SMC_ResetAxisGroup.

Entrées (VAR INPUT) de la fonction :

pAG : POINTER TO SMC_AXIS_GROUP

Pointeur sur le groupe d'axes.

Valeur de réponse de la fonction :

TRUE.

2.2.3 Modules de configuration

Les modules de configuration ci-dessous sont disponibles dans SM_DriveBasic.lib :

SMC_ChangeGearingRatio

Grâce à ce module, le programme IEC peut modifier le rapport de transmission et le type d'entraînement en cours d'exécution.

Attention : Après l'exécution de ce module, le groupe d'axes doit être redémarré au moyen de SMC_ResetAxisGroup (bKeepRatioSettings=TRUE) car c'est la seule manière de garantir que toutes les variables soient correctement initialisées !

Entrées (VAR INPUT) du module :

bExecute : BOOL

Le module est activé avec un front montant.

dwRatioTechUnitsDenom : DWORD, iRatioTechUnitsNum : DWORD

Rapport de conversion entre les unités SoftMotion et les incréments (voir 2.1)

fPositionPeriod : LREAL

Période de position, valeur Modulo (uniquement pour entraînements rotatifs) (voir 2.1)

iMovementType : INT

0: axe rotatif, 1: axe linéaire.

Entrées / sorties (VAR_IN_OUT) du module :

Axis : AXIS_REF

Entraînement dont le rapport de transmission doit être modifié.

Sorties (VAR_OUTPUT) du module :

bDone : BOOL

TRUE dès que l'action est exécutée.

bError : BOOL

TRUE si une erreur survient.

nErrorID : SMC_Error

Décrit l'erreur.

2.2.4 Modules de mode de régulation

Les modules de configuration ci-dessous sont disponibles dans SM_DriveBasic.lib :

SMC_SetControllerMode

Ce module de la bibliothèque SM_DriveBasic.lib permet de passer à un autre mode de régulation - si cela est supporté par l'entraînement.

Entrées (VAR_INPUT) du module :

bExecute : BOOL

Le module est activé avec un front montant.

nControllerMode : SMC_CONTROLLER_MODE

Mode de régulation souhaité : SMC_torque (couple), SMC_velocity (vitesse), SMC_position (position), SMC_current (courant)

Entrées / sorties (VAR_IN_OUT) du module :

Axis : AXIS_REF (VAR_IN_OUT)

Entraînement dont le mode de régulation doit être modifié.

Sorties (VAR_OUTPUT) du module :

bDone : BOOL (VAR_OUTPUT)

TRUE dès que l'action est exécutée.

bError : BOOL (VAR_OUTPUT)

TRUE si une erreur survient.

nErrorID : SMC_Error (VAR_OUTPUT)

Décrit l'erreur.

2.2.5 Saisie directe de valeurs de consigne

SoftMotion donne à l'utilisateur la possibilité de saisir directement des valeurs de consigne pour la position, la vitesse et le rapport couple / puissance (voir SMC_SetTorque) et d'écrire ces valeurs sur les axes. Ces modules peuvent p.ex. être utilisés pour écrire des profils auto-calculés sur les axes. Un autre domaine d'application important est la commande CNC pour laquelle les valeurs de consigne de X/Y/Z doivent d'abord parcourir un module de transformation et être converties en valeurs de

consigne pour les axes individuels. Afin d'envoyer ces valeurs de consigne aux axes et le cas échéant procéder à une surveillance des valeurs limites, on dispose des modules ci-dessous.

Les modules SMC_FollowPosition/Velocity écrivent les valeurs de consigne définies sans autre contrôle sur la structure de l'axe. Le module SMC_CheckLimits vérifie les valeurs de consigne actuelles quant au respect des limites.

Les modules SMC_ControlAxisByPos/Vel sont spécialement conçus pour des applications CNC et offrent les fonctionnalités suivantes : Tant que l'on reste endéans les limites, ces modules fonctionnent de la même manière que SMC_FollowPosition/Velocity. Si cependant un dépassement de valeur de consigne survient (p.ex. suite à l'exécution séquentielle de deux programmes CNC dans lesquels les points de fin et de départ ne coïncident pas), ils arrêtent l'interpolateur, le mettent en place à la position de consigne actuelle et lèvent le barrage, permettant à l'interpolateur de poursuivre son déplacement.

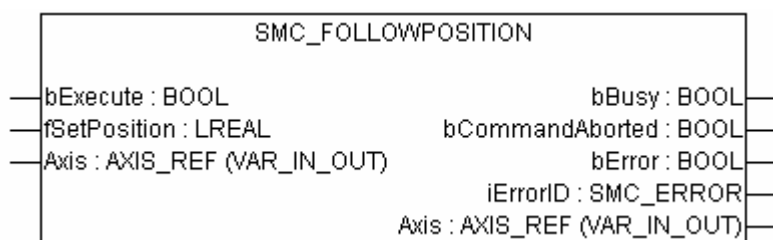
Ces modules sont ainsi conçus pour compenser les dépassements des positions de consigne. Ils constatent un tel dépassement en contrôlant le respect des maxima (de vitesse et d'accélération). Le dépassement des limites peut également survenir lorsque la vitesse du programme CNC ou de la valeur d'override sur l'interpolateur est trop élevée, ou si les valeurs maximum des axes sont trop faiblement réglées dans la configuration de commande ou ne le sont pas du tout. Il en résulterait un déplacement marqué par des à-coups vu que dès que la vitesse de la trajectoire est si élevée qu'un groupe d'axes est dépassé, l'interpolateur est toujours freiné et les axes sont déplacés sur la position correcte. L'interpolateur redémarrerait alors pour être à nouveau immédiatement freiné.

Les modules ci-dessous permettent de commander un entraînement par la saisie directe de ses valeurs de consigne:

- SMC_FollowPosition
- SMC_FollowPositionVelocity
- SMC_FollowVelocity
- SMC_CheckLimits
- SMC_ControlAxisByPos
- SMC_ControlAxisByPosVel
- SMC_ControlAxisByVel

SMC_FollowPosition

Ce module écrit les valeurs de consigne pour la position sur l'axe en ne procédant à aucun contrôle.



Entrées (VAR INPUT) du module :

bExecute: BOOL

Avec un front montant, le module commence à écrire sur la structure de données d'axe les valeurs de consigne saisies dans fSetPosition. Ce module reste actif jusqu'à ce qu'un autre module (p.ex. MC_Stop) vienne l'interrompre.

fSetPosition : LREAL

Position de consigne en unités techniques

Sorties (VAR OUTPUT) du module :**Busy : BOOL** (valeur par défaut : FALSE)

Si cette sortie est TRUE, le module est activé et écrit les valeurs de consigne sur l'axe.

bCommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, le module est interrompu par un autre module.

bError : BOOL (valeur par défaut : FALSE)

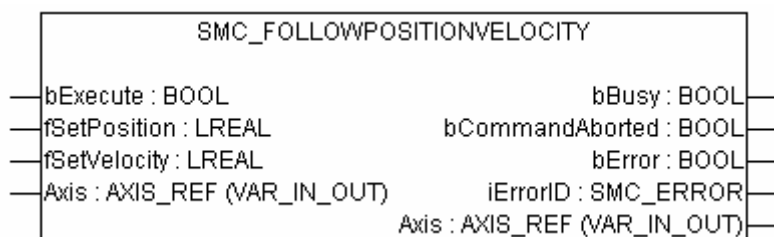
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

iErrorID : SMC_Error (INT)

Numéro d'erreur

SMC_FollowPositionVelocity

Ce module écrit les valeurs de consigne pour la position et la vitesse sur l'axe en ne procédant à aucun contrôle. L'utilisateur doit veiller à ce que les valeurs soient compatibles entre elles.

Entrées (VAR INPUT) du module :**bExecute : BOOL**

Avec un front montant, le module commence à écrire sur la structure de données d'axe les valeurs de consigne saisies dans fSetPosition. Ce module reste actif jusqu'à ce qu'un autre module (p.ex. MC_Stop) vienne l'interrompre.

fSetPosition : LREAL

Position de consigne en unités techniques

fSetVelocity : LREAL

Position de consigne en unités techniques

Sorties (VAR OUTPUT) du module :**Busy : BOOL** (valeur par défaut : FALSE)

Si cette sortie est TRUE, le module est activé et écrit les valeurs de consigne sur l'axe.

bCommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, le module est interrompu par un autre module.

bError : BOOL (valeur par défaut : FALSE)

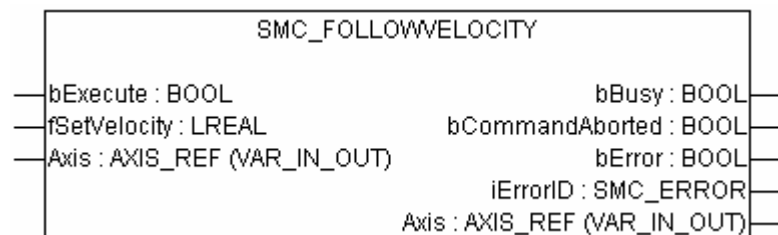
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

iErrorID : SMC_Error (INT)

Numéro d'erreur

SMC_FollowVelocity

Ce module écrit les valeurs de consigne pour la vitesse sur l'axe en ne procédant à aucun contrôle.



Entrées (VAR INPUT) du module :

bExecute : BOOL

Avec un front montant, le module commence à écrire sur la structure de données d'axe les valeurs de consigne saisies dans fSetPosition. Ce module reste actif jusqu'à ce qu'un autre module (p.ex. MC_Stop) vienne l'interrompre.

fSetVelocity : LREAL

Position de consigne en unités techniques

Sorties (VAR OUTPUT) du module :

Busy : BOOL (valeur par défaut : FALSE)

Si cette sortie est TRUE, le module est activé et écrit les valeurs de consigne sur l'axe.

bCommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, le module est interrompu par un autre module.

bError : BOOL (valeur par défaut : FALSE)

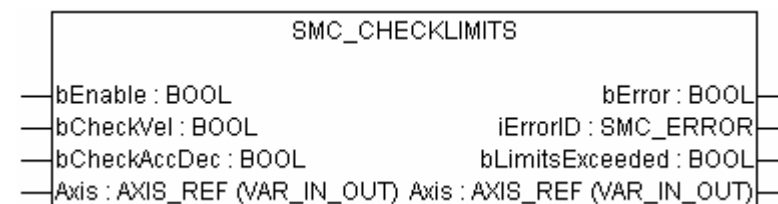
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

iErrorID : SMC_Error (INT)

Numéro d'erreur

SMC_CheckLimits

Ce module vérifie si la valeur de consigne actuelle de l'entraînement ne dépasse pas le maximum défini au sein de la commande et signale ceci par le biais de la sortie bLimitsExceeded.



Entrées (VAR INPUT) du module :

bEnable : BOOL

Le contrôle est effectué tant que bEnable est activé.

bCheckVel : BOOL

Si cette entrée est activée, la vitesse de consigne est contrôlée.

bCheckAccDec : BOOL

Si cette entrée est activée, l'accélération / la décélération de consigne est contrôlée.

Sorties (VAR OUTPUT) du module :

bError : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

iErrorID : SMC_Error (INT)

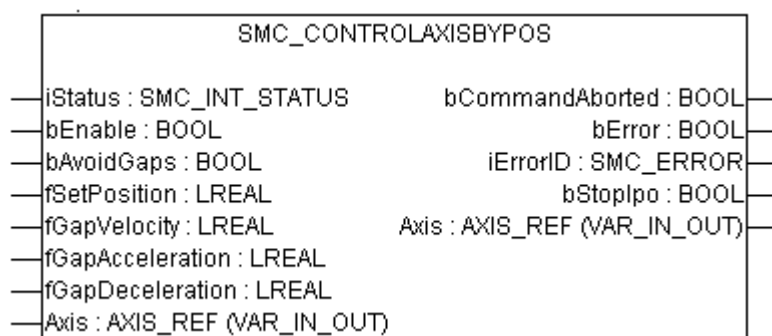
Numéro d'erreur

bLimitsExceeded : BOOL (valeur par défaut : FALSE)

TRUE signifie que les valeurs de consigne actuelles définies dans la configuration de la commande et enregistrées dans l'axe sous fSWMaxVelocity, fSWMaxAcceleration ou fSWMaxDeceleration sont dépassées.

SMC_ControlAxisByPos

Ce module écrit les positions de consigne dans une structure d'entraînement AXIS_REF et surveille cette dernière quant à des sauts.

Entrées du module :

iStatus : SMC_INT_STATUS

Statut du module d'interpolation. Cette entrée est liée à la sortie du même nom du SMC_Interpolator.

bEnable : BOOL

Contrôle de l'axe tant que l'entrée est TRUE.

bAvoidGaps : BOOL

TRUE : Le module surveille la position et la vitesse. Si la vitesse dépasse la valeur maximum enregistrée sur l'axe fSWMaxVelocity (et configurée dans le dialogue d'entraînement sous „Valeurs maximum“), le module active la sortie bStoplpo et commande l'axe selon les paramètres fGapVelocity, fGapAcceleration et fGapDeceleration en cette position, il supprime ensuite la sortie bStoplpo.

fSetPosition : LREAL

Position de consigne de l'axe. Il s'agit normalement d'une sortie du module de transformation.

fGapVelocity, fGapAcceleration, fGapDeceleration : LREAL

Paramètres de déplacement pour le pontage d'un saut.

Sorties du module :

bCommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, le module est interrompu par un autre module.

bError : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

iErrorID : SMC_Error (INT)

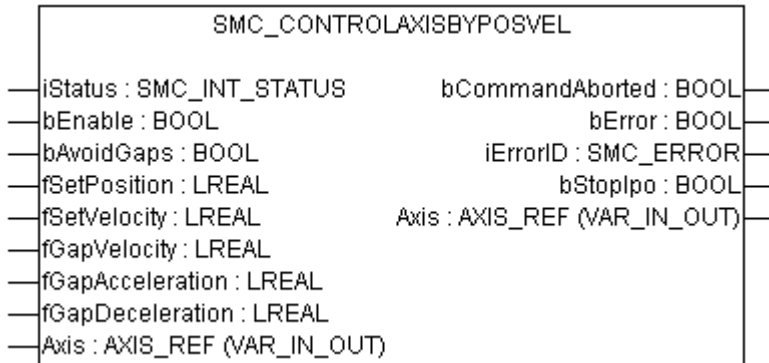
Numéro d'erreur

bStoplpo : BOOL (valeur par défaut : FALSE)

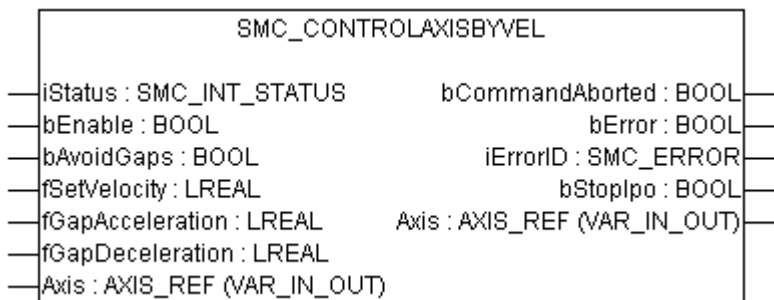
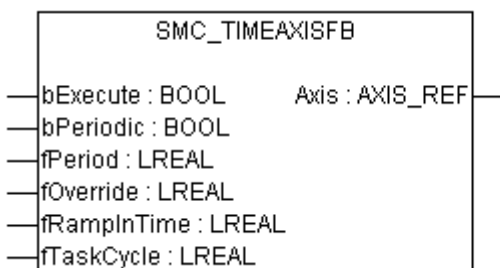
TRUE indique que le module a détecté un saut de vitesse ou de position et se déplace sur la nouvelle position. C'est la raison pour laquelle cette sortie doit être liée à l'entrée EmergencyStop du SMC_Interpolator de sorte que l'interpolateur attende que l'axe se trouve à la position correcte.

SMC_ControlAxisByPosVel

Ce module fonctionne de manière similaire au module SMC_ControlAxisByPos, à cette différence près qu'il faille également définir la vitesse.

**SMC_ControlAxisByVel**

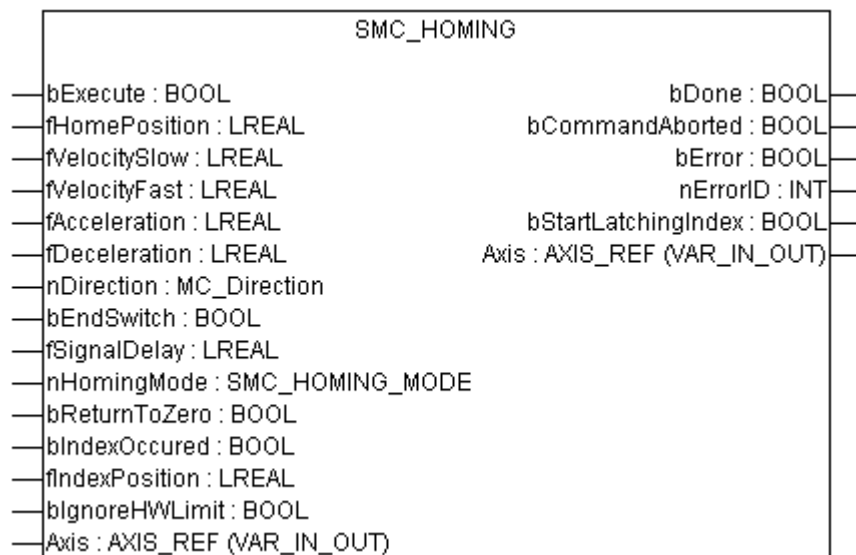
Ce module fonctionne de manière similaire au module SMC_ControlAxisByPos, à cette différence près que l'axe est commandé non pas par la position, mais bien par la vitesse.

**2.2.6 Axe temps virtuel**

Ce module fonctionnel de la bibliothèque SM_DriveBasic.lib crée un axe de temps qu'il met à disposition à la sortie Axis (AXIS_REF). Avec un front montant au niveau de l'entrée *bExecute*, la position de consigne de l'axe de temps commence à compter vers le haut, en secondes et en partant de 0. Si l'entrée *bPeriodic* est activée, on recommence à 0 dès que l'on atteint la durée *fPeriod*. L'entrée *fOverride* indique un multiplicateur de temps réglé par défaut sur 1. Un 2 entraîne un écoulement deux fois plus rapide du temps. *fRampInTime* permet de définir le temps dont le module dispose suite à la reprise des nouvelles valeurs de consigne de rampe pour le nouvel Override.

L'entrée *fTaskCycle* contient le temps de cycle (en secondes) de la tâche à partir de laquelle le module est appelé.

2.2.7 Référencement via entrées numériques de matériel hardware



Ce module fonctionnel permet de procéder à la course de référence d'un axe. L'interrupteur est ici une valeur BOOLEenne qui est normalement une Hardware_Input.

Si le module est lancé avec un front montant dans bExecute, il déplace l'axe à vitesse fVelocityFast dans le sens spécifié par nDirection, jusqu'à ce que bEndSwitch soit FALSE et donc que l'interrupteur de référencement soit fermé. L'axe est alors freiné et commandé à vitesse fVelocitySlow dans le sens opposé. La position de référence (fHomePosition) est définie à l'endroit où l'interrupteur de référencement s'ouvre (bEndSwitch = TRUE) et l'entraînement est stoppé.

Entrées du module :

bExecute : BOOL (valeur par défaut : FALSE)

La course de référence de l'entraînement doit être lancée avec un front montant.

fHomePosition : REAL

Mention de la position absolue sur la position de référence [u].

fVelocitySlow, fVelocityFast : REAL

Vitesses de consigne pour les phases 1 et 2 en [u/s].

fAcceleration, fDeceleration : REAL

Accélération de consigne et décélération en [u/s²].

nDirection : MC_Direction (valeur par défaut : négative)

Direction de la course de référence : valeurs admissibles : positive / négative.

bEndSwitch : BOOL (valeur par défaut : TRUE)

Interrupteur de référencement : TRUE (ouvert), FALSE (fermé).

fSignalDelay : REAL (valeur par défaut : 0.0)

Temps de transmission de l'interrupteur de référencement en s. Si un temps >0 est saisi, le module n'utilise pas comme position de référence la position à laquelle bEndSwicth est devenu TRUE mais bien la position que l'axe avait fSignalDelay secondes auparavant.

nHomingMode : SMC_HOMING_MODE (valeur par défaut : FAST_BSLOW_S_STOP)

FAST_BSLOW_S_STOP:

L'entraînement est commandé dans le sens indiqué selon une vitesse fVelocityFast (FAST) jusqu'à ce que l'entrée bEndSwich devienne FALSE, il est ensuite arrêté puis commandé à vitesse fVelocitySlow dans le sens opposé (BSLOW) jusqu'à ce que bEndSwitch redevienne TRUE. Le point de référence (S) est défini en cet endroit et le processus est arrêté (STOP).

FAST_BSLOW_STOP_S:

À l'inverse de FAST_BSLOW_S_STOP, l'arrêt a lieu ici après la course libre puis le point de référence est défini.

FAST_BSLOW_I_S_STOP:

À l'inverse de FAST_BSLOW_S_STOP, une impulsion Index (bIndexOccured=TRUE) est définie après la course libre et on attend sa position fIndexPosition qui est alors définie comme point de référence. Un arrêt n'a lieu qu'après.

FAST_BSLOW_S_STOP/ FAST_BSLOW_STOP_S / FAST_BSLOW_I_S_STOP:

Ces modes fonctionnent exactement comme ceux décrits précédemment, à cette différence près que dès que l'interrupteur de référencement est atteint, la course n'est pas inversée mais se poursuit dans le même sens. Il convient de noter qu'avec ces modes, l'entrée blgnoreHWLimits doit être FALSE pour des raisons évidentes de sécurité.

bReturnToZero : BOOL (valeur par défaut : FALSE)

Si ce drapeau est défini, le module se positionne sur le point zéro dès que la procédure définie dans nHomingMode est terminée.

bIndexOccured : BOOL (valeur par défaut : FALSE)

Uniquement pour nHomingMode FAST_BSLOW_I_S_STOP: Indique si l'impulsion Index a eu lieu.

fIndexPosition : REAL (valeur par défaut : 0.0)

Uniquement pour nHomingMode FAST_BSLOW_I_S_STOP: Position verrouillée de l'impulsion Index.

blgnoreHWLimit : BOOL (valeur par défaut : FALSE)

Si cette entrée est TRUE, le contrôle du matériel hardware de la fin de course est désactivé. Utilisez cette option si vous souhaitez utiliser le même interrupteur physique pour la fin de course et l'interrupteur de référencement du matériel hardware.

Sorties du module :

bDone : BOOL (valeur par défaut : FALSE)

Si TRUE, l'entraînement est référencé puis mis à l'arrêt.

bCommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, la commande est interrompue par une autre commande.

bError : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

nErrorID : SMC_Error

Numéro d'erreur

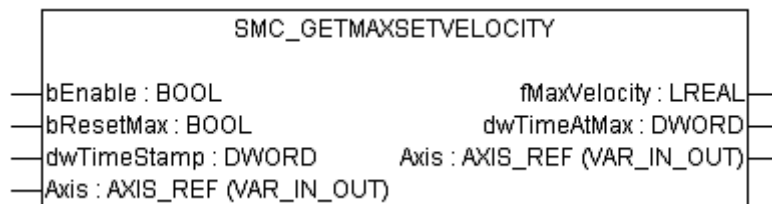
2.2.8 Modules de diagnostic

Les modules de diagnostic ci-dessous sont disponibles dans *SM_DriveBasic.lib* :

- SMC_GetMaxSetVelocity
- SMC_GetMaxSetAccDec
- SMC_GetTrackingError

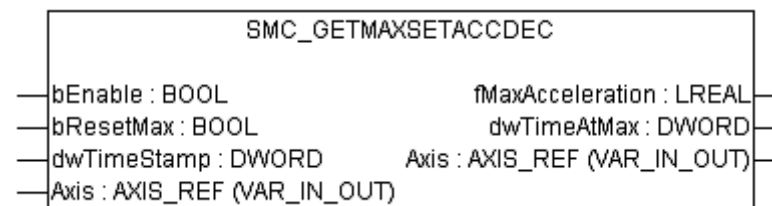
SMC_GetMaxSetVelocity

Ce module de diagnostic permet de mesurer la vitesse (de consigne) maximale d'un axe en termes de chiffres. La mesure est effectuée si bEnable est TRUE et ramenée à 0 tant que bResetMax est TRUE. dwTimeStamp permet de placer un DWORD au gré (p.ex. un compteur d'appels) qui peut être repris et édité avec une nouvelle valeur maximum.



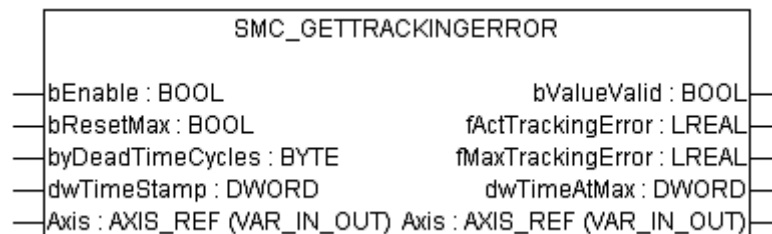
SMC_GetMaxSetAccDec

Ce module de diagnostic fonctionne de manière analogue à SMC_GetMaxSetVelocity et définit la plus grande accélération ou décélération en termes de chiffres.



SMC_GetTrackingError

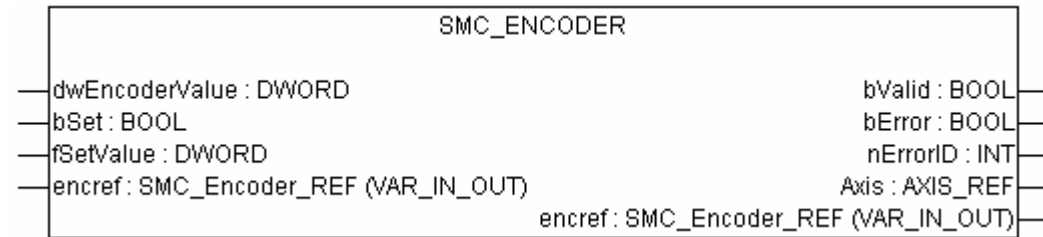
Ce module de diagnostic mesure les erreurs de poursuite actuelle et maximum en fonction du temps mort généré par la communication via un bus de terrain et signalé en nombre de cycles (byDeadTimeCycles). Comme pour SMC_GetMaxSetVelocity, il est possible d'utiliser un chronotimbre (dwTimeStamp) afin de mesurer le moment du maximum.



2.2.9 SMC_Encoder

La configuration de commande permet d'incorporer un encodeur dans un groupe d'axes et de l'y configurer. Les structures de données ainsi créées de type SMC_ENCODER_REF doivent être traitées par une instance du module SMC_Encoder. Ceci donne comme sortie une structure de données AXIS_REF qui, dès que la sortie bValid confirme la validité des données, peut servir d'entrée pour tous les autres modules fonctionnels (p.ex. MC_CamIn, MC_GearIn, MC_TouchProbe).

L'entrée booléenne bSet peut régler la valeur actuelle de l'encodeur sur l'entrée fSetValue.



2.2.10 Modèles de visualisation

Cette bibliothèque contient un modèle de visualisation pour les deux types d'entraînements (linéaire / rotatif), ce modèle peut être lié à une structure d'axe (AXIS_REF) et la position réelle de l'entraînement est visualisée :

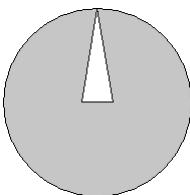
LinDrive



La figure ci-dessus s'applique à une entraînement linéaire. Le chariot est positionné conformément à sa position actuelle relative par rapport aux limites supérieure et inférieure et devient bleu dès qu'il se trouve dans la plage de régulation. Une condition pour l'utilisation de modèles est que les paramètres fSWLimitPositive et fSWLimitNegative soient activés.

Le modèle **LinDrive_V** représente l'entraînement sous forme verticale.

RotDrive



La figure ci-dessus s'applique à une entraînement rotatif. La position actuelle est représentée par la position de la flèche et devient bleue dès que l'entraînement se trouve dans la plage de régulation. Une condition pour l'utilisation de modèles est que le paramètre fPositionPeriod soit activé.

2.3 Pilotes d'entraînement <DésignationBusInterface>Drive.lib

- Les pilotes d'entraînement sont responsables de la communication entre le programme IEC, en particulier les *Structures AXIS_REF*, et les entraînements. Il s'agit de bibliothèques CoDeSys qui contiennent au moins les trois fonctions <DésignationBusInterface>DriveExecute_Start, <DésignationBusInterface>DriveInit et <DésignationBusInterface>DriveExecute_End. Ces bibliothèques sont normalement fournies par le fabricant des entraînements et doivent être intégrées au projet.
- DummyDrive.lib est un exemple de bibliothèque de pilote d'entraînement qui est fourni avec les bibliothèques SoftMotion. Même si cette bibliothèque ne commande pas d'entraînement proprement dit, elle fonctionne selon le même principe.

2.3.1 SercosDrive.lib

Cette bibliothèque qui utilise également la bibliothèque externe SercosBase.lib comme interface vers le matériel hardware permet de commander tous les entraînements compatibles Sercos.

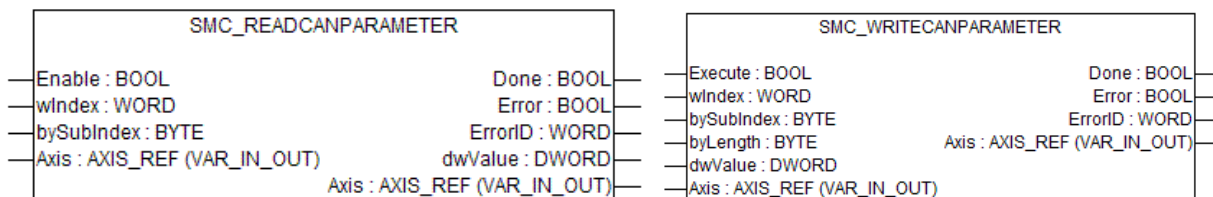
Comme pour CAN, il existe des blocs fonctionnels pour la lecture et pour l'écriture des paramètres :

- SMC_ReadSercosParameter
- SMC_WriteSercosParameter
- SMC_ReadSercosList
- SMC_WriteSercosList
- SMC_ReadSercosString

L'ampleur de la fonction est détaillée dans le document SercosDrive.pdf.

2.3.2 SM_CAN.lib

- À l'inverse de Sercos, on a besoin d'un pilote propre pour chaque entraînement CAN incorporé.
- On peut cependant définir pour tous les entraînements CAN – pour autant que ceci soit activé dans le fichier cfg – dans la configuration de commande (dialogue des groupes d'axes) le débit en bauds ainsi que le numéro du contrôleur CAN (à partir de 0). Afin de maintenir le bus déterministe, on exploite par canal CAN soit des E/S, soit des entraînements, mais jamais les deux ensemble. Si la bibliothèque 3S-CANopen est utilisée, celle-ci prend automatiquement le premier contrôleur CAN, permettant ainsi d'en réserver un autre pour les groupes d'axes.
- Toutes les bibliothèques CAN créées pas 3S reposent sur la bibliothèque SM_CAN.lib. Elle comprend deux modules qui s'avèrent pratiques pour l'utilisateur puisqu'ils permettent la lecture et l'écriture en toute simplicité des paramètres de l'entraînement : **SMC_ReadCANParameter** et **SMC_WriteCANParameter**. Leur mode de fonctionnement ressemble à celui des deux modules MC_ReadParameter et MC_WriteParameter :



Entrées des modules :

Enable : BOOL

TRUE : le paramètre est lu de manière continue.

Execute : BOOL

Front montant : l'écriture unique du paramètre est activée.

wIndex, bySubIndex : BYTE

Index et sous-index du paramètre qui doit être lu ou écrit.

Axis : AXIS_REF (VAR_IN_OUT)

Entraînement dont le paramètre doit être lu/écrit.

Sorties des modules :**dwValue : DWORD**

Valeur lue ou valeur à écrire.

Done : BOOL

TRUE : action correctement exécutée.

Error : BOOL

TRUE : une erreur est survenue.

ErrorID : WORD

Numéro d'erreur conformément à SMC_Error.

2.4 Variables de la structure AXIS_REF

Pour chaque entraînement créé dans la configuration de commande (Drive), CoDeSys crée lors de la compilation une variable structurelle **AXIS_REF** (définie dans la bibliothèque SM_DriveBasic.lib, voir chapitre 2.2). Cette structure forme l'interface entre l'application et l'interface de l'entraînement. Elle permet l'échange de données tant cycliques qu'acycliques.

La plupart des variables de la structure sont sans importance pour l'utilisateur, elles sont utilisées en interne par le système. L'utilisateur doit utiliser des blocs fonctionnels et ne jamais accéder directement à la structure, du moins pas en écriture.

N°	Nom	Type de donnée	Valeur init	Commentaire
1000	nAxisState	INT	standstill	Statut d'axe : 0: power_off 1: errorstop 2: stopping 3: standstill 4: discrete_motion 5: continuous_motion 6: synchronized_motion 7: homing
1001	wControlType	WORD	PLC-Config*	Chiffre qui indique les composants de la structure qui sont envoyés et reçus de manière cyclique. 0: défini par param 1002-1008 1: SetTorque 2: SetVelocity,ActVelocity 3: SetVelocity,ActPosition 4: SetPosition,ActPosition 5: SetVelocity,SetPosition,ActVelocity,ActPosition 6: SetVelocity
1002 1003	wCyclicDataS1 wCyclicDataS2	WORD	PLC-Config* ou Init-FB de <désignation	Nombre de paramètres 3S qui doivent être envoyés par cycle

N°	Nom	Type de donnée	Valeur init	Commentaire
1004	wCyclicDataS3		BusInterface>Drive.lib	
1006 1007 1008	wCyclicDataR1 wCyclicDataR2 wCyclicDataR3	WORD	PLC-Config* ou Init-FB de <désignation BusInterface>Drive.lib	Numéros des paramètres 3S qui doivent être envoyés par cycle
1010	bRegulatorOn	BOOL	FALSE	Activer / couper régulateur (puissance)
1011	bDriveStart	BOOL	FALSE	Activer / désactiver frein
1012	bCommunication	BOOL	FALSE	TRUE : l'entraînement répond
1013	wCommunicationState	WORD	0	À usage interne
1015	bRegulatorRealState	BOOL	FALSE	Statut du régulateur
1016	bDriveStartRealState	BOOL	FALSE	Statut du frein
1020	wAxisGroupId	WORD	PLC-Config*	Index du groupe d'axes dans la configuration
1021	wDriveId	WORD	PLC-Config*	Numéro de nœud d'entraînement sur le bus de terrain
1022	iOwner	INT	0	Numéro ID du propriétaire momentané (bloc fonctionnel)
1023	iNoOwner	INT	0	Nombre de propriétaires précédents et momentanés
1024	bMovedInThisCycle	BOOL	FALSE	L'entraînement a été déplacé au cours de ce cycle IEC
1025	fTaskCycle	REAL	PLC-Config*	Temps de cycle de tâche en s
1026	bRealDrive	BOOL	PLC-Config*	TRUE : créé par Config; FALSE : créé par programme IEC
1030	bError	BOOL	FALSE	Une erreur est survenue
1031	wErrorID	WORD	0	Numéro d'erreur
1032	bErrorAckn	BOOL	FALSE	Erreur confirmée
1035	FBErroID	WORD	0	Numéro d'erreur bloc fonctionnel
1051	dwRatioTechUnitsDenom	DWORD	PLC-Config*	Conversion des unités techniques en incréments : dénominateur
1040	fContrRpmP	REAL	1	Facteur d'amplification du régulateur de vitesse
1041	fContrRpmI	REAL	0	Durée d'action dérivée du régulateur de vitesse (ms)
1042	fContrRpmD	REAL	0	Durée de réajustement du régulateur de vitesse (ms)
1043	fContrPosP	REAL	1	Facteur d'amplification du régulateur de position
1044	fContrPosI	REAL	0	Durée d'action dérivée du régulateur de position (ms)

N°	Nom	Type de donnée	Valeur init	Commentaire
1051	dwRatioTechUnitsDenom	DWORD	PLC-Config*	Conversion des unités techniques en incréments : dénominateur
1052	iRatioTechUnits Num	INT	PLC-Config*	Conversion des unités techniques en incréments : numérateur
1053	nDirection	MC_Direction	positive	-1 : négatif (fSetVelocity < 0), 1 : positif
1054	fScalefactor	REAL	1	Conversion d'unités de bus en unités techniques, par unité reçue sur le bus
1055	fFactorVel	REAL	1	Conversion d'unités de bus en unités techniques / s
1056	fFactorAcc	REAL	1	Conversion d'unités de bus en unités techniques / s ²
1057	fFactorTor	REAL	1	Conversion d'unités de bus en Nm ou N
1058	fFactorJerk	REAL	1	Conversion d'unités de bus en unités techniques / s ³
1060	iMovementType	INT	1	0: 0: rotatif (modulo); 1: linéaire
1061	fPositionPeriod	REAL	1000	Longueur de période pour entraînement rotatifs en unités techniques
1091	byControllerMode	BYTE	3	1: Contrôle de couple 2: Contrôle de vitesse 3: Contrôle de position
1092	byRealControllerMode	BYTE	0	Mode de régulation effectif
1100/1	fSetPosition	REAL	0	Position définie en unités techniques
1101	fActPosition	REAL	0	Position actuelle en unités techniques
1105	fAimPosition	REAL	0	Position cible (pour quelques MC_FB)
1106	fMarkPosition	REAL	0	Repère interne de position
1107	fSavePosition	REAL	0	Position interne en début de cycle
1110,11	fSetVelocity	REAL	0	Vitesse définie en unités techniques / sec
1111,10	fActVelocity	REAL	0	Vitesse actuelle en unités techniques / sec
1112,9	fMaxVelocity	REAL	100	Vitesse maximale en unités techniques / sec

N°	Nom	Type de donnée	Valeur init	Commentaire
1113	fSWMaxVelocity	REAL	100	Vitesse maximale pour mouvements implicites en unités techniques / sec
1115	bConstantVelocity	BOOL	FALSE	L'axe se déplace à vitesse constante
1116	fMarkVelocity	REAL	0	Repère interne de vitesse
1117	fSaveVelocity	REAL	0	Vitesse interne en début de cycle
1120	fSetAcceleration	REAL	0	Accélération définie en unités techniques / sec ²
1121	fActAcceleration	REAL	0	Accélération actuelle en unités techniques / sec ²
1122,13	fMaxAcceleration	REAL	100	Accélération maximale en unités techniques / sec ²
1123	fSWMaxAcceleration	REAL	100	Accélération maximale pour mouvements implicites en unités techniques / sec
1125	bAccelerating	BOOL	FALSE	L'axe est en cours d'accélération
1126	fMarkAcceleration	REAL	0	Repère interne d'accélération
1127	fSaveAcceleration	REAL	0	Accélération interne en début de cycle
1130	fSetDeceleration	REAL	0	Décélération définie en unités techniques / sec ²
1131	fActDeceleration	REAL	0	Décélération actuelle en unités techniques / sec ²
1132,15	fMaxDeceleration	REAL	100	Décélération maximale en unités techniques / sec ²
1133	fSWMaxDeceleration	REAL	100	Décélération maximale pour mouvements implicites en unités techniques / sec
1135	bDecelerating	BOOL	FALSE	L'axe est en cours de décélération
1137	fSaveDeceleration	REAL	0	Décélération interne en début de cycle
1140	fSetJerk	REAL	0	Jerk défini en unités techniques / sec ³
1141	fActJerk	REAL	0	Jerk actuel en unités techniques / sec ³
1142,16	fMaxJerk	REAL	100	Jerk maximal en unités techniques / sec ³
1143	fSWMaxJerk	REAL	100	Jerk maximal pour mouvements implicites en unités techniques / sec
1146	fMarkJerk	REAL	0	Repère interne de jerk
1150	fSetCurrent	REAL	0	Intensité de courant définie (A)

N°	Nom	Type de donnée	Valeur init	Commentaire
1151	fActCurrent	REAL	0	Intensité actuelle de courant (A)
1152	fMaxCurrent	REAL	100	Intensité maximale de courant (A)
1153	fSWMaxCurrent	REAL	0	Intensité de courant maximale pour mouvements implicites en unités techniques / sec
1160	fSetTorque	REAL	0	Couple défini en Nm ou N (linéaire)
1161	fActTorque	REAL	0	Couple actuel en Nm ou N (linéaire)
1162	fMaxTorque	REAL	0	Couple maximum en Nm ou N (linéaire)
1200,2	fSWLimitPositive	REAL	0	Limite de positionnement dans le sens positif en unités techniques
1201,3	fSWLimitNegative	REAL	0	Limite de positionnement dans le sens négatif en unités techniques
1202	fCaptPosition	REAL	0	Position de capture en unités techniques
1204	bSWEndSwitchActive	BOOL	FALSE	Fin de course logicielle activée
1205	bSWLimitEnable	BOOL	FALSE	Activer fin de course logicielle
1206	bHWLimitEnable	BOOL	FALSE	Activer / désactiver fin de course matérielle
1207	bCaptureOccurred	BOOL	FALSE	Une capture a eu lieu (conformer par écriture)
1208	bStartCapturing	BOOL	FALSE	Lancer / terminer capture pour déclencheur actuel
1210	bStartReference	BOOL	FALSE	TRUE : lancer course de référence
1211	fReference	REAL	0	Position de référence
1215	fOffsetPosition	REAL	0	Décalage de point zéro
1220	fFirstCapturePosition	REAL	0	Fenêtre de position de départ pour capture
1221	fLastCapturePosition	REAL	0	Fenêtre de position de départ pour capture
1222	tiTriggerInput	TRIGGER_REF		Description de l'entrée capture
1223	bCaptureWindowActive	BOOL	FALSE	Capture limitée à la fenêtre
1230	dwPosOffsetForResiduals	DWORD	0	Variable interne pour traitement valeur résiduelle
1231	dwOneTurn	DWORD	0	Variable interne pour traitement valeur résiduelle
1232	fLastPosition	REAL	0	Variable interne pour traitement valeur résiduelle
1234	iRestNumerator	INT	0	Variable interne pour traitement valeur résiduelle

N°	Nom	Type de donnée	Valeur init	Commentaire
1235	iTurn	INT	0	Variable interne pour traitement valeur résiduelle
1236	dwBusModuloValue	DWORD	0	Variable interne pour traitement valeur résiduelle
1237	dwPosOffsetForResidualHoming	DWORD	0	Variable interne pour traitement valeur résiduelle
1300	bDisableDriveInAxisGroup	BOOL	FALSE	Supprimer entraînement de groupe d'axes
1301	bErrorDuringStartup	BOOL	FALSE	Activé si erreur en cours de lancement
	pMS	POINTER TO BYTE	0	Pointeur sur structure spécifique d'axe <Désignation BusInterface>_AXIS_REF

Chaque variable structurelle AXIS_REF se comporte conformément à la spécification PLCopen „Function blocks for motion control“, Version 1.0.

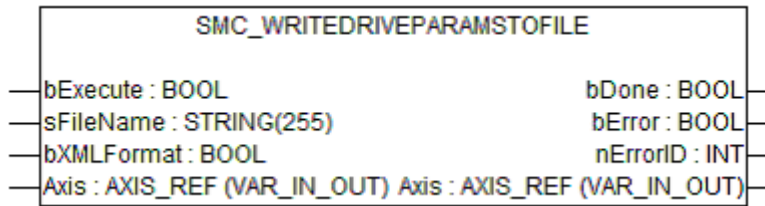
2.5 Paramétrage de l'entraînement

De nombreuses données importantes de configuration sont enregistrées dans l'entraînement. Bien que SoftMotion permette la saisie de valeurs de paramètres dans la configuration de la commande, celles-ci étant transmises dans la phase de lancement (voir 2.1.3), cette possibilité s'avère laborieuse pour l'utilisateur normal car il ne sait en général pas quels paramètres doivent être transmis ni quelle valeur ces paramètres doivent avoir. Ainsi, les outils spécifiques aux entraînements sont souvent requis pour la première mise en service (même de plusieurs machines de série identique) et pour le remplacement des entraînements. SoftMotion intègre à cet égard une fonction qui tient compte au moins partiellement de cet état de fait.

C'est pourquoi les variantes suivantes sont possibles:

- A. L'utilisateur configure l'entraînement à l'aide de l'outil du fabricant. Il ouvre une session en ligne avec un projet SoftMotion dans lequel les entraînements sont enregistrés et appelle un module qui lit tous les paramètres des entraînements et les enregistre dans un fichier XML ; l'utilisateur consulte ce fichier XML et saisit les données dans le dialogue approprié de configuration. Tous les paramètres nécessaires sont alors **enregistrés dans le projet** et ils sont transmis à chaque démarrage. L'outil de mise en service n'est utilisé ni pour la mise en service d'autres machines, ni pour le remplacement d'un entraînement défectueux.
- B. L'utilisateur configure l'entraînement à l'aide de l'outil du fabricant. Il prévoit dans l'application qu'il peut enregistrer les paramètres de l'entraînement dans un fichier ASCII **sur la commande**, cela par le biais d'un bloc fonctionnel. À l'inverse de A, les paramètres sont enregistrés dans un fichier sur la commande et non pas au sein de l'application.

Les deux variantes présentent des avantages et des désavantages. Pour modifier ultérieurement un paramètre avec la variante A, il faut charger à nouveau le projet (téléchargement). Cela est par contre possible avec la variante B, mais il faut savoir qu'à chaque nouvelle mise en service, il faut soit enregistrer également les fichiers de paramètres sur la commande, soit utiliser l'outil de mise en service de l'entraînement.

SMC_WriteDriveParamsToFile

Ce module lit tous les paramètres de configuration de l'entraînement et les enregistre dans un fichier. Comme il procède à un accès à un fichier qui peut bloquer pendant quelques ms l'exécution de l'application, il ne peut en général pas être appelé au sein de la tâche Motion mais doit être exécuté dans une tâche moins prioritaire.

Le pilote d'entraînement communique au module les paramètres qui doivent être lus ; soit l'entraînement a communiqué au pilote ces paramètres (= Sercos), soit ce dernier contient une liste de paramètres standard (via CAN : voir liste de variables globales du pilote standard 3S). Avec des entraînements CAN, il est possible de créer une propre liste de format identique et de la communiquer à l'entraînement via l'affectation ci-dessous :

```
<Drive>_MS.pParameterlist := ADR(<NouvList>);
```

Entrées / sorties du module :**bExecute : BOOL**

Le module est démarré avec TRUE.

sFileName : STRING(255)

Nom de fichier.

bXMLFormat : BOOL

Si cette variable est TRUE, le fichier est créé en format XML (et peut ensuite être importé dans le dialogue de l'entraînement) ; il est sinon créé en format texte (ce dernier est disponible sur la commande et peut être lu et envoyé par le pilote d'entraînement lors du lancement).

Axis : AXIS_REF

Entraînement dont les paramètres doivent être lus.

bDone : BOOL

Action terminée.

bError : BOOL

une erreur est survenue.

nErrorID : INT

Numéro d'erreur

3 L'éditeur CNC dans CoDeSys

3.1 Éditeur CNC - Aperçu

L'éditeur CNC dans CoDeSys permet de programmer des déplacements pluridimensionnels à la fois sous forme de graphique et de texte, conformément au langage CNC DIN66025.

Voir à ce sujet les chapitres: 3.4: Editeur de texte, 3.5: Editeur graphique, ainsi 1: Exemples de programmation.

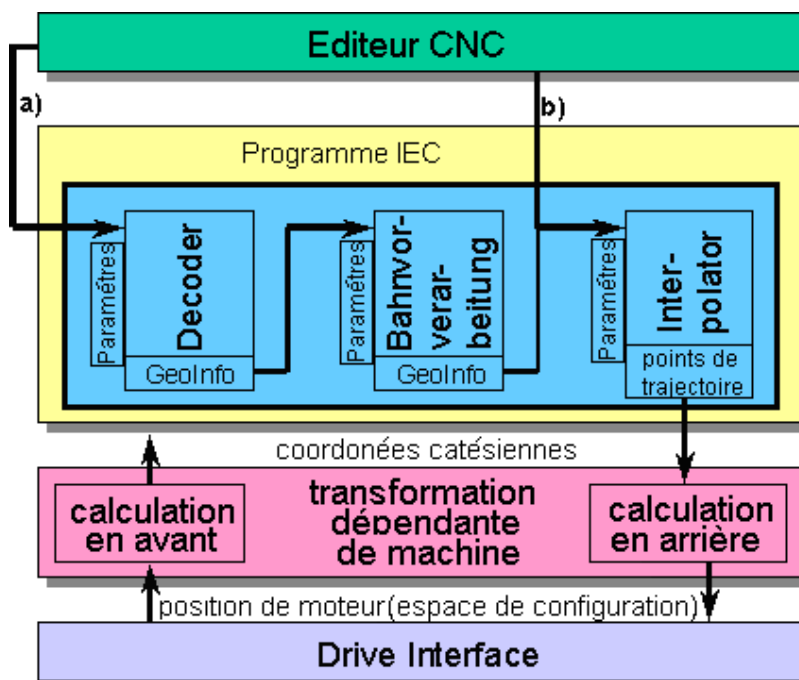
Il est en principe possible de réaliser des mouvements à max. 9 dimensions, et seulement deux dimensions peuvent être interpolées de manière autre que linéaire. Cela signifie que sur deux dimensions, il est possible de programmer des lignes droites, des cercles, des arcs de cercle, des paraboles et des splines; les autres directions ne sont interpolées que de manière linéaire.

Pour chaque trajectoire créée, CoDeSys génère automatiquement une structure de données globales (**données CNC**) qui sont à disposition dans le programme IEC. Ceci se passe de plusieurs façons :

(a) Le programme CNC est enregistré comme un **tableau à partir de mots de code G** et est décodé au cours de l'exécution du programme SPS à l'aide d'un module décodeur, de sorte que des objets structurels GEOINFO soient à disposition pour chaque objet de la trajectoire. Ces objets peuvent être édités à l'aide de modules de préparation de trajectoire (voir SM_CNC.lib, p.ex. correction du rayon d'outil), ensuite interpolés, transformés et enfin à nouveau transmis à l'interface d'entraînement à des fins de communication avec le matériel hardware. Voir à ce sujet les modules de SM_CNC.lib, chapitre 0.

(b) Le programme CNC est écrit sous la forme d'une **liste (structure OUTQUEUE) d'objets structurels GEOINFO** dans une structure de données et peut ainsi être saisi directement dans l'interpolateur. Comparée à a), cette deuxième possibilité permet de se passer des modules fonctionnels de décodeur et de préparation de trajectoire. En revanche, on ne dispose pas de la possibilité de modifier le programme en cours d'exécution.

(c) Le programme CNC est écrit dans le **système de fichier** de la commande sous la forme (a) ou (b) et est lu et converti pas à pas en cours d'exécution. Cette méthode convient surtout pour des grands programmes qui ne peuvent pas être complètement enregistrés dans la mémoire.



3.2 Éléments supportés et étendus du langage CNC DIN 66025

Afin de proposer au programmeur une solution toute simple pour créer une trajectoire géométrique, SoftMotion supporte en partie le langage CNC DIN66025. Comme le concept global SoftMotion est intégré dans le puissant langage IEC61131, **les seules parties supportées de DIN66025 sont celles qui servent à la création d'une trajectoire.**

Structure prescrite d'un programme CNC :

```
%
N<zahl> G<zahl> ...
...
N<zahl> G<zahl> ...
```

Exemple :

```
%1 example
N10 G01 X100 Y100 E100 F100 E-200
N20 G01 Z40 F20
N30 G03 X-100 R200 F100
...
```

Un programme CNC SoftMotion doit commencer par un pourcentage %. La même ligne peut contenir – séparé par un espace ou un tabulateur – un **nom de programme**. Le programme CNC lui-même se compose de plusieurs *jeux de données*.

Chaque **jeu** (ligne) se compose de **mots en nombre indéfini**.

Un mot se compose d'une lettre (**identificateur de mot**) suivie d'un chiffre (p.ex. G01 ; voir également liste ci-dessous). L'emploi de majuscules et de minuscules ne joue aucun rôle, de même que les zéros en tête (G01 = G1).

Le premier mot de chaque jeu forme le **numéro de jeu** (N<nombre>), p.ex. „N01“, qui se compose de la lettre du jeu et du nombre du jeu. Les mots d'un jeu sont séparés par des espaces ou des tabulateurs. Le numéro de jeu n'a pour l'instant aucune signification mais est exigé pour des raisons de conformité. Les autres mots au sein d'un jeu sont exécutés de droite à gauche. À cet égard, tous les mots sauf les **commandes de déplacement** (G<nombre>, p.ex. „G02“ ; voir également liste ci-dessous) entraînent l'écriture du nombre du mot dans une variable en fonction de la lettre de ce mot à laquelle la commande de déplacement accède.

Chaque jeu ne peut contenir qu'une commande de déplacement, celle-ci doit se situer directement à droite du numéro de jeu. Si la commande de déplacement n'est pas saisie dans un jeu, la commande utilisée dans le jeu précédent est utilisée à la place.

Chaque commande de déplacement est comprise comme un objet de trajectoire (ligne droite, arc de cercle, ...). La vitesse à laquelle les objets de la trajectoire sont interpolés résulte principalement des valeurs de consigne réglées pour la vitesse, l'accélération et la décélération. L'interpolateur doit veiller à ce que les valeurs limites ne soient pas dépassées. La vitesse lors de la transition entre deux objets voisins est définie selon les règles ci-dessous :

- Un des deux objets se rapporte à un positionnement (G0) : vitesse de transition = 0
- L'angle entre les tangentes des deux objets est supérieur à la tolérance d'angle au niveau de la transition : vitesse de transition = 0
- Autres cas : la vitesse de transition est la plus petite vitesse de consigne des deux objets de trajectoire.

Normalement, une commande de déplacement entraîne l'interpolation à partir de la position cible de la dernière commande de déplacement vers la position cible de la commande actuelle de déplacement. La première commande de déplacement débute à la position de départ spécifiée (au sein du décodeur ou de l'éditeur CNC). Si une telle position n'est pas définie, on débute à X=0, Y=0, Z=0. On a en outre la possibilité de définir une position via G92 dans le programme CNC. Ceci est possible tant au début du programme CNC (entraînant ainsi la définition de la position de départ)

qu'au milieu du programme, entraînant dans ce dernier cas un saut de la position de consigne vers la position définie via G92. Si plusieurs commandes G92 se suivent, la dernière prévaut ; les précédentes sont ainsi ignorées. Si on veut cependant également garantir que les positions G92 précédentes soient éditées (le temps d'un cycle), il suffit d'insérer entre elles la commande G1 avec les mêmes coordonnées. Cette méthode est utilisée lorsque la trajectoire entre ces points n'est pas en soi intéressante, mais bien le fait de parvenir à la position de consigne le plus rapidement possible. Les modules SMC_ControlAxisByPos détectent alors un saut dans les valeurs de consigne, arrêtent l'interpolateur et y interpolent chaque axe séparément le plus rapidement possible. Exemple :

G92 X100 Y100 (définir position de consigne à 100/100)

G1 X100 Y100 (garantir l'édition unique de la position)

G92 X50 Y100 (définir position de consigne à 50/100)

Un jeu commençant avec le symbole „/“ est sauté si la **suppression de jeu** est activée.

Les signes se trouvant entre parenthèses rondes sont des **commentaires** qui n'ont aucune influence sur la trajectoire. Les commentaires imbriqués ne sont pas supportés.

Toutes les valeurs chiffrées peuvent être des valeurs à virgule flottante, sauf dans le cas des commandes de déplacement (G<nombre>) et du numéro de point de commutation (H<nombre>).

Identificateurs de mots :

D	Rayon d'outil (pour correction G40-42 ou arrondir angle G50-51) ou valeur de variable (G36/G37)
E	Accélération (>0) / décélération (<0) max.[unités de trajectoire/sec ²]
F	Vitesse à laquelle le déplacement doit avoir lieu [unités de trajectoire/sec]
G	Commande de déplacement (voir ci-dessous)
H	Activer (>0) / désactiver (<0) point de commutation
I	Coordonnées X de point central de cercle / ellipse (G02/G03/G08/G09) ou coordonnées X de point d'intersection parabole tangente
J	Coordonnées Y de point central de cercle / ellipse (G02/G03/G08/G09) ou coordonnées Y de point d'intersection parabole tangente
K	Direction de l'axe principal d'ellipse dans le sens mathématique (0° O, 90° N,...) ou condition de saut (G20) ou valeur de paramètre dT1 (fonction M)
L	Position absolue de commutation mesurée du début de l'objet de trajectoire (>0) / de sa fin (<0) ou destination de saut (G20) ou valeur de paramètre dT2 (fonction M)
M	Fonction auxiliaire
O	Position relative de commutation [0..1] ou variable à modifier (G36/G37) ou structure de données de paramètre M (M)
P	Valeur cible de l'axe supplémentaire P
Q	Valeur cible de l'axe supplémentaire Q
R	Rayon de cercle (G02/G03) (alternative à I,J) ou rapport de longueur axe secondaire / axe principal (G08/G09)]0..1]
S	Activer (>0) / désactiver (<0) profil S pour axes linéaires 3: axe Z , 7: axe P , 8: axe Q, 9: axe U, 10: axe V, 11: axe W
U	Valeur cible de l'axe supplémentaire U
V	Valeur cible de l'axe supplémentaire V
W	Valeur cible de l'axe supplémentaire W

X	Coordonnées X de la cible
Y	Coordonnées Y de la cible
Z	Valeur cible de l'axe supplémentaire Z

Commandes de déplacement :

G00	Déplacement direct sans intervention d'outil, positionnement
G01	Déplacement linéaire (rectiligne) avec intervention d'outil
G02	Cercle (arc de cercle) dans le sens horaire
G03	Cercle (arc de cercle) dans le sens anti-horaire
G05	Point d'une spline cardinale
G06	Parabole
G08	Ellipse (section d'ellipse) dans le sens horaire
G09	Ellipse (section d'ellipse) dans le sens antihoraire
G20	Saut conditionné (vers L si K<>0)
G36	Écrire valeur (D) sur variable (O)
G37	Incrémenter variable (O) de la valeur (D).
G40	Fin de correction d'outil
G41	Début de correction de rayon d'outil à gauche dans le sens de déplacement
G42	Début de correction de rayon d'outil à droite dans le sens de déplacement
G50	Fin d'arrondi / de ponçage d'angle
G51	Début de ponçage d'angle
G52	Début d'arrondi d'angle
G60	Fin d'évitement de boucles
G61	Début d'évitement de boucles
G90	Indication absolue (standard) des coordonnées suivantes (pour X/Y/Z/P-W)
G91	Indication relative des coordonnées suivantes (pour X/Y/Z/P-W)
G92	Définition de position sans déplacement
G98	Indication absolue des coordonnées suivantes de I/J
G99	Indication relative (standard) par rapport au point de départ des coordonnées suivantes de I/J

Il convient de noter que la bibliothèque „SM_CNC.lib“ doit être intégrée afin de garantir la compilation correcte du projet.

3.2.1 Fonction de point de commutation, Fonction H

La fonction de point de commutation (ou fonction H) donne aux programmeurs CNC la possibilité d'utiliser des interrupteurs binaires en fonction de la trajectoire. En général, il faut toujours spécifier en premier le numéro du point de commutation (<numéro>H) pour définir ensuite la position du point de commutation dans l'objet soit de manière absolue via le mot <position>L, soit de manière relative via le mot <position>O. Dans l'exemple ci-dessous, le point de commutation2 est désactivé à la position X=40/Y=25 (après le premier quart de l'objet).


```
N90 G1 X20 Y20
N100 G1 X100 Y40 H-2 O0.25
```

Remarque : Il convient de noter que l'on ne peut effectuer qu'un nombre restreint d'actions de commutation (MAX_SWITCHES) par objet de trajectoire.

Une position de point de commutation ne peut être insérée que dans l'éditeur de texte CNC ! Elle s'affiche dans l'éditeur graphique sous la forme d'un point vert sur la courbe.

3.2.2 Fonction auxiliaire, Fonction M

Les fonctions auxiliaires ou fonctions M permettent de créer une sortie binaire dans le programme CNC, cette sortie démarrant une autre action. À l'inverse des points de commutation, on reste à la position actuelle jusqu'à ce que la fonction M soit confirmée par l'activation d'une entrée. Cette fonction est fréquemment utilisée lorsque la suite de l'exécution du programme dépend d'autres processus. La ligne ci-dessous démarre la fonction M 10 et attend que celle-ci soit confirmée :

```
N90 M10
```

Il est possible de saisir deux paramètres via K et L, lesquels sont copiés dans l'objets SMC_GeoInfo correspondant. L'application a alors la possibilité - en cours d'exécution et si l'interpolateur se trouve sur une fonction M et attend - d'accéder à ces valeurs via la module SMC_GetMParameters.

Le mécanisme décrit ci-dessous ne peut être utilisé que si la trajectoire est traitée en ligne (avec SMC_NCDDecoder):

Afin de transmettre des paramètres supplémentaires, on peut utiliser la structure de données globales gSMC_MParameters (type : SMC_M_PARAMETERS) créée dans la bibliothèque SM_CNC.lib. Celle-ci correspond aux paramètres dP1..dP8.

Si on souhaite créer soi-même une structure à la place de la structure de données globales gSMC_MParameters, on peut transmettre cette structure via O\$var\$. Par exemple, g_myMParams (type : SMC_M_PARAMETERS) contient les paramètres à transmettre :

```
N150 M13 O$g_myMParams$
```

3.2.3 Utilisation de variables

Lors de la programmation CNC, on peut utiliser des variables à la place de valeurs chiffrées fixes.

Les variables doivent se trouver entre deux symboles \$ (p.ex. R\$g_fVar\$). Il faut noter dans ce contexte que l'utilisation de variables ne fonctionne que si le programme est compilé comme variable de programme et peut être édité en ligne via le module de décodeur. Les variables sont remplacées au moment où le décodeur traite la ligne correspondante.

Si le programme CNC est par contre compilé comme une OutQueue, le mécanisme des variables ne fonctionne pas car la trajectoire est alors créée hors ligne et transmise à l'application sous la forme d'une structure de données ne pouvant pas être modifiée. Dans ce cas – comme pour l'affichage hors ligne –, l'éditeur remplace les variables par leurs valeurs initiales.

L'utilisation de variables dans des programmes de code G lus en ligne nécessite une préparation supplémentaire (voir SMC_VARLIST).

3.2.4 Interpolation circulaire

Pour décrire un arc de cercle, les coordonnées cible (X/Y) doivent être définies.

Il y a deux possibilités pour définir une courbe : on saisit soit le rayon (R) de l'arc de cercle soit les coordonnées de son point central (I/J).

Dans le cas de la variante de saisie du rayon, seules des courbes de moins de 180° (demi cercle) sont possibles, car cette méthode (sauf dans le cas du demi cercle) décrit toujours deux solutions

possibles, l'une plus petite et l'autre plus grande qu'un demi cercle. Le système choisit toujours celle qui est plus petite que le demi cercle.

Exemple pour un demi cercle :

N10 G1 X100 Y100

N20 G2 X200 Y100 R50

Si on souhaite un arc de cercle dont l'angle d'ouverture est supérieur à 180° , il faut le définir via la variante (I/J). Cette méthode est toujours univoque sauf si les points de départ et de fin du cercle sont identiques. On pourrait ainsi avoir en cet endroit un cercle nul ou un cercle complet. Dans un tel cas, le système insère un cercle complet.

Il faut également noter que cette méthode (I/J) définit si les coordonnées I/J (voir G98/G99) sont saisies de manière relative ou absolue. Si I et J ne sont pas saisis de manière correcte (la distance entre le centre et le point de départ de celui de fin doit être identique), aucun arc de cercle n'est possible et le système remplace celui-ci par une ligne droite.

Exemple pour le même arc de cercle défini via un point central relatif :

N10 G1 X100 Y100

N15 G99

N20 G2 X200 Y100 I50 J0

Exemple pour le même arc de cercle défini via un point central absolu :

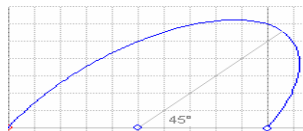
N10 G1 X100 Y100

N15 G98

N20 G2 X200 Y100 I150 J100

3.2.5 Interpolation d'ellipse

Pour décrire un arc ellipsoïdal, les coordonnées cible (X/Y), le point central de l'ellipse (I/J), la direction de l'axe principal de l'ellipse (K) et le rapport de longueur entre l'axe principal et l'axe secondaire (R) doivent être spécifiés.



Exemple :

N10 G0 X100 Y100

N15 G98

N20 G8 X200 Y100 I150 J100 K45 R0.5

3.2.6 Interpolation de spline

Afin d'écrire une spline, il suffit de saisir les coordonnées cible de la section de spline suivante. La section de spline est calculée automatiquement par le système de telle sorte que le vecteur final de l'élément précédent et le vecteur de départ de la spline coïncident entre eux, ainsi que le vecteur final de la spline et le vecteur de départ de l'élément suivant ; on n'a ainsi pas de coude dans la trajectoire.

3.2.7 Sauts conditionnels

La commande G20 permet d'effectuer un saut conditionné. Il faut pour ce faire saisir le numéro de ligne de destination du saut (L) ainsi que la condition pour le saut (K). Si aucune condition n'est donnée pour le saut, la variable Decoder implicite est automatiquement appliquée. Le saut est toujours exécuté si la condition est différente de zéro.

3.2.8 Modifier valeurs de variables

Les commandes G36 et G37 permettent à l'utilisateur d'écrire ou de modifier des variables. L'utilisateur saisit la variable qui doit être modifiée via O\$var\$. On définit via D la valeur à laquelle la variable doit être écrite (G36) ou ajoutée (G37).

Dans l'exemple suivant, la variable globale g_i est réglée sur 5 :

```
N1000 G36 O$g_i$ D5
```

L'exemple suivant fait que les lignes 1010 et 1020 sont parcourues cinq fois :

```
N1000 G36 O$g_i$ D5
```

```
N1010 G1 X100 F100 E100 E-100
```

```
N1020 G1 X0
```

```
N1030 G37 O$g_i$ D-1
```

```
N1040 G20 L1010 K$g_i$
```

Il convient de noter que ce mécanisme ne fonctionne **que si la trajectoire est travaillée en ligne** car c'est le seul moyen d'utiliser des variables ! Ce mécanisme ne fonctionne donc pas au sein de l'éditeur CNC. On peut à la place procéder comme suit.

Si aucune variable n'est spécifiée pour O, une variable implicite de décodeur (de type : INT) est utilisée. Ce mécanisme fonctionne également hors ligne dans l'éditeur. Il faut cependant noter qu'on ne dispose ainsi que d'une seule variable et que l'on ne peut pas programmer de sauts ni de boucles imbriqués.

Exemple :

```
N1000 G36 D5
```

```
N1010 G1 X100 F100 E100 E-100
```

```
N1020 G1 X0
```

```
N1030 G37 D-1
```

```
N1040 G20 L1010
```

3.3 Démarrage de l'éditeur, insertion et gestion des programmes CNC

L'éditeur CNC est lancé sous „Ressources“ au sein de l'organisateur d'objets. On accède alors à une fenêtre „**Liste de programmes CNC**“ divisée en trois parties. La colonne gauche affiche une liste avec les noms des programmes créés. La section supérieure droite de la fenêtre sert d'éditeur de texte, pour saisir le programme CNC conformément à la norme DIN66025. Ceci est représenté graphiquement en bas à droite et peut être modifié tant dans la zone de texte que dans celle de graphique, les changements étant alors automatiquement actualisés dans l'éditeur correspondant.

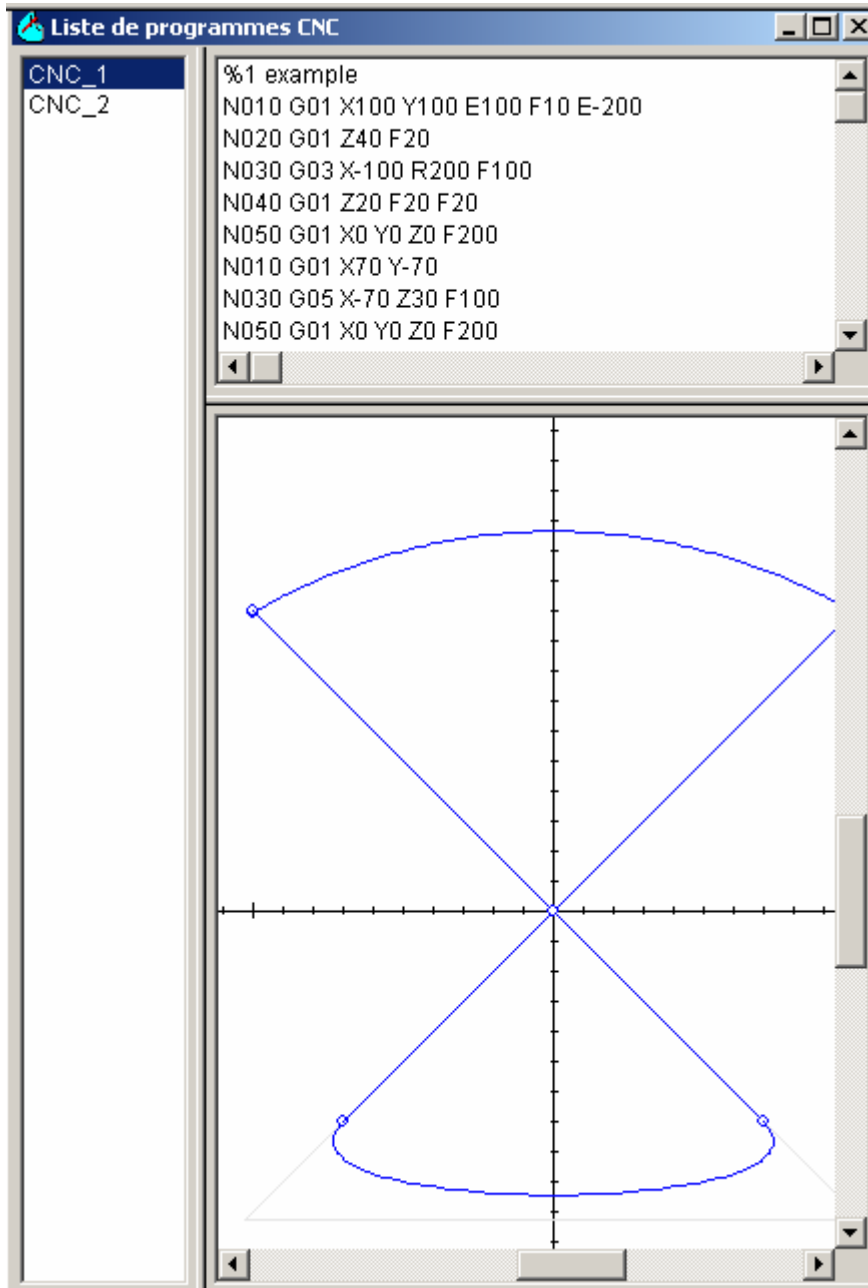
Dans la barre de menu, le menu „Insérer“ est remplacé par le menu „**Programme CNC**“.

Programme CNC : Nouveau programme CNC

Sélectionner la commande „**Nouveau programme CNC**“ dans le menu „Programme CNC“ ou dans le menu contextuel si l'accent est mis dans le Programme sur la colonne de liste CNC. On accède au dialogue „Nom du programme“ qui permet d'attribuer le nouveau nom de programme. De série, les programmes CNC sont pourvus de la désignation „_CNC<n>“, n représentant une énumération

continue commençant avec 1. Le nom de programme saisi au sein de ce dialogue peut être édité mais la saisie d'un nom déjà attribué n'est cependant pas acceptée. Suite à la fermeture du dialogue au moyen de OK, le nouveau programme apparaît marqué dans la liste. La première ligne de programme est affichée dans l'éditeur de texte : pour un nouveau programme „%comment“, l'éditeur graphique est alors encore vide.

Le programme actuellement marqué peut maintenant être édité tant avec l'éditeur de texte qu'avec l'éditeur graphique.



Programme CNC : Renommer programme CNC

Marquer le programme dans la liste des programmes Programme CNC puis sélectionner la commande **"Renommer programme CNC"** dans le menu "Programme CNC" ou dans le menu contextuel. On accède au dialogue "Nom du programme" permettant d'éditer le nom.

Programme CNC : Effacer programme CNC

Marquer le programme dans la liste des programmes Programme CNC puis sélectionner la commande „**Effacer**“ dans le menu "Programme CNC" ou dans le menu contextuel. Le programme est alors supprimé de la liste, l'accent est ensuite mis sur la liste suivante.

Programme CNC : Info

Marquer un programme dans la liste des programmes Programme CNC puis sélectionner la commande "Info" dans le menu "Programme CNC" ou dans le menu contextuel. On accède au dialogue qui contient des informations sur le programme CNC choisi.

La fenêtre **Informations sur le programme CNC** contient les données ci-dessous relatives au programme marqué dans la liste ad hoc :

Nom du programme, Lignes du programme, Objets de trajectoire, Longueur totale de trajectoire, Position de départ, p.ex. (X=1.50, Y=0.00), Tolérance d'angle, Durée du mouvement La fenêtre Erreur<nom de module> décrit toutes les erreurs dans le programme (p.ex. erreur IPO "Vitesse 0 ou accélération négative" décrit une erreur au sein du module IPO)

Programme CNC : Définir taille queue

Il est possible de définir ici la taille de la mémoire tampon des données OutQueue. Cette fonction s'avère intéressante si on souhaite proposer les modules fonctionnels NC dans le programme IEC avec une taille de mémoire limitée ; tous les objets GeolInfo n'y trouvent alors pas place et la fonctionnalité de mémoire tampon circulaire peut être utilisée. Des effets spéciaux peuvent alors survenir (p.ex. ralentir dans des zones sans courbe) et cette fonction vous permet de les simuler et de les comprendre.

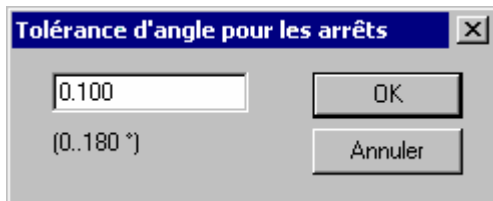
Valeurs possibles: 5000 – 100 000 octets. OK permet de reprendre les valeurs.

Programme CNC : Définir position de départ

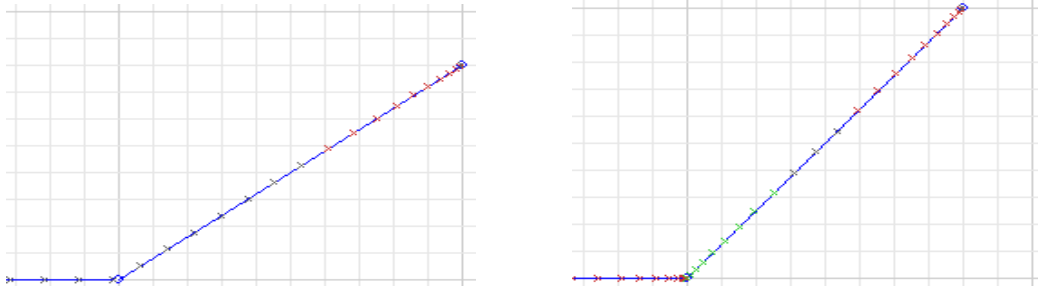
Le dialogue **Point de départ** permet de saisir une position de départ différente de la valeur standard de 0 pour la trajectoire, à des fins de simulation. Les saisies sont possibles pour les axes suivants : X, Y, Z, P, Q, U, V, W.

Programme CNC : Définir tolérance d'angle pour les arrêts

Le dialogue **Définir tolérance d'angle pour les arrêts** permet de régler la sensibilité au niveau d'un coude de la trajectoire. La saisie définit l'angle pour les tangentes de deux objets de trajectoire à partir duquel la trajectoire est interrompue:



Exemple: Tolérance d'angle 45° :



Programme CNC : Déplacer programme

Sélectionner cette option de menu puis saisir un **vecteur de mouvement** dans le dialogue Vecteur de mouvement. Le programme CNC actuel est déplacé selon ce vecteur. Pour les axes suivants, il est possible de saisir les grandeurs de vecteurs pour un déplacement : X, Y, Z, P, Q, U, V, W.

Programme CNC : Pivoter programme

Pour pivoter le programme actuel, il faut sélectionner cette option de menu puis saisir l'angle de rotation dans la dialogue **Angle de rotation**. Le programme NC est pivoté dans le sens anti horaire autour de son point zéro, selon l'angle saisi.

Programme CNC : Étendre programme

Définir le **facteur d'extension** dans le dialogue Facteur d'extension. Le Programme NC est adapté en fonction du facteur saisi.

Programme CNC : Éditeur CNC - Inverser la direction

Si vous sélectionnez cette option de menu, la trajectoire est modifiée en ce sens qu'elle est parcourue dans l'autre sens. Les positions de commutation restent les mêmes.

Programme CNC : Diviser l'objet

Si on a sélectionné un objet de trajectoire (marqué en rouge), il est possible de définir une position de division (0 = départ, 1 = fin) au sein de cet objet. Il est alors divisé en deux objets de trajectoire à l'endroit spécifié.

Exemple : Commande de déplacement N10 avec mention d'une position de division de 0,5 :

```
...
N0 G01 X123.000000 Y73.550000
N10 G01 X40.0 Y50.0
...
```

Résultat: nouvelle position de division à X=20

```
...
N0 G01 X20.000000 Y40.000000
N10 G01 X40.000000 Y50.000000
N20 G01 X123.000000 Y73.550000
...
```

La nouvelle position intermédiaire Y est à chaque fois adaptée en fonction du parcours de la trajectoire.

Programme CNC : Lire programme CNC du fichier

Cette option de menu vous permet de charger un programme NC enregistré dans un fichier ASCII. Le fichier TXT souhaité peut être sélectionné dans le dialogue standard d'ouverture de fichier. Saisir le nom du programme dans le dialogue apparaissant à l'écran.

Programme CNC : Écrire programme CNC dans fichier

Cette option de menu permet d'écrire le programme CNC actuel dans tout fichier ASCII. Si le fichier existe déjà, CoDeSys demande une confirmation.

Programme CNC : Importer un fichier DXF

Cette option de menu permet d'importer un fichier DXF. Le fichier DXF souhaité peut être sélectionné dans le dialogue standard d'ouverture de fichier. On accède alors au dialogue **Options d'importation DXF** dans lequel une des options ci-dessous peut être réglée :

- Un programme NC pour le fichier DXF complet : toutes les trajectoires contenues dans le fichier DXF sont écrites dans un programme CNC,
- Un programme NC par couche DXF : un programme CNC propre est créé par couche DXF
- Un programme NC par section cohérente : un programme CNC propre est généré pour chaque section cohérente. Comme les objets de trajectoire sont enregistrés sans succession définie dans un fichier DXF, CoDeSys tente de relier les objets entre eux afin de générer sur cette base une trajectoire cohérente.

Programme CNC : Écrire OutQueue dans fichier

Cette fonction permet de convertir tout le programme CNC en une OutQueue – une liste d'objet structuraux GEOINFO – et de l'enregistrer dans un fichier qui peut être chargé dans le système de fichiers de la commande et y être lu pour exécution. Cette fonction s'avère pratique pour des grands programmes CNC qui ne peuvent être contenus dans la mémoire des données globales de la commande ou qui doivent être échangés sans modifier le projet CoDeSys.

Programme CNC : Créer variable de programme en compilant

Cette option correspond à la variante a).

Le programme CNC est enregistré dans une structure SMC_CNC_REF contenue dans la bibliothèque SM_CNC.lib. Cette variable structurée doit être exécutée par les modules Decoder et de préparation de trajectoire, elle ne peut pas être saisie directement dans le module d'interpolateur.

Bien que cette variable nécessite plus de ressources en ligne, elle offre la possibilité d'utiliser des variables dans le programme NC, ou encore de procéder à des modulations du programme NC via le programme API.

Programme CNC : Créer OutQueue en compilant

Cette option correspond à la variante b).

Le programme CNC est enregistré dans une structure SMC_OUTQUEUE contenue dans la bibliothèque SM_CNC.lib. Cette variable structurée peut être saisie directement dans le module d'interpolateur.

Bien qu'aucune variable ne puisse ainsi être utilisée dans la trajectoire, cette méthode présente l'avantage selon lequel les ressources en ligne nécessaires sont réduites au minimum.

Programme CNC : Ne pas compiler

Cette option correspond au cas de figure (c). Le programme est disponible comme fichier ASCII ou OutQueue dans le système de fichiers de la commande ; ce programme doit être lu pour exécution

par un des modules de la bibliothèque SM_FileFBs.lib du chapitre 10.2. C'est pourquoi il ne doit pas être repris dans les données IEC.

3.4 Éditeur CNC littéral

L'éditeur littéral se situe dans la partie supérieure droite de la fenêtre de programme CNCliste. On peut y saisir et éditer un programme CNC conformément à la norme DIN66025. Voir à ce sujet : éléments supportés et étendus du langage CNC DIN66025). Le programme est représenté de manière appropriée dans l'éditeur graphique. S'il y est modifié voire créé, il est également automatiquement actualisé dans la partie texte.

Pour l'édition, on dispose des commandes des menus **Programme CNC** et **Extras**, voir également la description de l'éditeur Graphique.

Une pression sur F2 donne accès à la liste de sélection pour l'édition et permet de saisir des variables globales dans le programme CNC. Pour la représentation dans l'éditeur graphique – si disponible –, on utilise la valeur initiale de la variable.

Remarque: il faut noter que les renvois à des variables globales dans le module de décodeur sont évalués comme une réaction à un front montant dans l'entrée Execute.

Si on a choisi l'option selon laquelle CoDeSys crée une structure complète OutQueue lors de la compilation (variante b)), le renvoi à la variable globale sera naturellement déjà remplacé par la valeur initiale lors de la compilation, ce qui rend à cet égard l'utilisation de variables globales totalement illogique.

3.5 Éditeur CNC graphique

L'éditeur graphique se situe dans la partie inférieure droite de la fenêtre de programme CNCliste. Il permet d'une part de visualiser le Programme CNC créé au sein de l'éditeur de texte, et d'autre part de procéder à l'aide de la souris à des modifications de ce programme, ces dernières étant automatiquement et immédiatement ajoutées au texte de Programme dans l'éditeur de texte.

(Voir à ce sujet également les exemples de programmation)

Visualisation :

Un système de coordonnées est défini dans lequel des repères sont donnés à intervalles réguliers. On peut en complément afficher une **grille** en gris clair (**Montrer grille** dans le menu Extras ou dans le menu contextuel).

En maintenant le bouton gauche de la souris enfoncé, il est possible de **déplacer** la représentation du programme NC. La roulette de la souris et la touche <Ctrl> permettent de modifier le facteur de **zoom**.

Les **positionnements** (G00) sont représentés en vert (comme les positions de commutation) tandis que tous les autres **éléments** sont en bleu. **L'objet actuel** dont la ligne de code dans l'éditeur de texte est désignée par le curseur est représenté en rouge.

Pour les **sections de spline** (G05), le contour convexe du polynôme cubique est représenté en gris clair.

Remarque: Une position de commutation peut être insérée uniquement dans l'éditeur littéral CNC! (voir chapitre 3.4! Source de référence non disponible). Un point vert apparaît sur la courbe dans l'éditeur graphique.

3.6 Commandes et options

Outre les commandes du menu "Programme CNC", le menu "Extras" propose des commandes et possibilités de réglage pour le travail et l'affichage dans l'éditeur graphique. (Il est bien entendu également possible de programmer via la saisie des commandes ad hoc dans l'éditeur littéral.) La barre d'outil propose les boutons correspondants. Une option activée dans le menu Extras est pourvue d'un crochet et le bouton apparaît enfoncé.

Il est possible de sélectionner les cinq mode de traitement ci-dessous :



Selectionner



Mode d'insertion pour les lignes



Mode d'insertion pour les cercles a droite



Mode d'insertion pour les cercles a gauche



Mode d'insertion pour les splines

Pour modifier la représentation, on dispose des commandes et options ci-dessous :



Adapter la grandeur

Nouvelle numerotation du programme

Montrer grille

Remplacer les splines et ellipses par des droites



Correction du rayon d'outil



Arrondir coins



Poncer coins



Eviter les boucles



Supprimer le jeu



Montrer interpolation

Changer les valeurs epsilon pour zero

Remarque: Il est tout à fait possible de sélectionner plusieurs fois les options Correction du rayon d'outil, Arrondir coins, Poncer coins et Éviter les boucles. Ceci permet de simuler les effets de modules de préparation de trajectoire se succédant. Les fonctions de préparation de trajectoire "Correction du rayon d'outil", "Arrondir coins", "Poncer coins" ne peuvent pas être combinées entre elles.

Extras : Adapter grandeur

Bouton :

Si cette option est activée dans le menu „Extras“, la partie visible de l'écran est adaptée de telle sorte que l'ensemble du programme NC puisse être visualisé.

Extras : Montrer grille

Si cette option est activée dans le menu „Extras“, l'éditeur graphique est représenté avec une grille visible.

Extras : Nouvelle numérotation du programme

Bouton : 

Le menu Extras comprend l'option Nouvelle numérotation du programme à l'aide de laquelle les numéros de ligne (<numéro>N) sont automatiquement et continuellement adaptés aux dizaines.

Extras : Remplacer les splines et ellipses par des droites

Lors de l'interpolation et comparées à d'autres éléments, les splines et les ellipses nécessitent des calculs longs et laborieux. Afin d'éviter ceci, cette option de menu permet de remplacer toutes les splines et ellipses au sein d'un programme NC par un certain nombre de lignes droites. Ainsi, on peut utiliser des splines et des ellipses lors de la conception de la trajectoire sans que ces éléments ne soient calculés pour l'exécution.

Si cette commande est choisie, un dialogue propose deux options de conversion :

a) **selon la longueur** : une ligne droite est créée par unité de longueur x (le nombre x peut être saisi au sein du dialogue) de la spline / de l'ellipse, ou

b) **selon l'angle** : l'objet original est divisé de telle sorte que les nouvelles lignes droites présentent un angle inférieur à x (à saisir dans le dialogue en degrés).

Un conseil : dans les réglages par défaut, une décélération jusqu'à la vitesse 0 a lieu en cas d'interpolation de ligne droite – à l'inverse de la spline / de l'ellipse – après chaque nouvelle section de ligne droite. On peut éviter cela en augmentant de manière appropriée la tolérance d'angle.

Extras : Correction du rayon d'outil

Bouton : 

Si cette option est activée dans le menu „Extras“ et si le départ (G41/G42) et la fin (G40) de la correction du rayon d'outil sont programmés dans le Programme CNC ainsi qu'un rayon d'outil (D<radius>), la trajectoire modifiée en conséquence est représentée. Cette option de menu correspond au module SMC_ToolCorr de la bibliothèque SoftMotion. La trajectoire initiale est représentée comme référence en gris clair. La trajectoire initiale est représentée comme référence en gris clair.

Extras : Poncer coins

Bouton : 

Si cette option est activée dans le menu „Extras“, la trajectoire programmée est représentée et souligne l'effet que le module SMC_SmoothPath de la bibliothèque SoftMotion SM_CNC.lib peut avoir. Celle-ci procède au ponçage de coin via un polynôme cubique (spline). La condition est que Départ (G51), Fin (G50) et Rayon d'arrondi (D<radius>) soient définis dans le programme CNC. La trajectoire initiale est représentée comme référence en gris clair.

Extras : Arrondir coins

Bouton : 

Si cette option est activée dans le menu „Extras“, la trajectoire est représentée avec les coins arrondis, soulignant ainsi l'effet que le module SMC_RoundPath de la bibliothèque SoftMotion peut avoir sur la trajectoire programmée. La condition est que Départ (G52), Fin (G50) et Rayon d'arrondi (D<radius>) soient définis dans le programme CNC. La trajectoire initiale est représentée comme référence en gris clair.

Extras : Éviter les boucles

Bouton : 

Si cette option est activée dans le menu „Extras“, la trajectoire représentée est celle qui est générée lorsqu'à un point de recoupement de ladite trajectoire avec elle-même, on procède à un raccourci. Ceci permet d'éviter des boucles. Cette option de menu fonctionne comme le module SMC_AvoidLoop de la bibliothèque SoftMotion.

La condition est que Départ (G61) et Fin (G60) soient définis dans le programme CNC. La trajectoire initiale est représentée comme référence en gris clair.

Extras : Supprimer le jeu

Bouton : 

Si cette option est activée dans le menu „Extras“, toutes les lignes de l'éditeur de texte qui commencent par „/“ sont ignorées.

Extras : Montrer interpolation

Bouton : 

Si cette option est activée dans le menu „Extras“, des points d'interpolation sont représentés par pas de 100 ms ; la position de l'outil est ainsi marquée par une petite croix grise toutes les 100 ms. Ainsi, on peut consulter le comportement par rapport à la vitesse (rapide = plus grands écarts, lente = écarts rapprochés).

Extras : Changer les valeurs epsilon pour zéro

Le contrôle interne d'une valeur x quant au zéro doit être remplacé en raison de l'imprécision du calcul à virgule flottante due au contrôle $x < e$. La grandeur de la valeur e doit si nécessaire (p.ex. en cas d'utilisation de notation à virgule flottante 32 bit au lieu de 64 bit, ou en cas d'importation d'un programme CNC à précision limitée) est adaptée ; on dispose pour ce faire du dialogue **Valeurs de tolérance zéro** auquel on accède via la commande "Changer les valeurs epsilon pour zéro" du menu Extras ou via le menu contextuel.

Attention: La modification de cette valeur n'est pas nécessaire pour une utilisation standard et devrait de ce fait être évitée !

Extras : Sélectionner

Bouton : 

Si cette option est activée dans le menu „Extras“, un clic sur un élément graphique dans l'éditeur CNC permet de sélectionner ledit élément (couleur rouge) et de marquer la ligne correspondante dans l'éditeur de texte. Un clic sur le point final d'un élément permet de déplacer ce dernier au gré.

Extras : Mode d'insertion pour les lignes

Bouton : 

Si cette option est activée dans le menu „Extras“, un clic de souris dans la champ d'éditeur permet d'insérer un élément G01 sous la forme d'une ligne. Le nouvel élément est placé derrière l'élément sélectionné (rouge). La position de la souris détermine la ligne.


Extras : Mode d'insertion pour les cercles a droite

Bouton : 

Si cette option est activée dans le menu „Extras“, un clic de souris dans la champ d'éditeur permet d'insérer un élément G02 sous la forme d'une croix pour un déplacement dans le sens horaire. Le nouvel élément est placé derrière l'élément sélectionné (rouge). La position de la souris détermine la

cible. Le rayon du cercle est défini par défaut à 100 et doit le cas échéant être modifié dans l'éditeur de texte.

Extras : Mode d'insertion pour les cercles à gauche

Bouton : 

Si cette option est activée dans le menu „Extras“, un clic de souris dans la champ d'éditeur permet d'insérer un élément G03 sous la forme d'une croix pour un déplacement dans le sens anti horaire. Le nouvel élément est placé derrière l'élément sélectionné (rouge). La position de la souris détermine la cible. Le rayon du cercle est défini par défaut à 100 et doit le cas échéant être modifié dans l'éditeur de texte.

3.7 Remplissage automatique de structure

Lors de la compilation du programme IEC, un répertoire de variables globales "CNC Data" est automatiquement créé. Les programmes NC de structure de données identique y sont enregistrés.

Comme on l'a vu dans le chapitre "Éditeur CNC - Aperçu", il existe deux variantes qui peuvent être réglées séparément pour chaque programme NC au sein de l'éditeur NC, dans le menu Programme CNC :

- 1) créer variable de programme en compilant
- 2) créer OutQueue en compilant
- 3) ne pas compiler

Cette option correspond au cas de figure (c). Le programme est disponible comme fichier ASCII ou OutQueue dans le système de fichiers de la commande; ce programme doit être lu pour exécution par un des modules de la bibliothèque SM_FileFBs.lib (voir chapitre 10.2). C'est pourquoi il ne doit pas être repris dans les données IEC.

4 L'éditeur CAM dans CoDeSys

4.1 Aperçu

L'éditeur CAM SoftMotion (Éditeur CAM) est intégré au niveau de programmation de CoDeSys. Il est possible de programmer ici sous forme de graphiques et de tableaux des CAM électroniques et mécaniques à partir desquelles CoDeSys crée des structures de données globales (CAM Data) à la compilation du projet. Le programme IEC peut accéder à ces structures.

Pour la conversion des CAM dans le programme IEC, on utilise les modules standardisés de PLCopen (bibliothèque SM_PLCOpen.lib).

(Voir également les remarques générales relatives à l'utilisation de CAM ainsi que les exemples de programmation)

4.2 Définition d'une CAM pour SoftMotion

De manière simplifiée, une CAM décrit la dépendance fonctionnelle d'une grandeur (esclave) par rapport à une autre (maître).

Afin de décrire cette dépendance, l'**axe maître** est divisé en différents intervalles. Pour chaque intervalle $[a,b]$, CoDeSys propose deux possibilités pour former une représentation fonctionnelle de l'axe maître sur l'**axe esclave** :

- **Rectiligne** : la dépendance est décrite par le biais d'une représentation linéaire. Dans cet intervalle, la première dérivation (vitesse) est constante selon la pente des lignes droites, la seconde dérivation (accélération) est égale à 0.
- **Polynôme de degré 5** : cette section décrit la dépendance par le biais d'un polynôme de degré 5. Les première et seconde dérivations deviennent ainsi des polynômes de degré 4 et 3.

Sur ces intervalles, les fonctions doivent se succéder les unes aux autres de telle sorte qu'aux transitions, tant la valeur fonctionnelle qu'au moins les première et seconde dérivations doivent être constantes.

Il est possible de définir dans l'éditeur CAM des points d'appui individuels et des lignes droites. Des polynômes de degré 5 relient les intervalles situés entre ces points, pour autant que les conditions de constance et de différenciation soient remplies.

Le long des lignes droites, des valeurs fonctionnelles sont définies ainsi que la première dérivation (vitesse, constante dans ce cas) et la seconde dérivation (accélération, toujours 0 dans ce cas). Un point peut être défini avec première et seconde dérivation au gré.

En plus, l'utilisateur a la possibilité de placer des cames (interrupteurs de position binaires) sur la CAM.

4.3 Démarrage e inserer nouvelle CAM

Démarrage

L'éditeur **CAM** est lancé à partir de l'onglet "Ressources". On accède alors à une fenêtre divisée en trois parties. Si aucune CAM n'est créée, cette fenêtre reste vide. Autrement, la colonne gauche reprend une arborescence avec toutes les CAM programmées; la moitié supérieure de la fenêtre représente graphiquement la dépendance de fonction entre les axes maître et esclave.

Il est possible de choisir entre **trois types** pour la moitié inférieure de la fenêtre. Les commandes correspondantes se trouvent dans le menu "Extras" :

1. **Afficher vitesse/accélération** : représentation de la première dérivation (courbe bleue, vitesse) et de la seconde (courbe verte, accélération) ;
2. **Courbe comme tableau** : représentation sous forme de tableau de tous les éléments CAM (points / lignes droites) ;
3. **Cames comme tableau** : représentation sous forme de tableau avec toutes les cames.

Les tableaux peuvent être édités.

Les menus "Insérer" et "Extras" proposent alors les commandes pour Créer et Éditer des CAM.

Insérer: Nouvelle CAM

Sélectionner la commande "**Nouvelle CAM...**" dans le menu "Insérer" ou dans le menu contextuel et procéder aux modifications souhaitées dans le dialogue "**propriétés CAM**" :

Nom de CAM : désignation de la nouvelle CAM

Type : une **CAM** peut contenir tant des éléments CAM que des cames elles-mêmes; une came mécanique (**Tableau de CAM digital**) ne contient que des cames

Graduation de l'axe maître : la graduation de l'axe maître est définie ici. si l'option **360°** est activée, les réglages de **Valeur minimum**, **Valeur maximum**, **Étape** et **Unité** sont automatiquement effectués (0, 360, 20, °) ; il est sinon également possible d'effectuer ces réglages de manière individuelle.

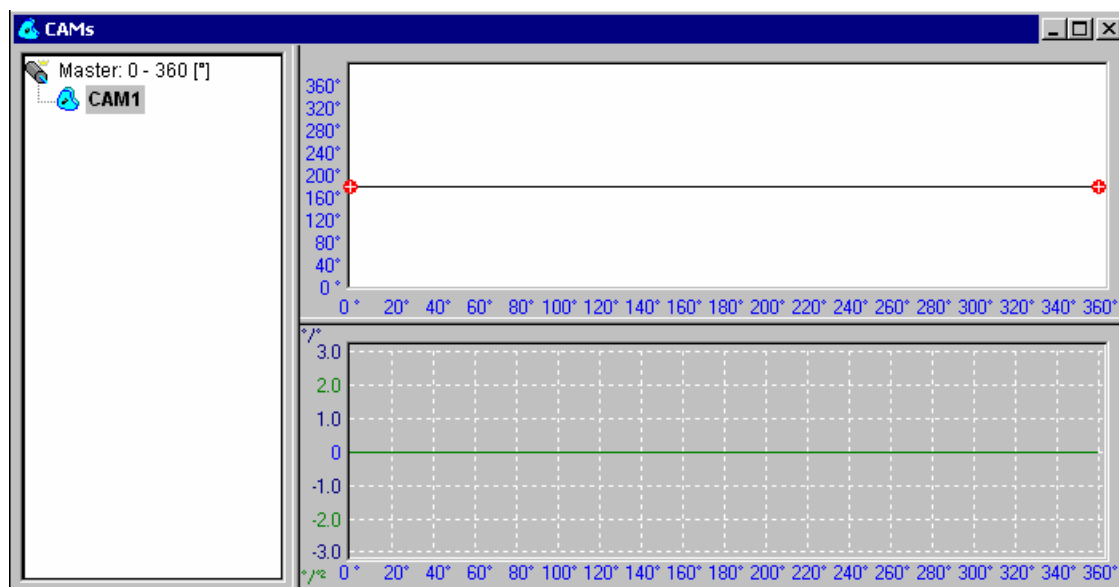
Graduation de l'axe esclave : la graduation de l'axe esclave est définie ici. Pour les réglages par défaut, voir la figure du dialogue à la fin de ce chapitre.

Propriétés : si l'option **périodique** est activée, les valeurs de fonction en début et en fin de CAM ainsi que les premières et secondes dérivations coïncident. Les modifications qui ont été effectuées sur le point final lors de l'édition de la courbe sont alors ignorées. Voir également ICI pour des remarques relatives à l'utilisation de CAM.

Fermer le dialogue en appuyant sur **OK** pour confirmer la saisie.

Ceci fait apparaître dans la colonne gauche (liste CAM) de la fenêtre d'éditeur CAM le nom de la nouvelle CAM. Tant que cette entrée est marquée, la CAM est représentée dans l'éditeur et peut donc être éditée. Les deux parties droites de la fenêtre permettent de représenter la nouvelle CAM. Vous

pouvez voir l'axe maître horizontal en bleu, l'axe de position bleu (esclave) dans la fenêtre supérieure ainsi que les graduations de vitesse (bleu foncé) et d'accélération (vert) dans la fenêtre inférieure. La représentation ci-dessous correspond aux réglages par défaut du dialogue des propriétés.



Afin de modifier les réglages décrits ci-dessus, on peut toujours accéder au dialogue des propriétés pour la CAM actuellement marquée par le biais de la commande "Configuration" disponible dans le menu "Extras" ou dans le menu contextuel. Il est également possible de double-cliquer sur l'entrée correspondante dans la liste CAM.

Arborescence CAM

La moitié gauche de la fenêtre contient une arborescence CAM qui reprend toutes les CAM et les cames mécaniques. Ces éléments sont toujours classés de sorte que tous les éléments d'une seule et même graduation maître (et donc se rapportent potentiellement au même axe) sont du même père.





4.4 Éditer CAM

Choisir dans la colonne gauche de l'éditeur la CAM que l'on souhaite modifier. Cliquer ensuite sur l'entrée afin de marquer celle-ci (elle apparaît sur un fond bleu) et le programme est affiché dans la fenêtre d'éditeur.

En appuyant sur la touche <Ctrl> et cliquant simultanément sur une ou plusieurs CAM du même père, ces CAM sont également affichées.

4.4.1 Configuration générale de l'éditeur

Le mode d'édition peut être choisi dans le menu "**Insérer**" ou via le bouton ad hoc de la barre d'outils :

-  sélectionner élément
-  insérer point
-  insérer ligne droite
-  insérer came

Édition du paramétrage CAM : Afin de modifier les réglages effectués au sein du dialogue "**propriétés CAM**" dès l'insertion, sélectionner la commande Configuration dans le menu "**Extras**".

Modifier l'affichage de la courbe : vous disposez pour ce faire des commandes "Compléter courbe" et "Afficher extrêmes" dans le menu "Extras".

Le mode d'affichage peut être choisi dans la fenêtre inférieure du menu "Extras" ou via le bouton ad hoc de la barre d'outils :



Afficher vitesse/accélération : la fenêtre inférieure permet de visualiser la première (bleue) et la seconde (verte) dérivée de CAM.



Courbe comme tableau: la fenêtre inférieure montre les éléments individuels (points, lignes droites) qui constituent la CAM ainsi que leurs propriétés sous la forme d'un tableau éditable.



Cames comme tableau: la fenêtre inférieure montre les cames et leurs propriétés sous la forme d'un tableau éditable.

4.4.2 Édition des propriétés des éléments de courbe

Les attributs de chaque élément peuvent être modifiés au sein du dialogue des propriétés, via sélection et déplacement dans la fenêtre d'éditeur.

1. Via le dialogue des propriétés :

Point, ligne droite : un double-clic sur l'élément donne accès au dialogue "propriétés élément CAM" dans lequel on peut modifier les propriétés suivantes des éléments via une saisie numérique :

Propriétés élément CAM	
Type d'élément:	Line
Démarrage maître:	312.308
Démarrage esclave:	314.286
Fin maître:	319.508
Fin esclave:	332.114
Vitesse:	2.476
Accélération:	0.000

Type d'élément: ligne (droite) ou point ; le type ne peut pas être modifié ; si on fait une ligne droite d'un point, une certaine longueur est automatiquement prédéfinie ; si on fait un point d'une ligne droite, les coordonnées du point de départ sont automatiquement reprises.

Départ maître, Fin maître: valeurs de départ et de fin (uniquement pour éléments rectilignes) par rapport à l'axe X (pour l'unité, voir dialogue "propriétés CAM" ("Extras" "Configuration"))

Départ esclave, Fin esclave: valeurs de départ et de fin (uniquement pour éléments rectilignes) par rapport à l'axe X (pour l'unité, voir "Extras" "Configuration")

Vitesse: (uniquement pour éléments de points)

Accélération: (uniquement pour éléments de points)

Came: un double-clic sur l'élément donne accès au dialogue "propriétés élément de came" qui permet de régler les valeurs suivantes :

Activer par: la came est activée : la variable booléenne liée à l'ID du groupe de cames (voir ci-dessous) dans le programme (Bit de came) devient TRUE lorsque la courbe est parcourue ; il est ainsi possible de régler l'une des trois options ci-dessous :

passage positive: uniquement si la courbe est parcourue de la gauche vers la droite ; cliquer sur "accepter" et la flèche verte au dessus de la came indique la droite

passage négatif: uniquement si la courbe est parcourue de la droite vers la gauche ; cliquer sur accepter et la flèche verte au dessus de la came indique la gauche

chaque passage: à chaque passage de la courbe ; cliquer sur accepter et la flèche verte au dessus de la came indique à la fois la gauche et la droite

Action: une des options ci-dessous peut être réglée afin de déterminer les effets de l'activation de la came sur l'action qui lui est liée au sein du projet :

mettre en circuit: l'action est entamée (le "bit de came" devient TRUE) ; le carré de came est vert

mettre hors circuit: l'action est arrêtée (le "bit de came" devient FALSE) ; le carré de came est rouge

inverser: si l'action est en cours, elle est alors arrêtée ; si elle est inactive, elle est alors entamée (le bit de came est inversé) ; le carré de came est jaune

contrôle par le temps: l'action est entamée avec les valeurs saisies dans les champs Temporisation et Durée ; le carré de came est bleu foncé

ID groupe: numéro d'identification de la came servant de référence au projet (INT) ; plusieurs cames peuvent obtenir la même ID de groupe et être "regroupées" de telle sorte qu'elles soient atteintes lors d'un passage par la même action de programme

Décélération [µs]: intervalle de temps après lequel l'action liée à la came au sein du projet est entamée suite au passage par l'élément de ladite came (après lequel le bit de came doit devenir TRUE). (uniquement si l'action est commandée dans le temps).

Durée [µs]: indication de la durée d'activation de l'action avec laquelle la came est liée au sein du projet, suite à son démarrage (durée pendant laquelle le bit de came doit rester TRUE)). (uniquement si l'action est commandée dans le temps).

Position du maître: position X de la came

Position de l'esclave: position Y de la came , non éditable car définie par le tracé de la courbe

Les valeurs saisies dans les dialogues de propriétés peuvent être confirmées via OK ou Accepter, suite à quoi la courbe est adaptée au sein de l'éditeur. OK ferme en même temps le dialogue, ce dernier reste ouvert avec Accepter.

(Édition des propriétés des éléments de courbe)

2. Via sélection et déplacement dans la fenêtre d'éditeur :

Un éléments peut être sélectionné d'un clic de souris puis déplacé en maintenant le bouton de la souris enfoncé. Ceci entraîne par réaction une modification des valeurs correspondantes dans le dialogue "Propriétés" (voir ci-dessus):

Point: un clic sur un point fait apparaître un petit carré rouge qui représente la pente (vitesse). Un décalage de ce carré rouge permet de modifier la pente du point. Cette dernière est affichée par le biais d'une tangente. De même, il est possible de déplacer le point lui-même.

Rectiligne: un clic sur une ligne droite fait apparaître des petits carrés rouges en ses extrémités. Un autre clic de souris sur une des extrémités et un déplacement du carré rouge permet de modifier la pente de la ligne ; par contre, un autre clic sur la ligne droite elle-même permet de la déplacer sans en modifier la pente.

Came: un clic sur une came fait apparaître le contour du carré de came en rouge. La came peut être déplacée en faisant bouger le pointeur sur la trajectoire.

4.4.3 Fonctions des menus ,Extras' et ,Insérer

Extras : Configuration

Cette commande donne accès à la boîte de dialogue „Propriétés CAM“ visible également à la création de CAM. Il est possible de modifier ici la graduation et les unités.

Extras : Compléter courbe

Si cette option est activée (crochet devant l'option de menu ou bouton „enfoncé“ dans la barre d'outils), les polynômes de degré 5 reliant les éléments (points, lignes droites, comes) sont affichés en plus dans la courbe. Dans les autres cas, les éléments individuels sont seuls affichés.

Extras : Afficher extrêmes

Si cette option est activée (crochet devant l'option dans le menu „Extras“ ou bouton „enfoncé“ au sein de la barre d'outils), on affiche en plus de la CAM et de ses dérivations ses valeurs extrêmes (maximum / minimum).

Extras : Options de compilation

Cette option de menu donne accès à un dialogue à l'aide duquel on peut régler la compilation des CAM.

Il existe trois modes principaux de compilation :

1. Polynomiale: lors de la compilation, des variables de structure MC_CAM_REF sont créées. Pour chaque section, celles-ci contiennent la description du polynôme de degré 5, celui-ci décrivant à son tour la CAM. Les structures de ce type sont utilisées comme entrée du module MC_CamIn. La structure fait partie intégrante de la bibliothèque SM_DriveBasic.lib.

2. Tableau de points d'appui équidistants: Conformément aux réglages dans la partie inférieure du dialogue, un tableau des points d'appui de type SMC_CAMTable_<Type de donnée>_<nombre d'éléments>_1 est créé. Le tableau de position de ce dialogue contient les valeurs esclaves CAM par rapport aux valeurs maîtres qui sont réparties uniformément sur la plage de définition de l'axe maître. La première valeur du tableau se rapporte à la position esclave au minimum du maître CAM. La dernière valeur se rapporte quant à elle à la position esclave au maximum du maître, cela pour des CAM non périodiques. Pour ces CAM périodiques, cette valeur ne doit pas à nouveau être écrite car elle correspond avec la valeur du minimum du maître ; c'est pourquoi les intervalles sont quelque peu réduits et la dernière valeur du tableau décrit la position esclave à maître.fin - (maître.fin-maître début)/nombre d'éléments.

3. Tableau de points d'appui optimisé en éléments: Conformément aux réglages dans la partie inférieure du dialogue, un tableau des points d'appui bidimensionnels (normalement non équidistants) de type SMC_CAMTable_<Type de donnée>_<nombre d'éléments>_2 est créé. Le tableau contient des paires de positions combinant maître et esclave s'y rapportant. La répartition est effectuée de telle sorte que les éléments à vitesse constante (lignes) ne comprennent qu'un point au début et un point à la fin. Les autres points d'appui sont si possible répartis uniformément sur le reste de la CAM.

4. ne pas compiler : Aucune variable globale n'est créée pour cette CAM. Cette option est principalement utilisée lorsque la CAM doit être lue en cours d'exécution à partir du système de

fichiers (voir chapitre 10.3), p.ex. parce qu'elle doit être changée sans que l'on ne touche au projet CoDeSys en cours.

La **Graduation maître** et la **Graduation esclave** ne sont importantes que pour le tableau des points d'appui. On peut définir ici une graduation tant pour le maître que pour l'esclave, soit via l'affectation du début et de la fin, soit une valeur initiale et une affectation d'unité.

Extras : Écrire CAM dans un fichier

Cette option (menu „Extras“) donne accès à un dialogue de sélection de fichier dans lequel on peut indiquer un fichier *.CAM permettant l'écriture de la CAM actuelle. Pour son exécution, ce fichier peut être lu par le bloc fonctionnel SMC_ReadCAM (voir 10.3) être modifié par rapport à sa structure standard de données. Selon les options de compilation paramétrées, la CAM est enregistrée sous forme polynomiale, équidistante ou d'éléments optimisés.

Extras : Lire CAM du fichier

Cette option (menu „Extras“) permet de lire à nouveau un fichier *.CAM dans CoDeSys. Dès que le fichier correspondant est sélectionné, on accède au dialogue „Propriétés“ dans lequel il faut définir un nom pour la CAM ainsi que la graduation esclave.

Comme à la création de la CAM, les seules informations éditables sont celles nécessaires à l'exécution de l'arc de cercle, on peut distinguer une CAM lue de la CAM originale.

Extras : Exporter CAM comme tableau ASCII

Cette fonction permet d'importer la came actuelle sous forme de fichier texte ASCII. Il est ainsi possible de spécifier le nombre de points. Le point de départ et celui de fin sont toujours contenus comme premier et dernier points. Un fichier texte est ensuite généré avec la structure suivante :

```
<Master-Position>;<Slave-Position><CR><LF>
```

Le fichier texte ainsi créé peut ensuite être importé dans d'autres programmes et être utilisé pour concevoir des entraînements.

Extras : Importer CAM du tableau ASCII

Cette fonction permet d'importer des tableaux CAM enregistrer sous forme de fichier ASCII au format ci-dessous :

```
<Master-Position>;<Slave-Position>
```

Il faut noter à cet égard que les points se rapportant à la position maître sont triés par ordre croissant.

Si la CAM a pu être lue, l'utilisateur peut en définir le nom, l'intervalle et la graduation dans le dialogue des propriétés. Il a ensuite la possibilité de réduire le nombre de points d'appui de la CAM.

Extras : Afficher vitesse/accélération

Voir description ad hoc sous Démarrage de l'éditeur.

Extras : Courbe comme tableau

Voir description ad hoc sous Démarrage de l'éditeur.

Extras : Cames comme tableau

Voir description ad hoc sous Démarrage de l'éditeur.

Insérer : sélectionner élément

Cette option de menu sert à activer ou désactiver le mode de sélection. Tant que ce mode est activé (crochet devant l'option de menu, le bouton apparaît enfoncé dans la barre d'outils), il est possible de sélectionner d'un simple clic du bouton gauche de la souris l'élément de la courbe (point, ligne droite, came) sur lequel se trouve le pointeur. Ensuite, les symboles de marquage correspondants (petits carrés rouges) apparaissent et l'élément peut être édité.

Insérer : Insérer point

Bouton : 

Si cette option est activée dans le menu "Insérer" ou dans la barre de menu, il est possible d'insérer comme suit un nouveau point dans la courbe. Déplacer le pointeur sur la position du nouveau point d'appui. En cliquant maintenant du bouton gauche de la souris, le point est inséré et obtient une tangente horizontale. Dès que le bouton gauche de la souris est relâché, le point est repris (signalé par un rond rouge réticulé) et on passe automatiquement au mode "Sélectionner élément".

Si le bouton gauche de la souris est maintenu enfoncé lors de l'insertion, la pente de la tangente (vitesse) peut être modifiée immédiatement en déplaçant la souris.

Insérer : Insérer ligne droite

Bouton : 

Si cette option est activée dans le menu "Insérer" ou dans la barre de menu, il est possible d'insérer comme suit une ligne droite dans la courbe. Il faut déplacer le pointeur sur la position de départ de la ligne droite et maintenir le bouton gauche de la souris enfoncé. Le point de départ est marqué par un petit carré rouge et on crée la ligne droite en déplaçant la souris sur le point final souhaité (également marqué par un carré rouge). Le point final de la ligne droite ne peut pas être placé à gauche du point de départ ni à droite du prochain point défini. Dès que le bouton gauche de la souris est relâché, le point final est repris et on passe automatiquement au mode "Sélectionner élément".

Insérer : Insérer came

Bouton : 

Si cette option est activée dans le menu "Insérer" ou dans la barre de menu, il est possible d'insérer comme suit une came dans la courbe. Il faut déplacer le pointeur sur la position souhaitée pour la came et maintenir le bouton gauche de la souris enfoncé. La came est incorporée à la courbe en fonction de la position X choisie via le pointeur. Tant que le bouton gauche de la souris est enfoncé, il est possible de déplacer la came en déplaçant la souris. Dès que le bouton gauche de la souris est relâché, la position est reprise et on passe automatiquement au mode "Sélectionner élément".

4.5 L'éditeur CAM en mode en ligne

L'éditeur CAM (came électronique) est disponible en mode en ligne. Ceci permet de modifier les cames en cours d'exécution.

L'éditeur en ligne peut être lancé avec la commande interne (intern CAM) lors de l'utilisation de CoDeSys HMI.

Les cames sont présentes comme structures de données globales sur le contrôleur. Bien évidemment, sans nouveau chargement, seulement le contenu, et non la taille des structures de données, peut être modifié. C'est pourquoi, lors du mode en ligne, les fonctionnalités de l'éditeur sont limitées de la façon suivante :

- Les points peuvent être déplacés librement
- Les droites peuvent être modifiées librement
- Les cames peuvent être déplacées librement
- Les paramètres de cames (Echelle, etc...) peuvent être modifiés
- Il n'est pas possible d'insérer un nouveau point/ligne ou came
- Il n'est pas possible d'afficher de nouvelle came

Lors du mode en ligne, les commandes suivantes sont disponibles:

- 'Extras' 'Lire la came dans le contrôleur'
- 'Extras' 'Écrire la came dans le contrôleur'

Extras : Ecrire fichier dans l'automate

À l'aide de cette commande et après modification de la came dans l'éditeur CAM, l'utilisateur peut à nouveau écrire dans le contrôleur.

Il n'est pas pris en considération que la came peut éventuellement être en cours d'exécution, ce qui peut amener des sauts. L'utilisateur doit s'assurer que la came est modifiée de façon sûre.

Note: Les modifications effectuées peuvent être annulées avec la fonction reset du contrôleur. Les modifications sont enregistrées de façon permanente manuellement (par ex. avec SMC_WriteCAM ou en copiant la came dans le mémoire remanente).

Extras : Charger fichier de l'automate

L'éditeur CAM affiche la came actuelle lors du mode en ligne. Celle-ci ne doit pas correspondre à celle effectivement présente sur le contrôleur si elle a été modifiée lors d'un mode en ligne précédent ou sur l'application.

Cette commande permet de lire et d'ajuster la came actuelle à partir du contrôleur selon le mécanisme de supervision. Ainsi, il est possible d'ajuster la came affichée dans l'éditeur avec la came réelle du contrôleur.

Note: Cette fonction ne doit être utilisée que si le mode de traduction est „polynomial“ car uniquement dans ce cas, l'information complète de la came est disponible dans le runtime. La récupération des points originaux à partir d'un tableau de points n'est pas implémentée.

4.6 Utilisation des CAM

Pour l'exécution des CAM, on dispose de modules ad hoc dans la bibliothèque SM_PLCCopen.lib.

Des explications sont données ci-dessous pour l'utilisation de CAM, en particulier pour la connexion de CAM et la signification précise des différents paramètres (périodicité, décalage etc.) :

1. Effets des paramètres de module
2. Commutation entre CAM

4.6.1 Effets des paramètres de module

Propriétés CAM

- Périodique, - Période

Ces deux réglages ne sont pertinents que pour la création de la courbe : ils n'agissent pas sur l'exécution de la courbe au sein de l'application.

„Périodique“ décide si la courbe doit être accrochée sans saut, c.-à-d. exécutée de manière périodique. Si cette option est sélectionnée, les points de départ et de fin ont la même position esclave ou la différence entre eux est un multiple de la période.

Même si l'option „périodique“ n'est pas sélectionnée, les CAM peuvent être exécutées les unes à la suite des autres ; l'utilisateur doit dans ce cas veiller à ce que les transitions soient constantes comme souhaité.

MC_CAMTableSelect**- „Périodique“**

Cette entrée décide si une CAM doit être à nouveau exécutée dès que la position maître a quitté la plage CAM.

Si cette entrée est FALSE, la sortie EndOfProfile est réglée sur le module CamIn à la fin de la CAM et l'esclave est maintenu à la dernière position programmée par le CAM. Il convient de noter que si le maître accède à nouveau à la plage admissible, l'esclave est toujours commandé conformément à la CAM ; ainsi, l'action de la CAM n'est pas terminée pas la sortie hors de la plage maître spécifiée.

Si cette entrée est TRUE, la CAM est accouplée, c.-à-d. la CAM est décalée dans le sens du maître jusqu'à sa fin alors applicable.

Ainsi, la période maître (largeur) d'une CAM n'est pas proportionnelle à la période de position de l'entraînement maître. Il convient dès lors de noter que l'attribution de position esclave = Cam (position maître) et donc la définition de la position esclave par le biais de la position maître via la CAM ne vaut que pour le premier cycle CAM ; cela ne s'applique plus par après, alors que la largeur de la CAM se distingue de la période du maître.

- „MasterAbsolute“

Si cette entrée est TRUE, la CAM démarre à la position actuelle du maître. Ce point peut également se situer en plein milieu d'une CAM. Si ce point se situe en dehors de la plage définie par la CAM, une erreur est générée.

Si cette entrée est FALSE, la CAM est déplacée jusqu'à la position actuelle, ce qui signifie que le point zéro de la CAM correspond à la position actuelle du maître. Les CAM dont la plage maître ne contient pas de 0 ne peuvent pas être utilisées avec ce mode sous peine de générer une erreur (le maître quitte la plage spécifiée).

- „SlaveAbsolute“

Cette entrée est en rapport avec l'entrée StartMode de MC_CamIn. Le tableau ci-dessous décrit ses effets sur le mode de démarrage :

CamIn.StartMode	absolue	relative	ramp_in	ramp_in_pos	ramp_in_neg
CamTableSelect.SlaveAbsolute					
TRUE	absolue	relative	ramp_in absolut	ramp_in_pos absolut	ramp_in_neg absolut
FALSE	relative	relative	ramp_in relativ	ramp_in_pos relativ	ramp_in_neg relativ

Pour une description plus précise des modes, voir CamIn.StartMode

MC_CamIn**- „MasterOffset“, - „MasterScaling“**

Ces deux réglages permettent de modifier la position maître selon les formules ci-dessous et d'utiliser cette nouvelle position pour l'évaluation de la CAM :

$$x = \text{Master-Position} * \text{MasterScaling} + \text{MasterOffset}$$

MasterScaling > 1 implique ainsi une exécution plus rapide de la CAM (elle est comprimée) tandis que < 1 l'allonge.

- „SlaveOffset“, - „SlaveScaling“

Ces deux réglages permettent de décaler et étendre une courbe dans le sens esclave : la courbe est tout d'abord étendue puis est décalée.

SlaveOffset > 1 implique un plus grand mouvement de l'esclave (la courbe est allongée) tandis que < 1 comprime la courbe.

- „StartMode“ : absolute / relative / ramp_in / ramp_in_pos / ramp_in_neg

Relative signifie que la CAM est décalée de la position esclave actuelle à son démarrage. Ceci n'a de sens que si la position de consigne esclave est au point de démarrage 0 conformément à la CAM, sans quoi cela entraîne un saut.

„Absolute“ signifie que la CAM est évaluée indépendamment de la position esclave actuelle. Si l'esclave se trouve sur une autre position que celle résultant de la CAM et de la position actuelle du maître, cela entraîne un saut.

L'option „ramp_in“ fait qu'un saut survenant (voir ci-dessus) au démarrage de la CAM peut être compensé par un mouvement de compensation (selon les limites définies pour VelocityDiff, Acceleration, Deceleration). Ainsi, „ramp_in_pos“ (pour autant qu'in ne s'agit pas d'un esclave rotatif) ne peut compenser que dans le sens positif ; „ramp_in_neg“ dans le sens négatif. Pour des esclaves linéaires, le sens de compensation est calculé automatiquement, „ramp_in_pos“ et „ramp_in_neg“ sont traités comme „ramp_in“.

4.6.2 Commutation entre CAM

On peut normalement toujours commuter entre deux CAM. Il faut cependant tenir compte des points ci-dessous : Avec un front montant, la position de consigne de l'esclave est définie strictement selon la position actuelle du maître (calculée avec décalage et graduation maître) et de la CAM utilisée, cette position n'est modifiée que par un décalage et une graduation esclave. Ainsi, si la période maître ou esclave de la CAM et la période effective de position des entraînements sont différentes, cela peut entraîner à chaque cycle de CAM ou rotation de l'entraînement d'autres décalages des positions maître et esclave, de sorte que l'équation ci-dessous :

$$\text{position esclave} = \text{CAM}(\text{position maître})$$

ne soit plus valable, mais bien :

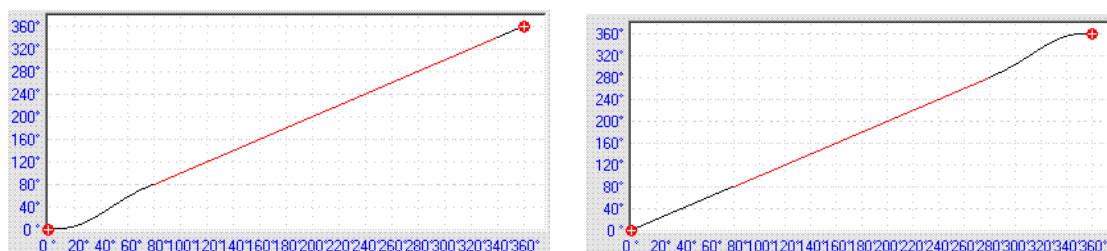
$$\text{position esclave} = \text{CAM}(\text{position maître} + \text{Offset1}) + \text{Offset2}$$

Offset1 et Offset2 peuvent être changés à la fin de toute CAM.

Il en résulte qu'au redémarrage du module CAM MC_CamIn (front montant à bExecute), ce dernier efface sa mémoire ainsi que Offset1 et Offset2, vu qu'une nouvelle action doit être lancée pour lui, revenant ainsi à l'équation originale qui peut entraîner une toute autre valeur de consigne esclave. C'est la raison pour laquelle on ne redémarre MC_CamIn-FB que lorsque l'on souhaite parcourir une autre CAM.

Noter les points suivants lors de la commutation entre deux courbes :

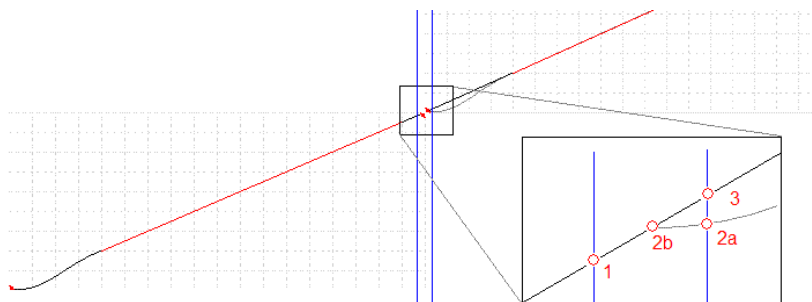
On passe p.ex. de la courbe 1 à la courbe 2 :



(1) Les vitesses et accélérations du point final de la dernière courbe et la nouvelle courbe doivent coïncider afin d'éviter des sauts et des à-coups.

C'est le cas dans l'exemple : le point final de la courbe 1 et le point de départ de la courbe 2 présentent les mêmes valeurs de vitesse et d'accélération.

(2a) Si la nouvelle courbe est définie de telle sorte que la position esclave débute à 0, on peut démarrer la courbe de manière relative, mais la courbe précédente doit démarrer de manière non périodique (MC_CamTableSelect). Raison : si on démarre cette courbe de manière périodique, les points d'appui ci-dessous pourraient être calculés :



Le passage de la courbe 1 à la courbe 2 est représenté ici. Le point 1 (première ligne bleue) est toujours sur la courbe 1 et est traité tout à fait normalement. À l'appel suivant du module, le maître (deuxième ligne bleue) se situe au delà de la fin de la courbe 1. Il est dès lors essentiel que la courbe précédente ait été exécutée de manière non périodique. Si tel était le cas, la position esclave 2b est calculée ; si elle avait été exécutée de manière périodique, on aurait eu la position 2a, position que l'on aurait obtenue si on avait souhaité répéter la courbe 1.

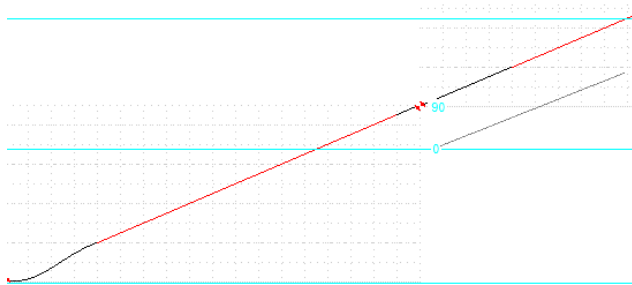
EndOfProfile est en même temps réglé, que l'exécution soit périodique ou non.

Cette sortie peut maintenant être utilisée pour démarrer la courbe 2. On appelle pour ce faire l'instance MC_CamIn via Execute=FALSE, instance que l'on appelle à nouveau dans le même cycle avec Execute=TRUE et la nouvelle CamTableID, ce qui entraîne la sortie de 3 sur l'entraînement esclave.

(2b) Si on veut démarrer la courbe de manière absolue, il faut noter que l'esclave doit se trouver à la position correcte au démarrage de la CAM, sous peine d'entraîner un saut. C'est normalement le cas lors de la commutation entre CAM, si la période de CAM correspond à celle de l'esclave.

Dans l'exemple ci-dessus, on pourrait démarrer la courbe 2 en mode absolu si la période de position du maître et de l'esclave est de 360° conformément à la plage CAM. Cela ne fait alors aucune différence si la CAM est démarrée de manière périodique ou non.

Ce ne serait p.ex. pas possible si la période esclave est de 270° (marquée par des traits en bleu clair) :



L'esclave serait alors sur la position 90 lors du passage de la courbe 1 à la courbe 2. Si on démarre maintenant la courbe 2 en mode absolu, on obtiendrait alors la courbe grise ainsi qu'un saut de position. On pourrait pallier à cela en programmant p.ex. un décalage esclave de 90.

4.7 Structure de données CAM

Lors de la compilation du projet, une liste de variables globales (registre „ressources“ dans CoDeSys) dénommée **CAM Data** est générée à partir des cames programmées dans l'éditeur CAM. Cette liste contient la description de chaque CAM selon une structure correspondant aux réglages du dialogue „Options de compilation“, cette description étant ainsi à la disposition du programme IEC ou des modules CAM. Pour une compilation correcte, les définitions structurelles appropriées doivent être disponibles dans le programme IEC.

(Voir à ce sujet également les exemples de programmation.)

En outre, une structure de données `_SMC_CAM_LIST` (ARRAY OF POINTER TO `MC_CAM_REF`) est créée lors de la compilation, désignant chaque CAM par le biais de pointeurs.

Naturellement, les structures de données appropriées peuvent également être créées ou complétées à partir du programme IEC d'exécution. Pour cette raison, elles seront plus amplement détaillées dans la suite. Les variables structurelles non décrites n'ont qu'une signification interne.

Les structures :

- `MC_CAM_REF`
- `SMC_CAMXYVA`
- `SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_1`
- `SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_2`

MC_CAM_REF

Cette structure de données représente une CAM générale et comprend les éléments suivants :

wCAMStructID: WORD

À l'aide de cette variable qui présente toujours une valeur fixe, les modules contrôlent si la structure de données saisie est bien de type `MC_CAM_REF`.

xStart, xEnd: LREAL

Plage de définition de la CAM : position de départ et de fin du maître.

byType: BYTE

Cette variable décrit le type de CAM et donc la manière de représentation de celle-ci.

1: équidistante, tableau unidimensionnel de positions esclave

2: non équidistante, tableau bidimensionnel de paires de points maître / esclave

3: description polynomiale de chaque point, se composant de position maître, position esclave, vitesse et accélération (XYVA).

byVarType: BYTE (uniquement pour `byType=1` ou `byType=2`)

Type de variable composant le tableau des courbes :

- 0: INT
- 1: UINT
- 2: DINT
- 3: UDINT
- 4: REAL
- 5: LREAL

nelements: INT

Nombre d'éléments selon le type de position esclave, position maître / esclave ou points XYVA.

byInterpolationQuality: BYTE (uniquement pour `byType=1` ou `byType=2`)

Degré d'interpolation fine : 1: linéaire (valeur par défaut), 3 : cubique

pce: POINTER TO BYTE

Pointeur sur élément effectif de donnée ; selon le type :

Type	
(équidistant)	<code>SMC_CAMTable_<VarType>_<néléments>_1</code>
2 (non équidistant)	<code>SMC_CAMTable_<VarType>_<néléments>_1</code>
3 (XYVA)	<code>ARRAY OF SMC_CAMXYVA</code>

nTappets: INT

Nombre d'actions de commutation de came.

pt: POINTER TO SMC_CAMTappet

Pointeur sur un ARRAY OF SMC_CAMTappet

strCAMName:STRING

Nom de la CAM

SMC_CAMXYVA

Une CAM XYVA est composée d'un tableau de SMC_CAMXYVA. Chaque variable de ce tableau décrit un point CAM via dX (position maître), dY (position esclave), dV (première dérivation dY/dX ; correspond à la vitesse esclave à vitesse maître constante 1) et dA (deuxième dérivation d²Y/dX² ; correspond à l'accélération esclave à vitesse maître constante 1). Le point de départ et celui de fin de la CAM doivent toujours être compris.

SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_1

Un tableau de courbes équidistantes est enregistré dans cette structure de données. Les positions esclave individuelles sont enregistrées dans le tableau: ARRAY[0..<Nombre d'éléments>-1] OF <Type de variable>. Le point de départ et celui de fin de la CAM doivent toujours être compris.

Les variables fEditorMasterMin, fEditorMasterMax, fTableMasterMin, fTableMasterMax décrivent une graduation supplémentaire du tableau qui reprend la plage de définition / de valeur en unités SoftMotion (fEditorMaster, fEditorSlave) graduée en unités de tableau (fTableMaster, fTableSlave).

SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_2

Un tableau de courbes non équidistantes est enregistré dans le tableau: ARRAY[0..<Nombre d'éléments>-1] OF <Type de variable>. À l'inverse des courbes équidistantes, le premier élément est la position maître et le second est la position esclave.

4.7.1 Exemple de CAM générée manuellement

Cet exemple montre comment on peut créer une CAM dans le programme IEC sans faire appel à l'éditeur :

Déclaration variables :

```
CAM : MC_CAM_REF := (
    byType:=2, (* non-equidistant *)
    byVarType:=2, (* UINT *)
    néléments:=128,
    xStart:=0,
    xEnd:=360);
Table : SMC_CAMTable_UINT_128_1 := (
    fEditorMasterMin := 0, fEditorMasterMax := 360,
    fTableMasterMin := 0, fTableMasterMax := 65536,
    fEditorSlaveMin := 0, fEditorSlaveMax := 360,
    fTableSlaveMin := 0, fTableSlaveMax := 65536);
```

Partie programme :

```
(* Créer CAM (l'exemple montre une CAM rectiligne); unique *)
FOR i:=0 TO 127 DO
    Table.Table[i][0]:=Table.Table[i][1]:=REAL_TO_UINT(i / 127.0 * 65536);
END_FOR
```

```
(* Relier pointeur; doit se faire à chaque cycle !!! *) *)  
CAM.pce := ADR(Table);
```

La CAM ainsi créée peut alors être saisie dans le module MC_CamTableSelect, sa sortie peut à nouveau être utilisée pour MC_CamIn.

5 Bibliothèque SM_PLCopen.lib

5.1 Aperçu

Les modules de la bibliothèque SM_PLCopen.lib se basent sur une spécification de PLCopen : ""Function blocks for motion control, Version 1.0".

La présente description se base sur cette spécification. Les modules qui répondent à la configuration standard de PLCopen commencent par MC_<nom de module>, tandis que les modules spécifiques 3S sont nommés SMC_<nom de module>.

Les blocs fonctionnels entièrement programmés selon IEC1131-3 se divisent en trois catégories:

- Modules pour utilisation, surveillance et paramétrage généraux d'entraînements individuels, voir chap.5.3.
- Modules pour commande indépendante d'entraînements individuels (single-axis). Ces modules permettent de déplacer des axes individuels de différentes façons, voir chap.5.3.
- Modules pour commande d'un entraînement (esclave) en fonction d'un autre entraînement (maître). Ces modules donnent la possibilité de réaliser des CAM, des réducteurs électroniques et des translations de phase, voir chap.5.4.
- Pour les modules supplémentaires (SMC_CAMRegister, SMC_GetTappetValue, MC_ReadSetPosition), voir chap. 5.5.
- De plus, des modèles de visualisation sont compris dans la bibliothèque pour tous les modules importants ; reliés à une instance du module correspondant, ces modèles montrent les entrées et sorties de ces modules importants et surtout d'autres objets essentiels pour les phases de création et de test de l'application.

Conditions:

Cette bibliothèque se base sur la bibliothèque "SM_DriveBasic.lib" (voir chap. 2.2) dans laquelle la structure AXIS_REF est entre autres définie.

5.2 Spécification PLCopen "Function blocks for motion control, Version 1.0"

Il est recommandé de lire en plus de la présente description la spécification PLCopen „Function blocks for motion control, Version 1.0“. Les points les plus importants sont brièvement repris ci-dessous:

Les modules peuvent être activés de deux manières:

a) **Entrée Enable** : Si le module dispose d'une entrée Enable (comme p.ex. MC_ReadParameter), celle-ci reste activée tant qu'Enable est TRUE.

b) **Entrée Execute** : Le module est activé par un front montant (passage de FALSE à TRUE) de l'entrée Execute et redevient inactivé dès qu'il a exécuté le déplacement, que le contrôle sur les axes (AXIS_REF) lui est retiré par un autre module, ou qu'il obtient un nouveau front montant au niveau de l'entrée Execute, recommençant ainsi le déplacement. Il faut également noter que toutes les variables d'entrée ne sont reprises qu'avec un front montant.

Les modules signalent par le biais de la **sortie Done** (ou une sortie similaire) soit la validité des sorties (p.ex. MC_ReadStatus) soit la fin du déplacement (p.ex. MC_MoveAbsolute).

Un module générant un déplacement qui est ensuite interrompu par un autre module active sa sortie CommandAborted pour signaler cet état de fait.

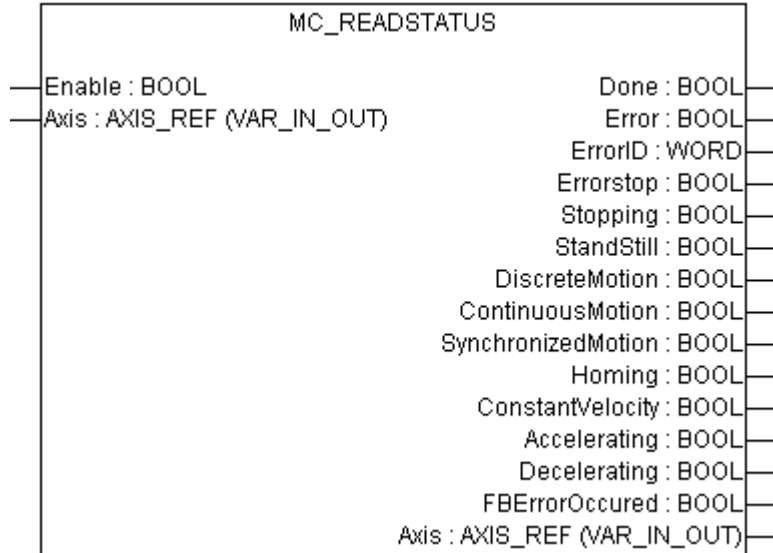
Suite à l'activation de leur sortie Done, les entrées des modules Execute entamés restent telles quelles tant que l'entrée Execute est activée. Un front descendant les supprime. Si un front descendant est détecté avant la fin, les sorties sont activées le temps d'un cycle pour être ensuite supprimées.

Tous les modules générant un déplacement supposent qu'au niveau de l'axe concerné, la validation de régulateur communique le desserrage du frein. Il signalent sinon une erreur.

5.3 Commande d'axes individuels

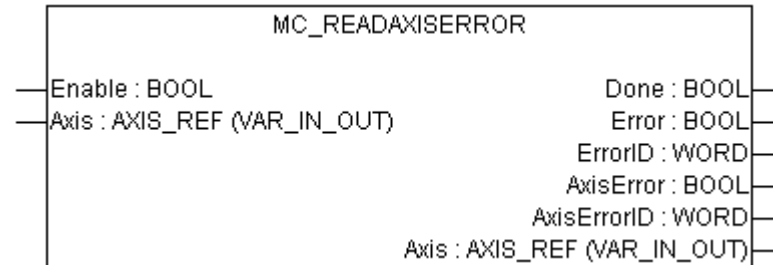
MC_ReadStatus

Ce module de la bibliothèque SM_PLCOpen.lib fournit les statuts sélectionnés d'un axe.



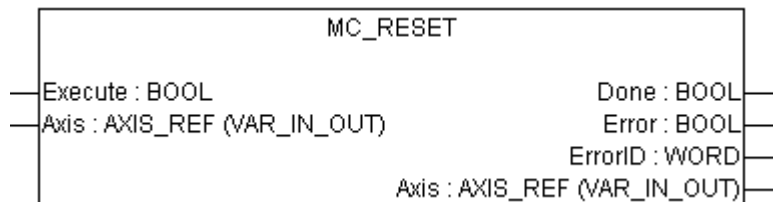
MC_ReadAxisError

Ce module de la bibliothèque SM_PLCOpen.lib affiche des erreurs générales qui sont survenues sur l'entraînement.



MC_Reset

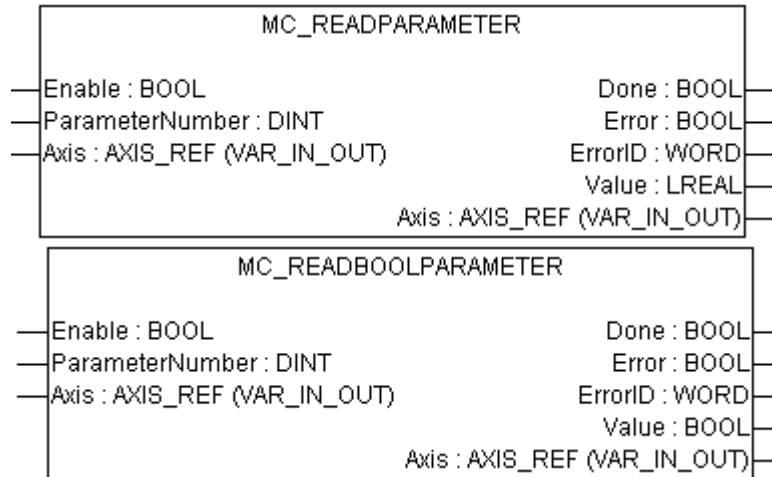
Ce module de la bibliothèque SM_PLCOpen.lib ramène le statut d'axe (SMC_AXIS_STATE) de error_stop à standstill.



MC_ReadParameter, MC_ReadBoolParameter

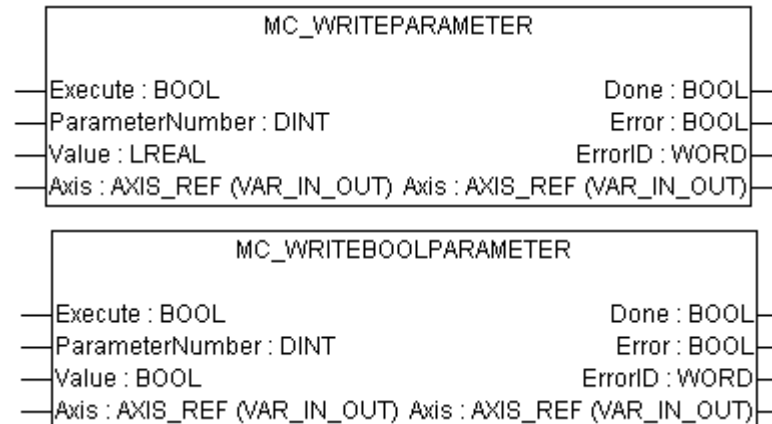
Avec ces modules de la bibliothèque SM_PLCopen.lib, il est possible de lire certains paramètres standard de la structure d'entraînement. Ses numéros sont en partie spécifiés par PLCopen, et en partie écrits par l'interface d'entraînement 3S SoftMotion.

Afin de lire des données de représentation d'entrée à partir de l'entraînement, on peut également utiliser ces modules. Un document relatif à chaque bibliothèque d'entraînement (XXXDrive.lib) décrit la codification des numéros de paramètres spécifiques aux entraînements.

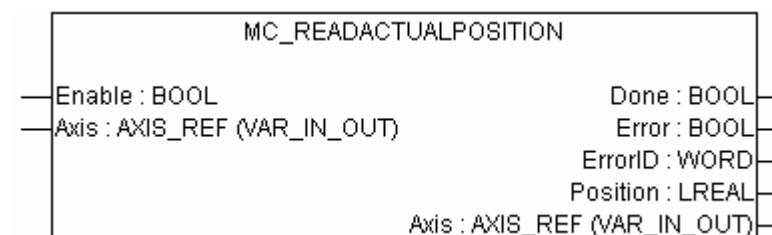
**MC_WriteParameter, MC_WriteBoolParameter**

Avec ces modules de la bibliothèque SM_PLCopen.lib, il est possible de définir certains paramètres standard de la structure d'entraînement. Ses numéros sont en partie spécifiés par PLCopen, et en partie par 3S – Smart Software Solutions GmbH dans l'interface d'entraînement SoftMotion.

Afin d'envoyer des données de représentation d'entrée à l'entraînement, on peut également utiliser ces modules. Un document relatif à chaque bibliothèque d'entraînement (XXXDrive.lib) décrit la codification des numéros de paramètres spécifiques aux entraînements.

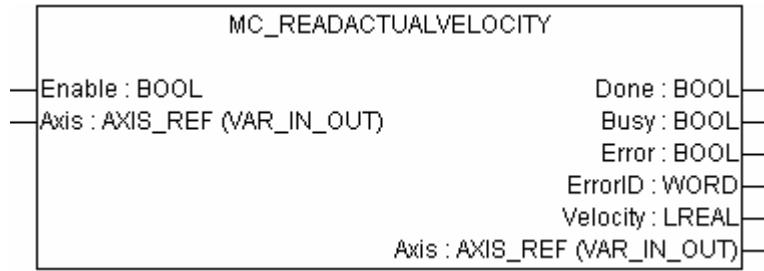
**MC_ReadActualPosition**

Ce module de la bibliothèque SM_PLCopen.lib indique la position momentanée de l'entraînement.



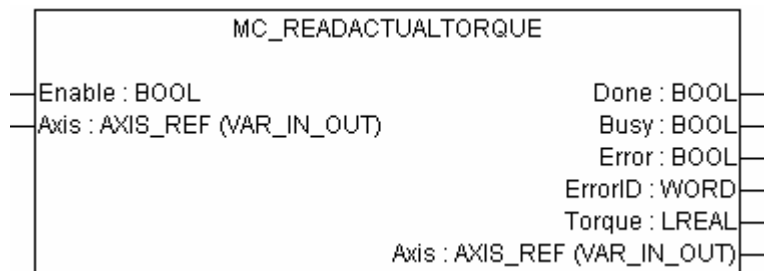
MC_ReadActualVelocity

Ce module de la bibliothèque SM_PLCOpen.lib indique la vitesse momentanée de l'entraînement.



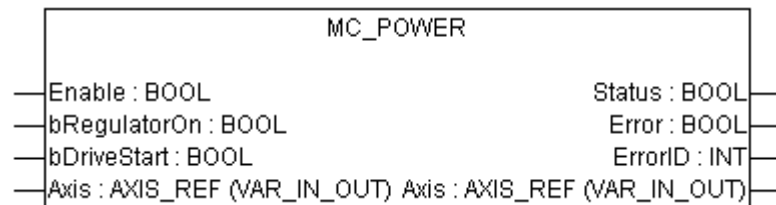
MC_ReadActualTorque

Ce module de la bibliothèque SM_PLCOpen.lib indique le couple actuel et la puissance momentanée de l'entraînement.



MC_Power

Ce module de la bibliothèque SM_PLCOpen.lib contrôle l'augmentation de puissance et le statut de décélération de l'entraînement. Si aucun entraînement n'est démarré sur cette trajectoire, si aucune validation de régulateur n'est réglée ou si le frein n'est pas desserré, ce module ne peut exécuter aucune commande de mouvement.



Entrées du module :

Enable : BOOL (valeur par défaut : FALSE)

Tant que cette variable est TRUE, l'entraînement est activé.

bRegulatorOn : BOOL (valeur par défaut : FALSE)

Met en marche / coupe l'entraînement.

bDriveStart : BOOL (valeur par défaut : FALSE)

Serre ou desserre le frein de l'entraînement.

Entrée / sortie (VAR_IN_OUT) du module :

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Sorties du module :

Status : BOOL (valeur par défaut : FALSE)

Indique si l'entraînement tourne actuellement avec (TRUE) ou sans régulation (FALSE).

Error : BOOL (valeur par défaut : FALSE)

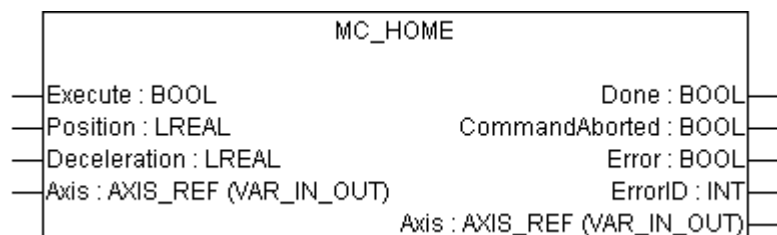
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT;

Numéro d'erreur

MC_Home

Ce module de la bibliothèque SM_PLCopen.lib provoque une course de référence de l'entraînement. Cette fonction est spécifique au fabricant des entraînements. Elle est activée uniquement par la DriveInterface. Dès que l'entraînement signale que la course de référence est correctement terminée, la sortie „Done“ est activée.



Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

La course de référence de l'entraînement doit être lancée avec un front montant.

Position : REAL

Mention de la position absolue au moment du signal de référencement.

Sorties du module :

Done : BOOL (valeur par défaut : FALSE)

Si TRUE, la course de référence est interrompue puis l'entraînement est à nouveau arrêté.

CommandAborted : BOOL (valeur par défaut : FALSE)

Si TRUE, la commande est interrompue par une autre commande.

Error : BOOL (valeur par défaut : FALSE)

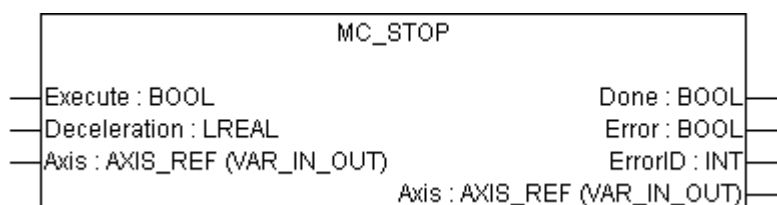
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_Stop

Ce module de la bibliothèque SM_PLCopen.lib freine l'axe jusqu'à la vitesse 0. Ce module ne peut pas être interrompu et bloque l'axe tant que l'entrée Execute est activée et que l'axe n'est pas complètement freiné.



Entrée / sortie (VAR IN OUT) du module:

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module:

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant, un processus de décélération est donc entamé.

Deceleration : REAL

Décélération, [u/s²).]

Sorties du module:

Done : BOOL (valeur par défaut : FALSE)

TRUE indique que l'entraînement est à l'arrêt

Error : BOOL (valeur par défaut : FALSE)

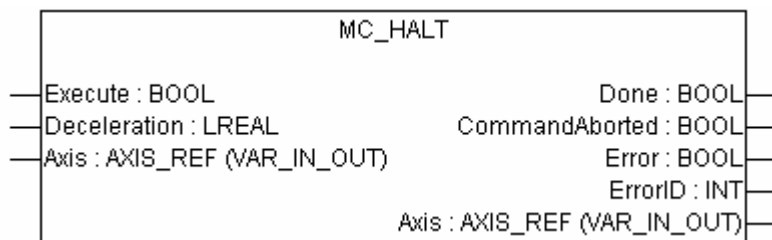
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_Halt

Ce module de la SM_PLCOpen.lib freine l'axe à vitesse 0. Ce bloc peut être avorté et bloque l'axe, contrairement au module MC_Stop.



Entrée / sortie (VAR IN OUT) du module :

Axe : AXIS_REF

La structure avec les données d'axe comprise dans Drive Interface (SM_DriveBasic.lib) est transmise.

Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant., c'est-à-dire le début de la décélération.

Deceleration : REAL

Décélération [u/s²]

Sorties du module :

Done : BOOL (valeur par défaut : FALSE)

TRUE montre que le moteur est immobilisé

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

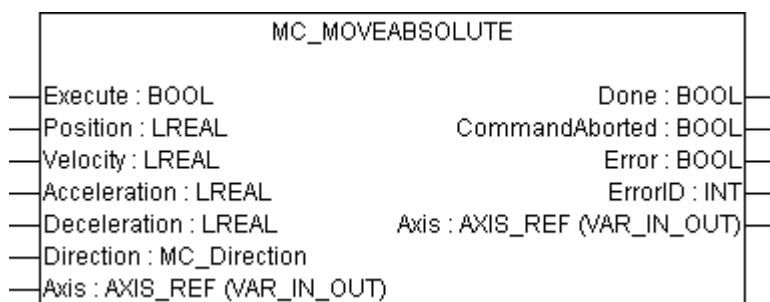
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro 'erreur

MC_MoveAbsolute

Ce module de la bibliothèque SM_PLCOpen.lib commande l'axe en une position absolue conformément aux valeurs saisies pour la vitesse, l'accélération et la décélération. Dans le cas d'un axe linéaire, la mention de la direction est sans importance ; pour des axes rotatifs, elle définit le sens de rotation.



Entrée / sortie (VAR IN OUT) du module :

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant.

Position : REAL

Position cible pour le mouvement (unité techn. [u])

Velocity : REAL

Valeur de consigne pour la vitesse qui ne doit pas nécessairement être atteinte, [μ/s]

Accélération : REAL

Accélération souhaitée, [u/s²].]

Deceleration : REAL

Décélération souhaitée, [u/s²].]

nDirection : MC_Direction (valeur par défaut : **shortest**)

Cette valeur d'énumération indique le sens souhaité ; pertinent uniquement pour des axes rotatifs ; voir SM_DriveBasic.lib. Valeurs admissibles : current (sens actuel de déplacement), positive, negative, shortest (distance la plus courte vue de la position actuelle), fastest (sens qui permet l'exécution la plus rapide).

Sorties du module :

Done : BOOL (valeur par défaut : FALSE)

TRUE indique que la position souhaitée a été atteinte

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ; exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

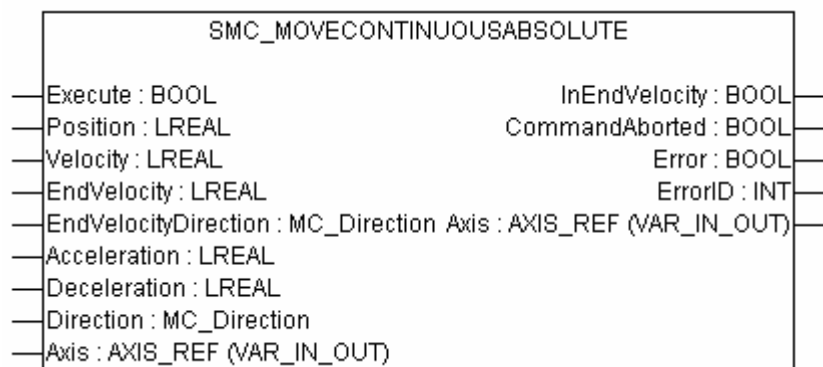
ErrorID : INT

Numéro d'erreur

SMC_MoveContinuousAbsolute

Ce module de la bibliothèque SM_PLCOpen.lib déplace l'axe vers une position fixe selon les valeurs entrées de vitesse, accélération et décélération. À la différence de MC_MoveAbsolute, l'utilisateur peut fixer une vitesse d'arrivée, affichée par le module via la sortie InEndVelocity, pour l'axe à cette position précise. Si le module n'est pas interrompu par un autre, il déplace les axes sans fin (statut de l'axe = move_continuous) à la vitesse d'arrivée définie.

En fonction des directives (chemin nécessaire pour amener les axes à la vitesse de fin < distance entre la position de départ et la position finale), il se peut que les axes effectuent au préalable un mouvement dans le sens inverse.



Entrée / sortie (VAR IN OUT) du module :

Axis : AXIS_REF

La structure qui au sein de l'interface 'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module :

Execute : BOOL (Default: FALSE)

Le module est activé avec un front montant.

Position : REAL

Position cible pour le mouvement (unité techn. [u])

Velocity : REAL

Valeur de consigne pour la vitesse qui ne doit pas nécessairement être atteinte, [μ /s]

EndVelocity : REAL

Valeur en chiffres de la vitesse de fin censée que l'axe doit avoir après avoir effectué la distance [u/s].

EndVelocityDirection : MC_Direction

Direction de la vitesse de fin censée (positive, négative ou courante (direction de rotation de l'axe au départ du mouvement)).

Acceleration : REAL

Accélération souhaitée, [u/s^2].]

Décélération souhaitée, [u/s^2].]

nDirection : MC_Direction (valeur par défaut : shortest)

Cette valeur 'énumération indique le sens souhaité ; pertinent uniquement pour des axes rotatifs ; voir SM_DriveBasic.lib. Valeurs admissibles : current (sens actuel de déplacement), positive, negative, shortest (distance la plus courte vue de la position actuelle), fastest (sens qui permet l'exécution la plus rapide).

Sorties du module :**InEndVelocity : BOOL (Default: FALSE)**

TRUE montre que la distance souhaitée a été parcourue et que la vitesse indiquée a été atteinte.

CommandAborted : BOOL (Default: FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (Default: FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

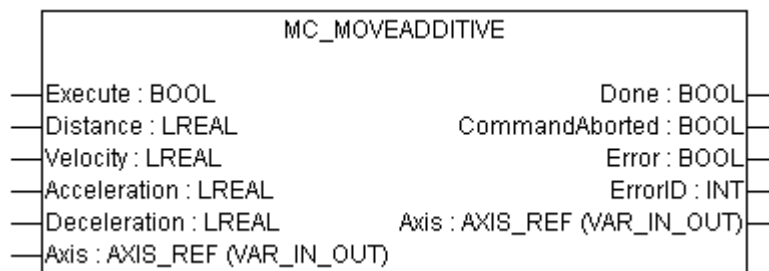
ErrorID : SMC_Error (INT)

Numéro 'erreur.

MC_MoveAdditive

Ce module de la bibliothèque SM_PLCopen.lib réagit de deux façons différentes en fonction du statut dans lequel l'axe se trouve :

1. **discrete_motion:**
La distance est ajoutée à la position cible du module travaillant à l'instant sur l'axe, et un déplacement a lieu jusqu'à la nouvelle position cible, conformément aux données.
2. **continuous_motion ou standstill:**
La distance est parcourue en arrière selon les données, vu à partir de la position momentanée.

Entrée / sortie (VAR IN OUT) du module:**Axis : AXIS_REF**

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module:**Execute : BOOL** (valeur par défaut : FALSE)

Le module est activé avec un front montant.

Distance : REAL

Distance relative pour le mouvement (unité techn. [u])

Velocity : REAL

Valeur de consigne pour la vitesse qui ne doit pas nécessairement être atteinte, [μ/s]

Accélération : REAL

Accélération souhaitée, [u/s²].]

Deceleration : REAL

Décélération souhaitée, [u/s²]

Sorties du module:

Done : BOOL (valeur par défaut : FALSE)

TRUE indique que la distance souhaitée a été atteinte

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

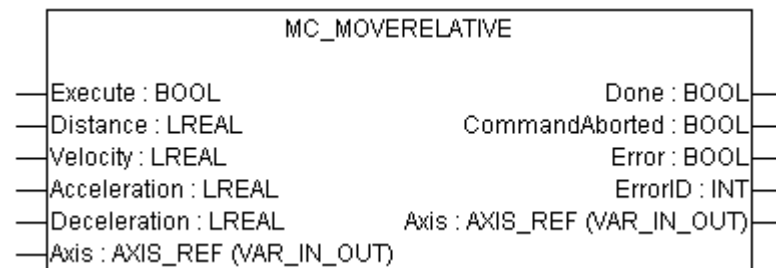
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_MoveRelative

Ce module de la bibliothèque SM_PLCOpen.lib commande l'axe en une position absolue conformément aux valeurs saisies pour la vitesse, l'accélération et la décélération. La distance peut ainsi être positive ou négative.



Entrée / sortie (VAR IN OUT) du module :

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant.

Distance : REAL

Distance relative pour le mouvement (unité techn. [u])

Velocity : REAL

Valeur de consigne pour la vitesse qui ne doit pas nécessairement être atteinte, [μ/s]

Accélération : REAL

Accélération souhaitée, [u/s^2].]

Deceleration : REAL

Décélération souhaitée, [u/s^2].]

Sorties du module :

Done : BOOL (valeur par défaut : FALSE)

TRUE indique que la distance souhaitée a été atteinte

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

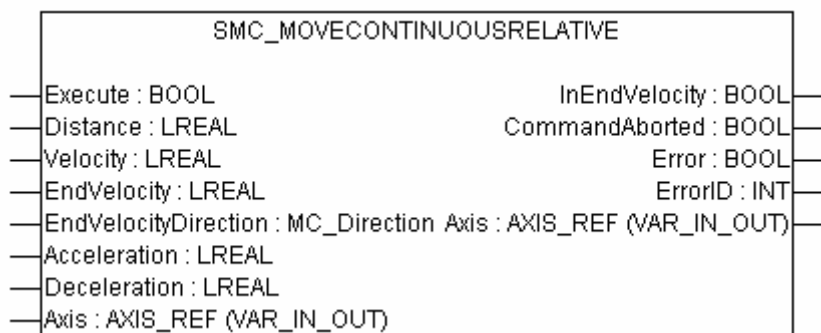
ErrorID : INT

Numéro d'erreur

SMC_MoveContinuousRelative

Ce module de la bibliothèque SM_PLCopen.lib déplace l'axe d'une distance relative aux valeurs entrées pour la vitesse, l'accélération et la décélération. La distance peut être positive ou négative. À la différence de MC_MoveRelative, l'utilisateur peut définir une vitesse d'arrivée de l'axe après que la distance fût parcourue. Cette valeur est affichée par le module par la sortie InEndVelocity. Si le module n'est pas interrompu par un autre, il déplace l'axe sans fin (statut des axes = move_continuous) à la vitesse d'arrivée définie.

En fonction des directives (chemin nécessaire pour amener l'axe à la vitesse de fin < distance entre la position de départ et la position finale), il se peut que l'axe effectue au préalable un mouvement dans le sens inverse.

**Axis : AXIS_REF**

La structure qui au sein de l'interface 'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module :**Execute : BOOL (valeur par défaut : FALSE)**

Le module est activé avec un front montant.

Distance : REAL

Distance relative mesurée entre la position de l'axe au moment du flanc montant lors de l'exécution Execute et au moment où la vitesse d'arrivée est atteinte.

Velocity : REAL

Valeur de consigne pour la vitesse qui ne doit pas nécessairement être atteinte, [μ /s]

EndVelocity : REAL

Valeur en chiffres de la vitesse de fin censée que l'axe doit avoir après avoir effectué la distance [u/s].

EndVelocityDirection : MC_Direction

Direction de la vitesse de fin censée (positive, négative ou courante (direction de rotation de l'axe au départ du mouvement)).

Acceleration : REAL

Accélération souhaitée, [u/s^2].]

Deceleration : REAL

Décélération souhaitée, [u/s^2].]

Sorties du module :

InEndVelocity : BOOL (Default: FALSE)

TRUE montre que la distance souhaitée a été parcourue et que la vitesse indiquée a été atteinte.

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

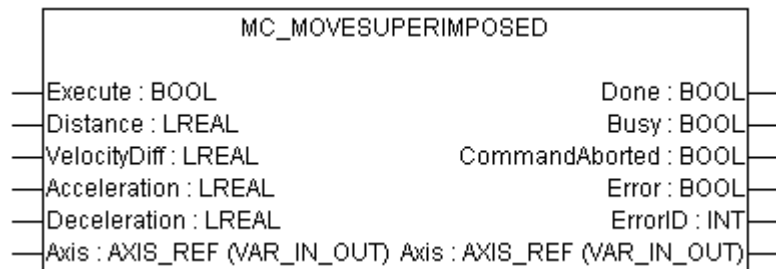
ErrorID : INT

Numéro 'erreur

MC_MoveSuperImposed

Ce module de la bibliothèque SM_PLCOpen.lib exécute, le cas échéant en plus de celui déjà en cours, un mouvement qui provoque en retour en arrière de l'axe selon une distance définie. Les valeurs indiquées pour la vitesse, l'accélération et la décélération sont à considérer comme étant relatives et donc indépendantes du mouvement en dessous. Le module initialement activé n'est pas interrompu par MC_MoveSuperImposed. Si le module initialement activé est interrompu par un autre module alors que MC_MoveSuperImposed est activé, le mouvement est interrompu par MC_MoveSuperImposed.

Il convient de noter que l'appel de MC_MoveSuperImposed suite à celui du module qui génère le mouvement en dessous doit être effectué !



Entrée / sortie (VAR IN OUT) du module :

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant.

Distance : REAL

Distance relative pour le mouvement (unité techn. [u])

VelocityDiff : REAL

Différence maximale de vitesse pour le mouvement momentané, qui ne doit pas nécessairement être atteinte, [μ /s]

Accélération : REAL

Accélération souhaitée, [u/s^2]

Deceleration : REAL

Décélération souhaitée, [u/s^2]

Sorties du module :**Done : BOOL (valeur par défaut : FALSE)**

TRUE indique que la distance souhaitée a été parcourue

Busy : BOOL (valeur par défaut : FALSE)

TRUE indique que la distance souhaitée a été parcourue

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;

Error : BOOL (valeur par défaut : FALSE)

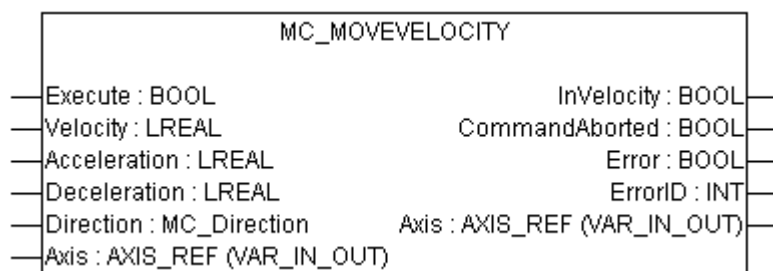
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_MoveVelocity

Ce module de la bibliothèque SM_PLCOpen.lib entraîne un mouvement infini de l'axe à vitesse prédéfinie. Pour atteindre cette vitesse, MC_MoveVelocity s'en tient aux valeurs programmées pour l'accélération et la décélération. La vitesse de consigne est toujours positive. La variable d'entrée Direction définit la direction.

Entrée / sortie (VAR_IN_OUT) du module:**Axis : AXIS_REF**

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

Entrées du module:**Execute : BOOL (valeur par défaut : FALSE)**

Le module est activé avec un front montant.

Velocity : REALValeur de vitesse maximale qui ne doit cependant pas nécessairement être atteinte, [μ/s]**Accélération : REAL**Accélération souhaitée, [u/s^2].]**Deceleration : REAL**Décélération souhaitée, [u/s^2]**Direction : MC_Direction (valeur par défaut : current)**

Cette valeur d'énumération indique le sens souhaité ; voir SM_DriveBasic.lib. Valeurs admissibles : current, positive, negative.

Sorties du module:**InVelocity : BOOL (valeur par défaut : FALSE)**

TRUE indique que la vitesse prédéfinie a été atteinte.

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_PositionProfile

Ce module de la bibliothèque SM_PLCOpen.lib effectue un déplacement selon un profil de position prédéfini. Il faut définir et remplir pour ce faire une variable structurelle MC_TP_REF.

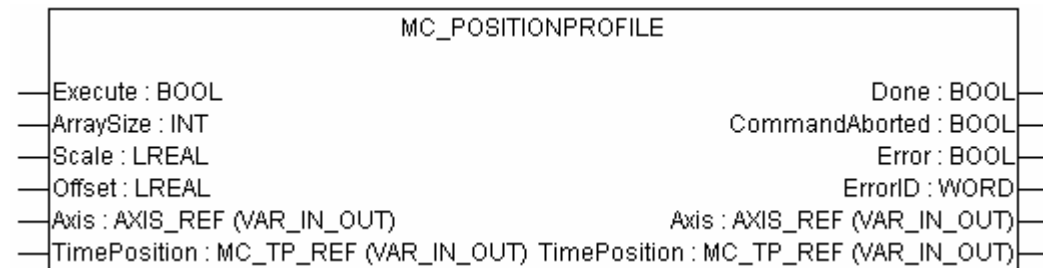
MC_TP_REF contient les variables ci-dessous :

Variable	Type	Valeur init	Description
Number_of_pairs	INT	0	Nombre de points de profil
IsAbsolute	BOOL	TRUE	Positions absolues ou relatives
MC_TP_Array	ARRAY[1..100] OF SMC_TP		Points

SMC_TP contient les variables ci-dessous :

Variable	Type	Valeur init	Description
delta_time	TIME	t#0s	Temps entre le point précédent et le point actuel
position	REAL	0	Position (absolue / relative) du point

Le module dessine via les positions définies une courbe continue double différentiable à partir de polynômes cubiques.



Entrées / sorties (VAR_IN_OUT) du module:

Axis : AXIS_REF

La structure qui au sein de l'interface d'entraînement (SM_DriveBasic.lib) est remplie avec les données des axes est transmise ici.

TimePosition : MC_TP_REF

Mention des valeurs de temps et de position, voir ci-dessus

Entrées du module:

Execute : BOOL (valeur par défaut : FALSE)

Le module est activé avec un front montant.

ArraySize : INT

Taille du tableau (max.1..100)

Scale : REAL (valeur par défaut : 1)

Facteur général de graduation pour le profil

Offset : REAL

Décalage général pour le profil [u]

Sorties du module:**Done : BOOL** (valeur par défaut : FALSE)

TRUE indique que la distance souhaitée a été parcourue

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

MC_VelocityProfile

Ce module de la bibliothèque SM_PLCOpen.lib fonctionne de manière analogue au module MC_PositionProfile. Cependant, les points sont ici définis dans les variables d'entrée de structure MC_TV_REF par le biais de leurs vitesses.

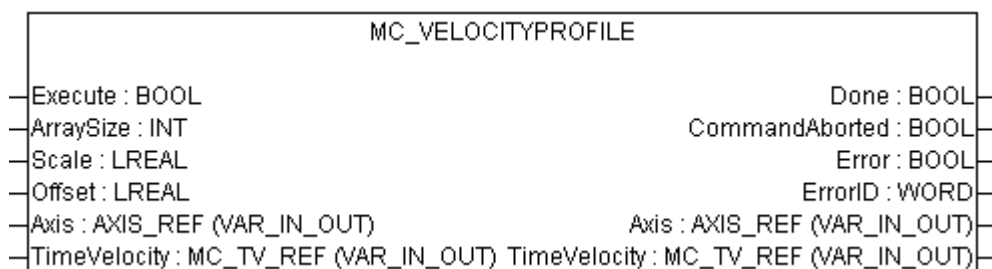
MC_TV_REF contient les variables ci-dessous:

Variable	Type	Valeur init	Description
Number_of_pairs	INT	0	Nombre de points de profil
IsAbsolute	BOOL	TRUE	Positions absolues ou relatives
MC_TV_REF	ARRAY[1..100] OF MC_TV_REF		Points

SMC_TV contient les variables ci-dessous:

Variable	Type	Valeur init	Description
delta_time	TIME	t#0s	Temps entre le point précédent et le point actuel
velocity	REAL	0	Vitesse (absolue / relative) du point

Le module dessine via les points définis une courbe continue à partir de paraboles. La position des axes résulte de la position au départ et de l'intégrale relative à la vitesse.



MC_AccelerationProfile

Ce module de la bibliothèque SM_PLCOpen.lib fonctionne de manière analogue au module MC_PositionProfile. Cependant, les points sont ici définis dans les variables d'entrée de structure MC_TA_REF par le biais de leurs accélérations.

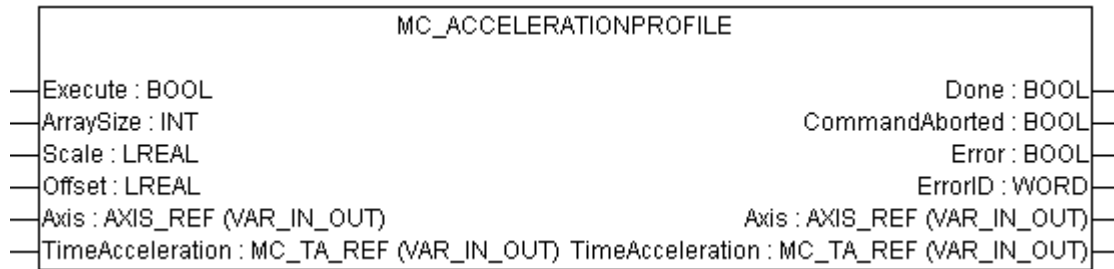
MC_TA_REF contient les variables ci-dessous:

Variable	Type	Valeur init	Description
Number_of_pairs	INT	0	Nombre de points de profil
IsAbsolute	BOOL	TRUE	Positions absolues ou relatives
MC_TA_Array	ARRAY[1..100] OF SMC_TA		Points

SMC_TA contient les variables ci-dessous :

Variable	Type	Valeur init	Description
delta_time	TIME	t#0s	Temps entre le point précédent et le point actuel
acceleration	REAL	0	Accélération (absolue / relative) du point

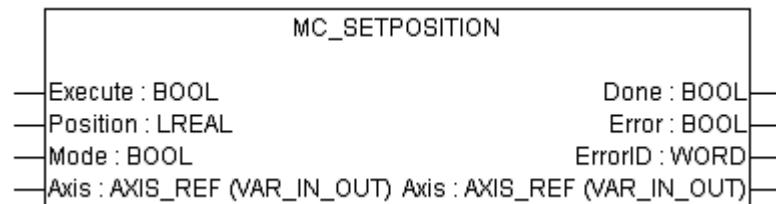
Le module dessine via les points définis une courbe continue à partir de lignes droites. La vitesse de la courbe résulte de la vitesse au départ du profilé et de l'intégrale relative à l'accélération. La position des axes résulte de la position au départ et de l'intégrale relative à la vitesse.

**MC_SetPosition**

Ce module de la bibliothèque SM_PLCOpen.lib décale le point zéro de l'axe de telle sorte que :

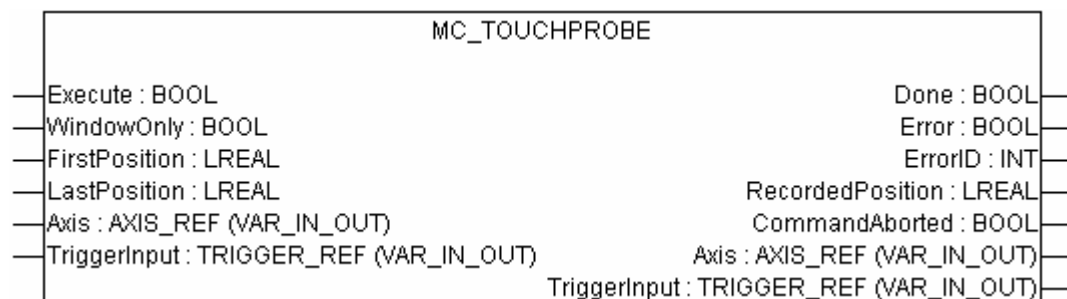
- en mode absolu (mode = FALSE; valeur par défaut), la valeur communiquée par l'entrée Position devienne la position momentanée de consigne, ou
- en mode relatif (mode = TRUE), la position de consigne actuelle soit décalée de la valeur sous Position.

En règle générale, ce module peut être appelé en tout temps. Il faut cependant noter qu'avec un mouvement commandé par trajectoire, si les positions de consigne sont transmises directement au module par le biais p.ex. de SMC_ControlAxisByPos, cela peut entraîner un saut de la position de consigne.



MC_TouchProbe

Ce module de la bibliothèque SM_PLCOpen.lib sert à enregistrer précisément la position de l'entraînement par le biais d'une entrée rapide. Comme ceci doit normalement être effectué plus rapidement que dans le cas de cycles PLC normaux, on a souvent recours à cette fonction pour solliciter l'entraînement ou cette dernière est exécutée – indépendamment de la cadence PLC – entre autres par le biais d'arrêts.



L'entrée TriggerInput est de type TRIGGER_REF et décrit plus précisément l'entrée Trigger:

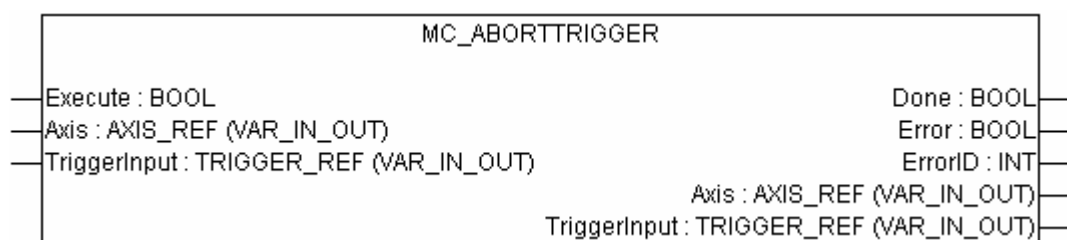
Variable	Type	Valeur init	Description
bFastLatching	BOOL	TRUE	Verrouillage rapide via DriveInterface (TRUE) ou verrouillage à cadence PLC (FALSE)
iTriggerNumber	INT	-1	Uniquement pour bFastLatching = TRUE: numéro d'enclenchement ; indépendant de DriveInterface
bInput	BOOL	FALSE	Uniquement pour bFastLatching = TRUE : signal d'entrée ; TRUE déclenche le verrouillage.
bActive	BOOL	FALSE	Variable interne

La fonction de fenêtre activée et définie par le biais de WindowOnly, FirstPosition, LastPosition, n'est possible que si elle est supportée par DriveInterface ; dans le cas contraire, cela entraîne une erreur.

Le module est fonction du statut de l'axe et reste activé jusqu'à ce qu'une position soit verrouillée ou que le processus ait été interrompu par MC_AbortTrigger.

MC_AbortTrigger

Ce module permet de lever le verrouillage se trouvant sur l'entrée TriggerInput.



5.4 Commande de mouvements synchronisés

MC_CamTableSelect

Ce module de la bibliothèque SM_PLCOpen.lib permet de sélectionner une CAM, de définir les axes maître et esclave et de procéder à quelques réglages préalables. L'objet CamTableID présent sous la forme de sortie devra être ajouté ultérieurement au module CAM MC_CamIn.

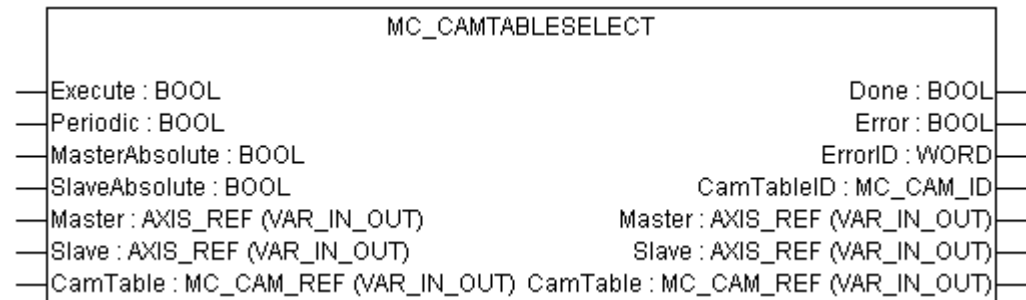
L'axe maître peut être virtuel, il ne doit pas être physiquement présent. Si la variable bRegulatorOn est TRUE, on utilise ses valeurs de consigne ; on reprend sinon ses valeurs réelles.

La CAM à parcourir peut soit être programmée manuellement dans un objet structuré MC_CAM_REF, soit être créée à l'aide de l'éditeur CAM intégré à CoDeSys (voir document „SoftMotion CAM-Editor“).

La variable Periodic définit si la CAM doit à la fin être à nouveau parcourue depuis le début ou non.

MasterAbsolute et SlaveAbsolute permettent de décider si la représentation CAM de l'axe maître sur l'axe esclave doit se rapporter à des valeurs absolues ou à des incréments.

Voir également ICI pour des détails relatifs aux paramètres individuels.



Entrées / sorties (VAR IN OUT) du module:

Master : AXIS_REF

Axe maître.

Slave : AXIS_REF

Axe esclave.

CamTable : MC_CAM_REF

Description de la CAM.

Entrées du module:

Execute : BOOL (valeur par défaut : FALSE)

Avec un front montant, le module choisit une nouvelle CAM.

Periodic : BOOL (valeur par défaut : TRUE)

CAM périodique / non périodique.

MasterAbsolute : BOOL (valeur par défaut : TRUE)

CAM se rapporte à la position maître absolue / relative (position à front montant dans Execute de CAMIn).

SlaveAbsolute : BOOL (valeur par défaut : TRUE)

CAM se rapporte à la position esclave absolue / relative (position à front montant dans Execute de CAMIn).

Sorties du module:**Done : BOOL (valeur par défaut : FALSE)**

TRUE indique que la distance souhaitée a été parcourue

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : SMC_Error (INT)

Numéro d'erreur

CAMTableID : MC_CAM_ID

Sortie qui décrit la CAM. Sert d'entrée pour l'entrée du même nom in MC_CamIn.

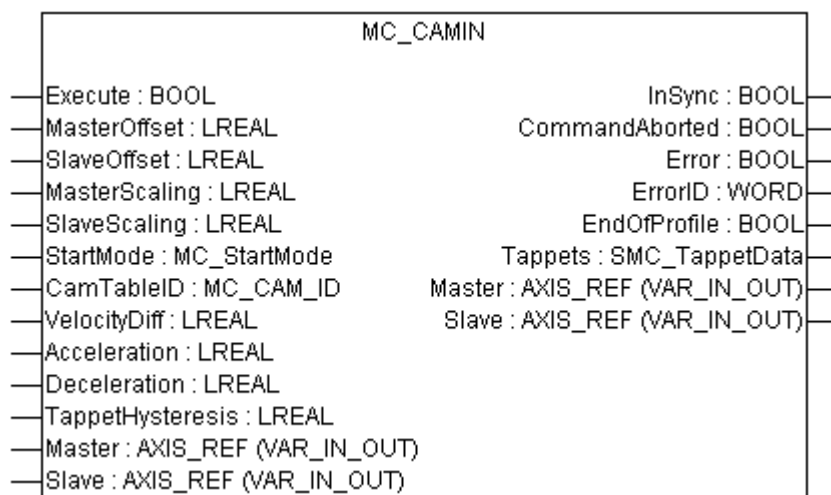
MC_CamIn

Ce module de la bibliothèque SM_PLCopen.lib convertit une CAM sélectionnée dans MC_CAMTABLESELECT.

Le mode de démarrage peut encore être défini pour les décalages et les graduations. Il convient de noter que Modi ramp_in, ramp_in_pos et ramp_in_neg - fonctions qui ont pour but de fournir en permanence la valeur de consigne de l'esclave à la CAM pour autant que la valeur réelle de l'esclave et la valeur de départ CAM ne coïncident pas au démarrage – supposent que les paramètres VelocityDiff, Acceleration et Deceleration soient définis.

ce module contient une fonction supplémentaire. Il détecte les cames et est à même de les transmettre via la sortie Tappets à un ou plusieurs modules fonctionnels SMC_GetTappetValue (voir SMC_GetTappetValue). Il convient de noter que le module CamIn peut enregistrer par cycle un maximum de cames simultanément. Le module SMC_CAMRegister n'est quant à lui pas concerné par cette restriction.

Voir également ICI pour des détails relatifs aux paramètres individuels.

Entrées / sorties (VAR IN OUT) du module :**Master : AXIS_REF**

Axe maître.

Slave : AXIS_REF

Axe esclave.

Entrées du module :**Execute : BOOL (valeur par défaut : FALSE)**

Le module et le mouvement débutent avec un front montant.

MasterOffset : LREAL (valeur par défaut : 0)

Décalage supplémentaire sur la position maître.

SlaveOffset : LREAL (valeur par défaut : 0)

Décalage supplémentaire sur la position esclave.

StartMode : MC_StartMode (absolute/relative/ramp_in/ramp_in_pos/ramp_in_neg) (valeur par défaut : absolute)

CAM est rapportée soit de manière relative (relative) par rapport à la position réelle, soit de manière absolue sans (absolute) ou avec rampe lente sur la trajectoire la plus courte (ramp_in), dans le sens positif (ramp_in_pos) ou négatif (ramp_in_neg).

CamTableID : MC_CAM_ID

Sortie de MC_CamTableSelect

Velocity, Acceleration, Deceleration: LREAL (valeur par défaut : 0)

Vitesse, accélération et décélération supplémentaires pour le mode ramp_in.

TappetHysteresis: LREAL (valeur par défaut : 0)

Largeur de bande d'hystérèse autour des cames.

Sorties du module :

InSync : BOOL (valeur par défaut : FALSE)

TRUE indique que le mouvement est en cours sur la CAM.

CommandAborted : BOOL (valeur par défaut : FALSE)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : SMC_Error (INT)

Numéro d'erreur

EndOfProfile : BOOL

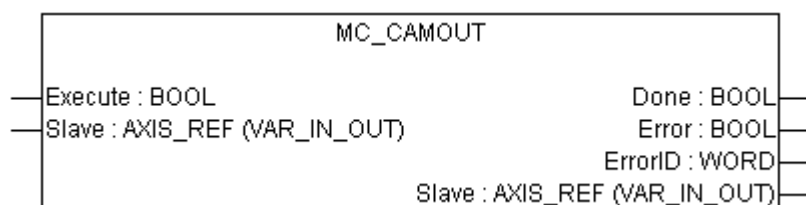
Indique la fin d'une CAM. Avec des CAM périodiques, cette sortie est par impulsions.

Tappets : SMC_TappetData

Sortie de came. Les positions individuelles des cames sont évaluées à titre définitif par le module SMC_GetTappetValue.

MC_CamOut

Ce module de la bibliothèque SM_PLCOpen.lib découple l'entraînement esclave du maître et continue à commander l'esclave selon la vitesse momentanée.

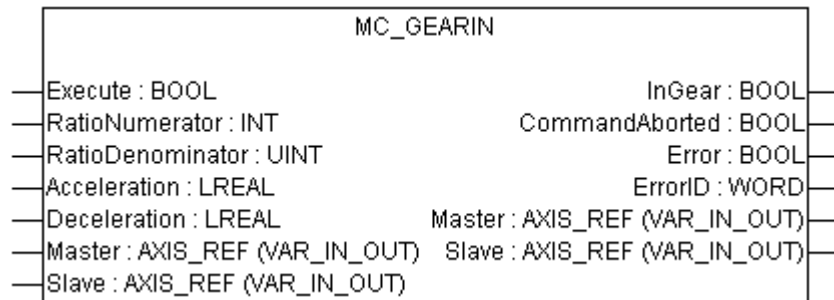


MC_GearIn

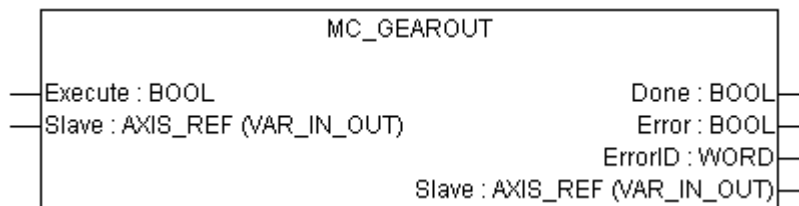
Ce module de la bibliothèque SM_PLCopen.lib couple l'axe esclave à l'axe maître. La vitesse de l'axe esclave est alors f fois plus grande que celle de l'axe maître. Ce facteur f est calculé à partir du quotient des paramètres de saisie RatioNumerator et RatioDenominator.

Le module accélère ou freine l'axe esclave jusqu'à ce que sa vitesse présente le rapport souhaité comparé à l'axe maître ; il utilise pour ce faire les valeurs Accelération et Deceleration. Dès que ceci est effectué, on peut déduire la vitesse de l'axe esclave de celle de l'axe maître.

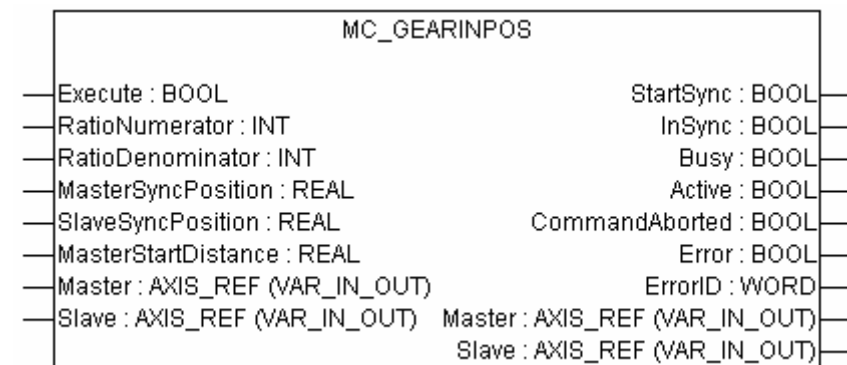
Si la variable bRegulatorOn de l'axe maître est TRUE, ses valeurs de consigne pour la vitesse sont utilisées, dans la négative, on utilise les valeurs réelles.

**MC_GearOut**

Ce module de la bibliothèque SM_PLCopen.lib découple l'entraînement esclave du maître et continue à commander l'esclave selon la vitesse momentanée.

**SMC_GearInPos**

Ce module de la bibliothèque SM_PLCopen.lib découple de manière synchrone la position et la vitesse des axes esclave et maître. Les positions, à partir desquelles l'axe esclave se couple peuvent être définies par les entrées.



Entrées / sorties (VAR_IN_OUT) du module :

Master : AXIS_REF

Axe maître.

Slave : AXIS_REF

Axe esclave.

Entrées du module :

Execute : BOOL (valeur par défaut : FALSE)

Lors du flanc montant, le module débute par le couplage. Si MasterStartDistance est 0, la couplage démarre directement et l'axe esclave accélère avec la fonction polynôme. À position donnée du maître, l'esclave se trouve à vitesse synchrone et à la position définie. Si MasterStartDistance est supérieure à 0, l'axe esclave attend jusqu'à ce que le maître se trouve dans la fenêtre de couplage et commence alors à accélérer. En fonction de la vitesse de l'esclave avant le couplage, la vitesse de synchronisation et la position entrée du maître et de l'esclave, ou du chemin de couplage, l'axe peut se déplacer brièvement dans la direction inverse de la rotation propre car seules les conditions données peuvent être respectées.

RatioNumerator: INT (valeur par défaut : 1)

Le compteur est configuré pour un rapport de traduction.

RatioDenominator: INT (valeur par défaut : 1)

Un dénominateur du rapport de traduction peut être défini.

MasterSyncPosition: REAL (valeur par défaut : 1)

L'esclave est entièrement couplé à partir de cette position.

SlaveSyncPosition: REAL (valeur par défaut : 1)

Position de l'esclave quand le maître se trouve à MasterSyncPosition.

MasterStartDistance: REAL (valeur par défaut : 1)

Parcours de couplage souhaité ou début de l'accélération si MasterStartDistance = 0.

Sorties du module :

StartSync: BOOL (valeur par défaut : 1)

La sortie est TRUE tant que l'axe esclave est en couplage. Dès que l'axe est entièrement couplé, StartSync devient FALSE et InSync devient TRUE.

InSync: BOOL (valeur par défaut : 1)

Cette sortie indique TRUE si l'esclave est couplé.

Busy: BOOL (valeur par défaut : 1)

Indique que le module n'est pas terminé.

Active: BOOL (valeur par défaut : 1)

Indique que le bloc fonctionnel contrôle l'axe.

CommandAborted : BOOL (valeur par défaut : 1)

Le mouvement entamé a été interrompu par un autre bloc fonctionnel qui agit sur le même axe ;
exception : MoveSuperImposed

Error : BOOL (valeur par défaut : 1)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : SMC_Error (INT)

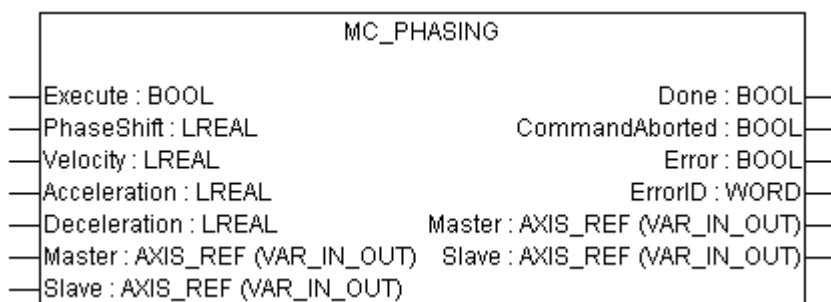
Numéro 'erreur

MC_Phasing

Ce module de la bibliothèque SM_PLCOpen.lib assure une distance constante entre l'axe maître et l'axe esclave. Dans ce cas, le maître et l'esclave présentent bien entendu les mêmes valeurs de vitesse et d'accélération.

Pour y arriver, l'axe esclave obtient la même vitesse que l'axe maître par le biais d'une accélération ou d'une décélération. Suite à cela, l'axe maître effectue un mouvement supplémentaire (similaire à MC_MoveSuperImposed) qui entraîne le décalage de phase souhaité.

Le module MC_Phasing reste activé jusqu'à ce qu'il soit interrompu par un autre module.



5.5 Modules supplémentaires

SMC_GetCamSlaveSetPosition

Ce module de la bibliothèque SM_PLCOpen.lib calcule la position de consigne actuelle d'un axe (esclave), comme si celui-ci était relié aux mouvements d'un autre axe (maître) par le biais d'une CAM. Les deux axes ne sont lors de ce calcul ni déplacés, ni influencés.

Ce module est utilisé lorsqu'un axe esclave doit, avant accouplement au sein d'une CAM, être déplacé à la position de consigne résultant de cet accouplement.

Le module calcule la valeur appropriée au sein d'un cycle, ce qui rend inutile toute sortie Done.

Entrées / sorties (VAR IN OUT) du module :

CamTableID : MC_CAM_ID

CAM ; sortie de MC_CamTableSelect.

Master : AXIS_REF

Axe maître.

Slave : AXIS_REF

Axe pour lequel la position de consigne CAM est calculée.

Entrées (VAR IN) du module (les entrées non décrites correspondent à celles de MC_CamIn):

Enable : BOOL

Active le module.

Sorties du module:

fStartPosition : LREAL

Position de consigne calculée pour l'esclave.

Error : BOOL (valeur par défaut : FALSE)

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : SMC_Error (INT)

Numéro d'erreur

SMC_CAMBounds

Ce module de la bibliothèque SM_PLCOpen.lib calcule les valeurs de consigne maximales de Position, Vitesse et Accélération/Décélération de l'esclave si ce dernier est relié de manière absolue à un maître par le biais d'une CAM donnée, ce maître étant commandé via Vitesse et Accélération/Décélération maximales prédéfinies.

Ce module s'avère particulièrement utile lorsqu'une CAM est créée ou éditée en ligne et que l'on souhaite contrôler au préalable le respect des valeurs maximales.

Entrées (VAR INPUT) du module :

bExecute : BOOL

Active le calcul.

dMasterVelMax : LREAL (valeur par défaut : 1)

Vitesse maximale (valeur absolue) du maître [u/s].

dMasterAccMax : LREAL (valeur par défaut : 0)

Accélération/Décélération maximale (valeur absolue) du maître [u/s²].

dMasterScaling, dSlaveScaling : LREAL (valeur par défaut : 1)

Facteur de mise à l'échelle du maître ou de l'esclave qui est utilisé pour la CAM.

Entrées / sorties (VAR IN OUT) du module :

CAM : MC_CAM_REF

CAM qui doit être représentée et éditée.

Sorties (VAR OUTPUT) du module :

bDone : BOOL

TRUE indique que le calcul est terminé.

bError : BOOL

TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

nErrorID : SMC_Error (INT)

Numéro d'erreur

dMaxPos, dMinPos : LREAL

Valeur maximale ou minimale de position de l'esclave [u].

dMaxVel, dMinVel : LREAL

Valeur maximale ou minimale (également négative) de vitesse de l'esclave [u/s].

dMaxAccDec, dMinAccDec : LREAL

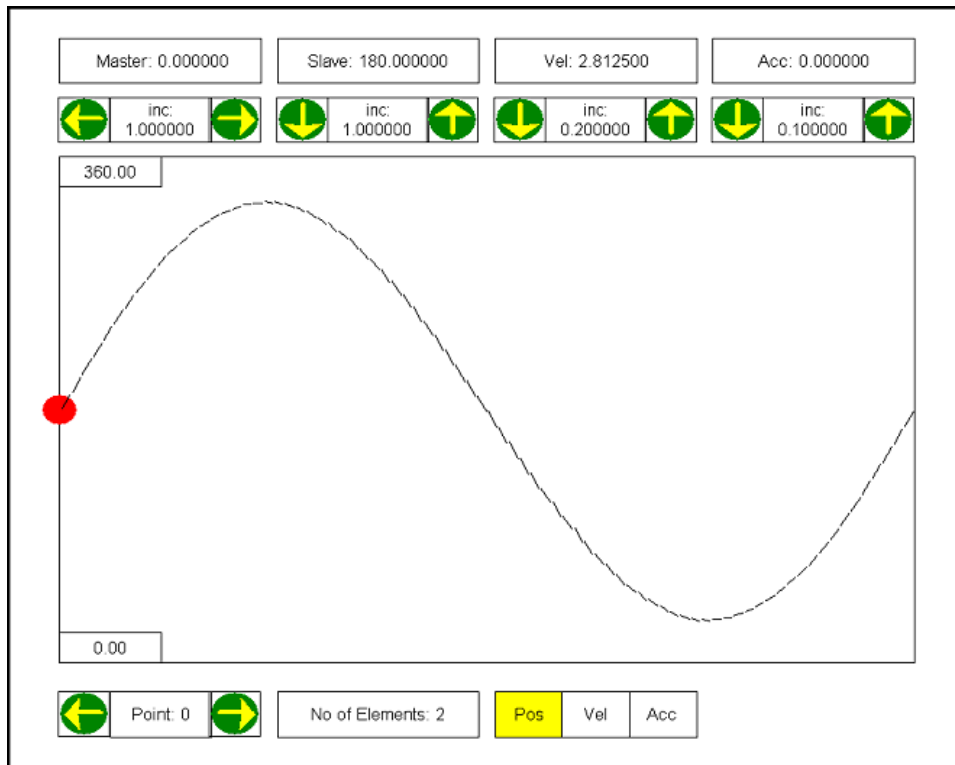
Valeur maximale ou minimale (également négative) d'accélération et de décélération de l'esclave [u/s²].

SMC_CAMEditor, SMC_CAMVisu

Ce module de la bibliothèque SM_PLCOpen.lib permet de créer un éditeur CAM en ligne.

SMC_CAMEditor doit être appelé au sein de la tâche SoftMotion alors que SMC_CAMVisu est appelé dans une tâche plus lente et moins prioritaire.

Les deux modules doivent être reliés au moyen du modèle de visualisation approprié (SMC_CAMEditor) qui représente les CAM saisies et permet à l'utilisateur de les éditer, même si elles sont en cours d'exécution.



Le cercle rouge indique le point CAM actuel. Celui-ci peut être changé en dessous à droite au moyen de la flèche. La barre en dessous à droite permet de sélectionner si la position, la vitesse ou l'accélération doit être affichée. Les flèches au dessus permettent de décaler la position, la vitesse et l'accélération du maître et de l'esclave de l'incrément indiqué.

Entrées / sorties (VAR IN OUT) du module SMC_CAMEditor:

CAM : MC_CAM_REF

CAM qui doit être représentée et éditée.

Entrées du module SMC_CAMEditor :

Enable : BOOL (valeur par défaut : FALSE)

Le module fonctionne si cette entrée est TRUE.

dYPeriod: LREAL

Période esclave (pour CAM périodiques).

bPeriodic: BOOL (valeur par défaut : TRUE)

TRUE pour des CAM périodiques, sinon FALSE.

Entrées / sorties (VAR IN OUT) du module SMC_CAMEditor:

ce: SMC_CAMEditor

Instance de SMC_CAMEditor

SMC_CAMRegister

Ce module de la bibliothèque SM_PLCopen.lib représente une came mécanique. Comme MC_CamIn, il fonctionne sur base d'une structure MC_CAM_REF, ignorant les informations CAM effectives et ne lisant que les informations se rapportant exclusivement aux comes.

Entrées / sorties (VAR IN OUT) du module :

Master : AXIS_REF

La structure d'axe qui doit commuter les comes est transmise ici.

CamTable : MC_CAM_REF

Description d'une CAM (même vide) qui contient les descriptions des cames.

bTappet : ARRAY [1..MAX_NUM_TAPPETS] OF BOOL

Bits de came.

Entrées du module:

Enable : BOOL (valeur par défaut : FALSE)

Avec TRUE, le module commence à commuter les cames.

MasterOffset : REAL

Décalage sur la position maître.

MasterScaling : REAL (valeur par défaut : 1)

Graduation générale pour l'axe maître

TappetHysteresis: REAL

Environnement d'hystérèse pour les cames.

DeadTimeCompensation : REAL

Temps mort en sec. Une extrapolation linéaire permet de calculer au préalable la position attendue de l'axe maître.

Sorties du module:

Error : BOOL (valeur par défaut : FALSE)

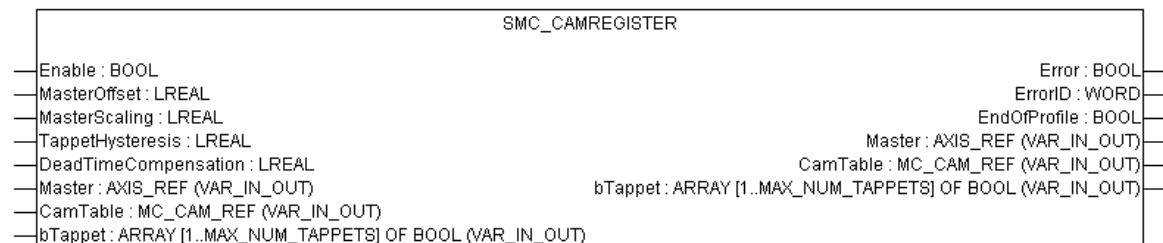
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : INT

Numéro d'erreur

EndOfProfile : BOOL

Lors du passage de la fin du profil au début du profil, cette sortie est TRUE le temps d'un cycle.

**SMC_GetTappetValue**

Ce module de la bibliothèque SM_PLCOpen.lib évalue la sortie Tappet du module MC_CamIn et fournit le statut actuel d'une came.

Entrées / sorties (VAR_IN_OUT) du module :**Tappets : SMC_TappetData**Entrées du module :**iID : INT**

ID du groupe de la came à évaluer

bInitValue : BOOL

Valeur initiale de la came. Est définie lors du premier appel.

bSetInitValueAtReset : BOOL

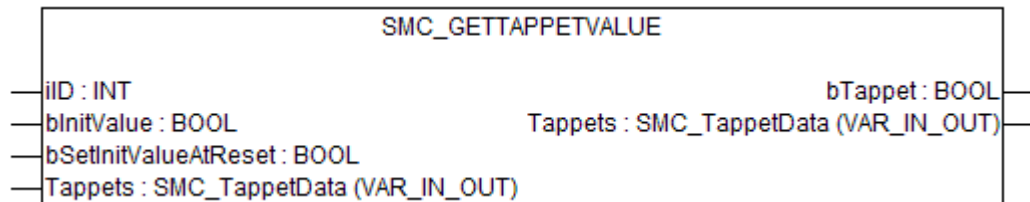
TRUE : au redémarrage du bloc fonctionnel CamIn, la valeur de la came est réglée sur blnitValue.

FALSE: au redémarrage du bloc fonctionnel CamIn, la valeur de la came reste intacte.

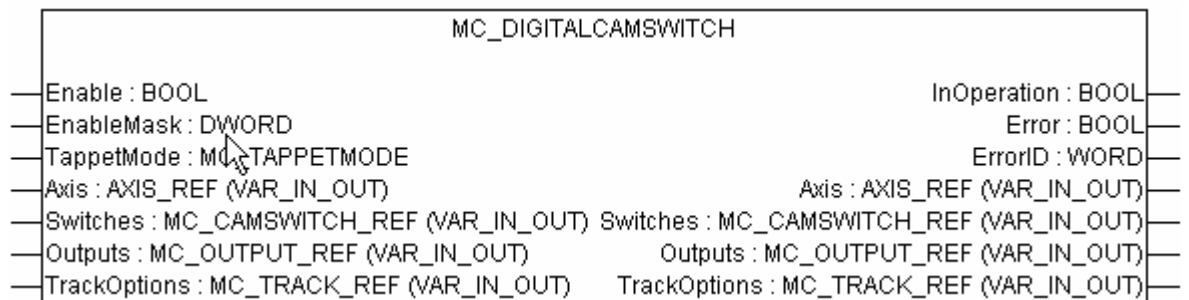
Sorties du module :

bTappet : BOOL (valeur par défaut : FALSE)

Valeur de la came

**MC_DigitalCamSwitch**

Ce module représente une came mécanique (comme SMC_CamRegister). Un maximum de 32 sorties peuvent être connectées. Les mouvements peuvent être dans tous les sens.

Entrées / sorties (VAR_IN_OUT) du module :**Axis : AXIS_REF**

Structure de l'axe que la came doit commuter.

Switches: MC_CAMSWITCH_REF

Définition des positions de commutation.

Les structures sont construites de la façon suivante:

```

TYPE MC_CAMSWITCH_REF :
STRUCT
    NoOfSwitches: BYTE;
    CamSwitchPtr: POINTER TO MC_CAMSWITCH_TR;
END_STRUCT
END_TYPE

```

NoOfSwitches donne le nombre de positions de commutation, CamSwitchPtr affiche un array de type MC_CAMSWITCH_TR.

```

TYPE MC_CAMSWITCH_TR :
STRUCT
    TrackNumber: INT;
    FirstOnPosition: LREAL;
    LastOnPosition: LREAL;
    AxisDirection: INT;
    CamSwitchMode: INT;
    Duration: TIME;

```

```
END_STRUCT
END_TYPE
```

Le numéro de tracé donne le numéro de sortie. Il est possible d'avoir plusieurs zones de position par sortie.

FirstOnPosition donne la position d'enclenchement de la sortie, LastOnPosition, la position de déclenchement.

AxisDirection 0: La sortie est enclenchée dans les 2 directions ; 1 uniquement pour la direction positive ; 2 uniquement négative.

CamSwitchMode : la valeur 0 signifie que les cames sont calculées en fonction de la position ; la valeur 1 signifie que l'enclenchement se fait en mode basé sur le temps. Ainsi, uniquement la valeur de FirstOnPosition est utilisée et la sortie reste sur TRUE pour une période définie (duration).

Example:

```
Positions: ARRAY [1..4] OF MC_CAMSWITCH_TR :=
(TrackNumber:=1,FirstOnPosition:=200,LastOnPosition:=300,AxisDirection:=2,CamSwitchMode:=0),
(TrackNumber:=2,FirstOnPosition:=100,LastOnPosition:=300,CamSwitchMode:=0),
(TrackNumber:=3,FirstOnPosition:=10,CamSwitchMode:=1,Duration:=T#2000ms),
(TrackNumber:=1,FirstOnPosition:=0,LastOnPosition:=100,AxisDirection:=1);
```

La sortie 1 est enclenchée dans la direction positive entre la position 0 et 100 et dans la direction négative entre la position 200 et 300.

La sortie 2 est enclenchée dans toutes les directions entre 100 et 300..

La sortie 3 est enclenchée dans toutes les directions pendant 2 secondes à la position 10.

Outputs: MC_OUTPUT_REF

Les 32 sorties de cames sont déterminées.

```
TYPE MC_OUTPUT_REF :
ARRAY [0..31] OF BOOL;
END_TYPE
```

TrackOptions: MC_TRACK_REF

D'autres paramètres peuvent être configurés pour chaque sortie de came grâce aux TrackOptions.

```
TYPE MC_TRACK_REF :
ARRAY [0..31] OF MC_TRACK_TR;
END_TYPE
```

Chaque élément de l'Array est utilisé pour la sortie correspondante.

```
TYPE MC_TRACK_TR :
STRUCT
OnCompensation: LREAL;
OffCompensation: LREAL;
Hysteresis: LREAL;
END_STRUCT
END_TYPE
```

OnCompensation: Un ralentissement (valeur positive) ou un processus d'enclenchement anticipé (valeur négative) peut être configuré. Le temps est donné en secondes.

OffCompensation: Un ralentissement (valeur positive) ou un processus de déclenchement anticipé (valeur négative) peut être configuré. Le temps est donné en secondes.

Hysteresis: Un hystérésis peut être également configuré. Ceci évite une commutation continue de la sortie, quand, par exemple, un servorégulateur se trouve exactement sur la position de commutation et qu'une oscillation se crée.

Entrées du module:**Enable : BOOL (Default: FALSE)**

TRUE : Le module débute la commutation des cames.

EnableMasks: DWORD (Default: 16#FFFFFFFF)

Les sorties peuvent être enclenchées et déclenchées. Quand le bit est 1, la sortie correspondante est active. Le bit le plus bas correspond à la sortie la plus basse. La valeur par défaut enclenche toutes les sorties de came. Avec 1 comme valeur, seule la première sortie est enclenchée.

TappetMode: MC_TappetMode (valeur par défaut : 1): tp_mode_auto)

Avec le TappetMode, il est possible de configurer sur quelles données de position les cames sont calculées. Dans la configuration tp_mode_auto, la position censée est affectée quand l'axe est enclenché. Sinon, la position réelle sera utilisée pour le calcul. Dans la configuration tp_mode_demandposition, la position censée est utilisée et la position réelle pour tp_mode_actualposition.

Sorties du module:**InOperation: BOOL** (valeur par défaut : FALSE)

TRUE: indique que le module est actif et que les cames vont être calculées.

Error : BOOL (valeur par défaut : FALSE)

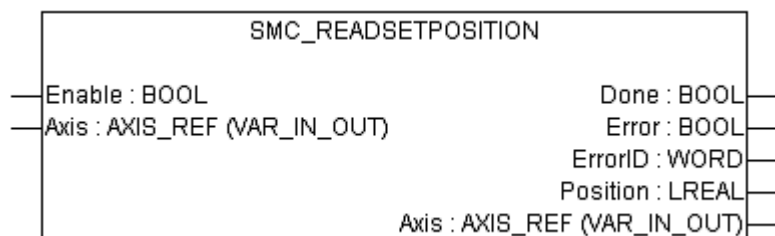
TRUE indique qu'une erreur est survenue au sein du bloc fonctionnel.

ErrorID : SMC_Error (INT)

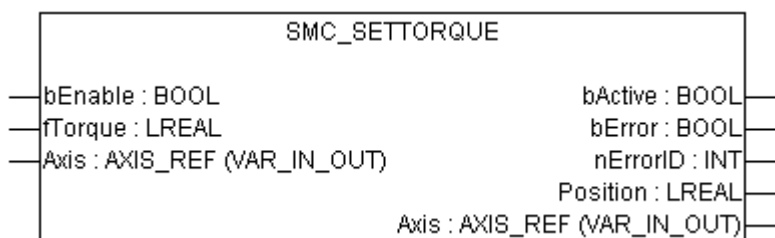
Numéro 'erreur

SMC_ReadSetPosition

Ce module de la bibliothèque SM_PLCopen.lib lit la position de consigne actuelle de l'entraînement..

**SMC_SetTorque**

Ce module permet de créer un couple de consigne, pour autant que l'entraînement se trouve en mode de régulation „couple“.



6 Bibliothèque SM_CNC.lib

6.1 Aperçu

Cette **bibliothèque de module** propose des modules et fonctions pour un programme IEC réalisé avec SoftMotion. Elle contient des fonctions de déplacement et est intégrée au programme IEC. Elle comprend entre autres les modules énumérés ci-dessous, ceux-ci exécutant d'une part les mouvements définis en ligne au sein des éditeurs, et réalisant d'autre part les trajectoires définies en ligne au sein du programme IEC. Pour ce faire, ils accèdent à des structures internes qui écrivent les objets individuels de trajectoire.

- SMC_NCDDecoder (décodage de la trajectoire programmée dans l'éditeur CNC en objets structurels)
- SMC_GCodeViewer (représentation du code G)
- SMC_ToolCorr (préparation de trajectoire : correction de rayon d'outil)
- SMC_AvoidLoop (préparation de trajectoire : éviter les boucles)
- SMC_SmoothPath (préparation de trajectoire : lisser coins avec splines)
- SMC_RoundPath (préparation de trajectoire : arrondir coins avec arcs de cercle)
- SMC_CheckVelocities (contrôle des vitesses finales des segments)
- SMC_Interpolator (conversion des objets de trajectoire décodés et le cas échéant préparés, en points discrets)
- SMC_GetMParameters (scruter les paramètres de la fonction M)
- Fonctions auxiliaires pour déplacer et faire pivoter une trajectoire créée
- Variables globales pour définir les paramètres internes
- Structures SMC_POSINFO, SMC_GEOINFO, SMC_VECTOR3D et SMC_VECTOR6D (enregistrement de positions, sections de trajectoire et vecteurs)
- Structure SMC_OUTQUEUE (gestion des objets GEOINFO dans une liste de taille définie)

(Voir à ce sujet également les exemples de programmation)

Remarque : il faut noter que l'éditeur CNC peut compiler un programme CNC de deux manières différentes : soit en tant que variable de programme (SMC_CNC_REF) qui doit parcourir les modules de décodeur et le cas échéant de préparation de trajectoire, soit en tant que mémoire OUTQUEUE qui peut transmettre les données directement à l'interpolateur.

6.2 Modules

6.2.1 SMC_NCDDecoder

Le module SMC_NCDDecoder a pour tâche de convertir un programme CNC créé dans l'éditeur CNC en une liste d'objets structurels SoftMotion GEOINFO.

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame le décodage dès que cette entrée présente un front montant.

LbAppend : BOOL

Si cette entrée est FALSE, la queue de sortie DataOut est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue DataOut.

bStepSuppress: BOOL

Si cette entrée est TRUE, les lignes du programme CNC qui commencent avec „/“ sont ignorées. Si l'entrée est FALSE (valeur pas défaut), ces lignes sont cependant exécutées.

piStartPosition

Position que le point à déplacer présente en début de trajectoire.

piStartPosition

Position que le point à déplacer présente en début de trajectoire.

nSizeOutQueue: UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structurels GEOINFO est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure GEOINFO. Si tel n'est pas le cas, le module SMC_NCD decoder ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation. Il est recommandé de créer la mémoire tampon effective comme décrit ci-dessous p.ex. dans `Exemple : Array of SMC_GeoInfo`. La taille appropriée de la mémoire tampon résulte de `sizeof(Exemple)`.

pbyBufferOutQueue: POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure OUTQUEUE. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par nSizeOutQueue. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un tableau SMC_GeoInfo (p.ex. `BUF : ARRAY[1..50] OF SMC_GEOINFO`; pour une mémoire tampon qui peut contenir 50 éléments de trajectoire). Cette valeur peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Entrée / sortie (VAR_IN_OUT) du module :

ncprog: SMC_CNC_REF

Le programme CNC (structure SMC_CNC_REF de la bibliothèque SM_DriveBasic.lib) est transmis dans cette variable IN_OUT. Cette variable est soit créée par le programme IEC, soit programmée au sein de l'éditeur CNC.

À l'aide de ces données, le module SMC_NCD decoder peut décoder une ligne de programme par cycle. Les résultats sont représentés comme suit :

Sorties du module :

bDone : BOOL

Cette variable devient TRUE lorsque l'exécution du programme est terminée. Sauf réinitialisation, SMC_NCD decoder n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Cette sortie d'énumération décrit le cas échéant une erreur survenue lors du décodage. Suite à une erreur, l'exécution est interrompue jusqu'à une réinitialisation.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui gère les objets SMC_GEOINFO décodés.

iStatus : SMC_DEC_STATUS (INT)

Cette variable d'énumération indique le statut actuel du module. Voici les statuts possibles :

WAIT_PROG	0	Programme pas encore trouvé
READ_WORD	1	Mot lu.
PROG_READ	2	Fin de programme atteinte.

iLineNumberDecoded : INT

Ces variables reprennent le numéro de la ligne (en non du jeu) exécutée en dernier.

6.2.2 SMC_GCodeViewer

Ce module de la bibliothèque SM_CNC.lib peut être utilisé en combinaison avec SMC_NCDecoder. Il recueille et enregistre la représentation textuelle (G-Code) de chaque objet GeoInfo et restitue celle-ci sous la forme d'un ARRAY OF STRING, lequel peut être édité dans la visualisation p.ex. via un tableau. Une entrée normalement reliée à la sortie de l'interpolateur iActObjectSourceNo permet au module de recevoir des informations sur la ligne déjà exécutée et de supprimer celle-ci. Il est ainsi possible de créer un élément d'affichage du code G actuel.

Il convient de noter que la sortie de ce module ne coïncide pas avec le code G enregistré dans l'Éditeur ou le fichier texte, vu qu'on ne peut p.ex. pas reprendre ici de ligne vide ni de commentaires. Les variables éventuellement utilisées sont également déjà remplacées.

Le module est appelé au sein du contexte de tâche du module SMC_NCDecoder.

Il convient de lui transmettre une mémoire tampon de taille suffisamment grande. Le nombre d'objets SMC_GCodeViewer_DATA doit être au moins aussi grand que la somme de tous les objets SMC_GeoInfo qui sont enregistrés dans les mémoires tampons des modules SMC_NCDecoder et de préparation de trajectoire.

Entrées du module :

bEnable : BOOL

Le module est activé si cette entrée est TRUE.

iActObjectSourceNo: INT

Numéro de la ligne actuellement exécutée par l'interpolateur.

uiNumberOfBufferEntries : UINT

Taille des tableaux transmis à pBuffer.

pBuffer : POINTER TO SMC_GCODEVIEWER_DATA;

Adresse du tableau qui a été réservé pour la mémoire tampon du module.

Entrée / sortie (VAR IN OUT) du module :

GCodeText : SMC_GCODE_TEXT

Source des données. Est reliée à la sortie du même nom du module SMC_NCDecoder.

Sorties du module :

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

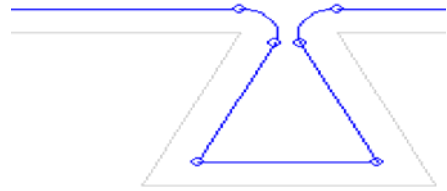
Cette sortie d'énumération décrit le cas échéant une erreur survenue. Après une erreur, l'exécution des données est interrompue jusqu'à ce que le module soit à nouveau démarré par une nouvelle bEnable.

asGCode : ARRAY[0..c_uiLines] OF STRING

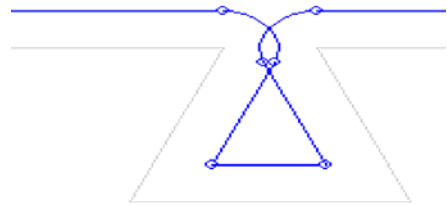
Lignes de texte que le segment actuel de code G contient. Le premier élément reprend la ligne qui est actuellement parcourue par l'interpolateur.

6.2.3 SMC_ToolCorr

Le module SMC_ToolCorr sert à la préparation de la trajectoire : À partir d'une trajectoire donnée, il crée une **trajectoire décalée** dans laquelle chaque point de chaque tronçon présente par rapport à son original et à son voisin direct une distance pouvant être spécifiée (correction de rayon d'outil). La trajectoire ainsi créée garantit ainsi que chacun de ses points présente une **distance fixe** par rapport à l'original. Une application typique est le fraisage d'un contour programmé avec une fraise d'une certaine épaisseur. La fraise doit parcourir une trajectoire décalée en conséquence – créée à l'aide du module SMC_ToolCorr – afin de pouvoir compenser le rayon de la fraise.



Il convient de noter la restriction ci-dessous : Si le contour et le rayon de fraisage sont sélectionnés de telle sorte que l'on trouve des points de recoupement de la trajectoire décalée avec elle-même – et donc le contour souhaité est endommagé par le parcours de la trajectoire décalée – il n'en est pas tenu compte (voir esquisse ci-contre). Afin d'éviter ce problème, il faut utiliser le module SMC_AvoidLoop.



Le module SMC_ToolCorr fonctionne comme suit :

Tous les objets SMC_GEOINFO se trouvant dans l'entrée de structure OUTQUEUE sont parcourus les uns après les autres. Si le bit1 (bit2) de la variable Intern_Mark est activé dans un de ces objets, on commence en cet endroit à décaler la trajectoire dans le sens de déplacement vers la gauche (droite) d'une valeur correspondant au rayon d'outil actuel. Afin d'obtenir une trajectoire constante, un objet de positionnement (MoveType = 100) est inséré, et lorsque l'objet précède un tel objet de positionnement, celui-ci est directement décalé sur le point de démarrage de la trajectoire décalée. Tous les objets suivants sont également décalés jusqu'à ce que le bit0 de Intern_Mark soit activé, ce qui entraîne l'interruption de la correction du rayon d'outil. Ici aussi, un objet de positionnement garantit une suite constante de la trajectoire. Une correction de rayon d'outil entamée doit tout d'abord être clôturée par l'activation du bit0 avant de pouvoir commencer le décalage dans l'autre sens. Le module SMC_NCDecoder active ces bits en réaction aux commandes G41/G42/G40. En d'autres termes, on ponce les coins de tous les objets qui se situent au sein des commandes G41 et G40 et G42 et G40.

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame la correction du rayon d'outil dès que cette entrée présente un front montant.

bAppend : BOOL

Si cette entrée est FALSE, la queue de sortie DataOut est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue DataOut.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire à décaler ; cet objet se trouve normalement sur la sortie DataOut du module précédent (p.ex. du SMC_NCDecoders).

dToolRadius : LREAL

Cette variable d'entrée reprend la valeur qui, ajoutée à la valeur ToolRadius de l'objet SMC_GEOINFO, indique le rayon d'outil dont la trajectoire doit être décalée (voir ci-dessus). Cette valeur peut être modifiée en ligne. Ainsi, on dispose de la possibilité de saisir cette valeur hors ligne

(via la structure SMC_GEOINFO) et de la moduler en ligne. Il convient cependant de noter qu'une modification du rayon d'outil lors de l'usinage d'un bloc à décaler entraîne l'interruption de la correction de rayon d'outil et est donc à éviter. Ceci est cependant tout à fait possible lors d'une remise à zéro ou lorsqu'il est certain que le module ne se trouve pas à l'instant en phase de décalage d'un bloc (`statut = TC_ORIG`). Valeur par défaut : 0.

nSizeOutQueue : UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structuraux GEOINFO est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure GEOINFO. Si tel n'est pas le cas, le module SMC_NCDecoder ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation. Il est recommandé de créer la mémoire tampon effective comme décrit ci-dessous p.ex. dans `Exemple`: `Array of SMC_GeoInfo`. La taille appropriée de la mémoire tampon résulte de `sizeof(Exemple)`.

pbyBufferOutQueue : POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure OUTQUEUE. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par nSizeOutQueue. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un tableau SMC_GeoInfo (p.ex. `BUF : ARRAY[1..50] OF SMC_GEOINFO`; pour une mémoire tampon qui peut contenir 50 éléments de trajectoire). Cette valeur peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Sorties du module :

b_Done : BOOL

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Si une erreur survient, cette valeur devient TRUE.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui contient la trajectoire décalée sous forme de liste d'objets SMC_GEOINFO.

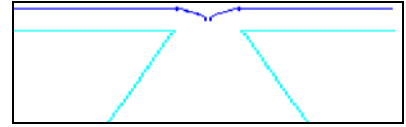
iStatus : SMC_TC_STATUS (INT)

Cette variable d'énumération indique le statut actuel du module. Voici les statuts possibles :

TC_ORIG	0	Pas de correction de rayon d'outil sur l'objet actuel.
TC_RIGHT	1	Objet décalé vers la droite.
TC_LEFT	2	Objet décalé vers la gauche.
TC_END	4	Exécution des objets terminée.

6.2.4 SMC_AvoidLoop

Le module *SMC_AvoidLoop* sert à la préparation de la trajectoire : À partir d'une trajectoire définie, il crée une copie sans boucle de cette trajectoire. S'il existe sur la trajectoire originale un point en lequel elle se recoupe, elle est interrompue en ce point, la boucle est ignorée et le reste est parcouru. Ceci permet d'obtenir une trajectoire continue et sans boucles.



Ce module trouve par exemple application dans le problème décrit sous le module *SMC_ToolCorr*.

Le module *SMC_AvoidLoop* fonctionne comme suit :

Le module parcourt tous les objets *SMC_GEOINFO* se trouvant dans la mémoire d'entrée *SMC_OUTQUEUE*. Si le bit7 de la variable *Intern_Mark* d'un de ces objets est activé, l'évitement de boucles est enclenché. Le module vérifie à partir de cet instant s'il y a un point de recoupement de l'objet actuel avec les objets *SMC_GEOINFO* suivants et se trouvant avant un objet *SMC_GEOINFO* dont le bit6 de la variable *Intern_Mark* est activé, désactivant ainsi l'évitement de boucles. Si ce n'est pas le cas, l'objet reste inchangé et est copié dans la sortie *SMC_OUTQUEUE* ; autrement, le premier des deux objets se recoupant est interrompu au point de recoupement, les objets *SMC_GEOINFO* se trouvant entre les objets recoupés sont rejetés, et la nouvelle trajectoire est continuée avec le second objet recoupé du point de recoupement. *SMC_NCDecoder* règle les bits 6 et 7 de *Intern_Mark* en réaction aux commandes G61/G60.

Le nombre d'objets se trouvant dans l'entrée *SMC_OUTQUEUE* est ici décisif, et donc la taille de l'entrée *SMC_OUTQUEUE*. Une entrée *SMC_OUTQUEUE* de petite taille fait qu'une boucle qui comprend plus d'objets que la mémoire *SMC_OUTQUEUE* ne peut pas être détectée.

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame la correction du rayon d'outil dès que cette entrée présente un front montant.

bAppend : BOOL

Si cette entrée est FALSE, la queue de sortie *DataOut* est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue *DataOut*.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel *SMC_OUTQUEUE* contenant les objets *SMC_GEOINFO* de la trajectoire ; cet objet se trouve normalement sur la sortie *DataOut* du module précédent (p.ex. du *SMC_NCDecoders*). Il faut dès lors la dimensionner suffisamment grande (voir ci-dessus).

nSizeOutQueue : UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structurels *GEOINFO* arrondis est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure *GEOINFO*. Si tel n'est pas le cas, le module *SMC_AvoidLoop* ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation. Il est recommandé de créer la mémoire tampon effective comme décrit ci-dessous p.ex. dans `Exemple: Array of SMC_GeoInfo`. La taille appropriée de la mémoire tampon résulte de `sizeof(Exemple)`.

pbyBufferOutQueue : POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure *OUTQUEUE*. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par *nSize_SMC_OUTQUEUE*. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un tableau *SMC_GeoInfo* (p.ex. `BUF : ARRAY[1..50] OF SMC_GEOINFO`; pour une mémoire tampon qui peut contenir 50 éléments de trajectoire). Cette valeur

peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Sorties du module :

bDone : BOOL

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Si une erreur survient, cette valeur devient TRUE.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui gère les objets SMC_GEOINFO de la trajectoire sans boucle.

iStatus : SMC_AL_STATUS (INT)

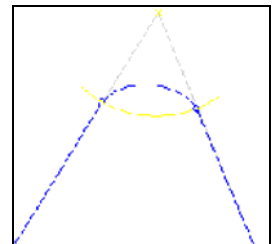
Cette variable d'énumération indique le statut actuel du module. Voici les statuts possibles :

AL_OFF	0	Évitement de boucle désactivé
AL_ON	1	Évitement de boucle activé
AL_END	2	Exécution des objets terminée.

6.2.5 SMC_SmoothPath

Le module *SMC_SmoothPath* est une partie possible de la préparation de trajectoire. Il arrondit les coins de la trajectoire afin d'obtenir une trajectoire sans à-coups (ponçage de coin). Le but est ici d'éviter des coins pour lesquels la vitesse doit de nature être ramenée à une valeur 0, cela si la précision du parcours joue un rôle moins important que la vitesse.

Pour ce faire, la trajectoire est coupée à une distance définie du coin et une **spline est insérée**. Le module calcule la grandeur de cette distance à partir d'une part de l'objet structurel SMC_GEOINFO du premier des objets à poncer et d'autre part d'une de ses entrées. Ces deux valeurs sont ajoutées et fournissent le rayon du cercle (centre dans le coin) qui présente une intersection avec les objets environnants.



Le module *SMC_SmoothPath* fonctionne comme suit :

Tous les objets SMC_GEOINFO se trouvant dans l'entrée de structure OUTQUEUE sont parcourus les uns après les autres. Si le bit4 de la variable Intern_Mark est activé dans un de ces objets, on commence en cet endroit à poncer les coins jusqu'à ce que le bit3 de la variable Intern_Mark d'un des objets suivants soit activé. Le module SMC_NCDecoder active ces bits en réaction aux commandes G51/G50. En d'autres termes, **on ponce les coins de tous les objets qui se situent au sein des commandes G51 et G50.**

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame la correction du rayon d'outil dès que cette entrée présente un front montant.

bAppend : BOOL

Si cette entrée est FALSE, la queue de sortie DataOut est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue DataOut.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire non arrondie ; cet objet se trouve normalement sur la sortie DataOut du module précédent (p.ex. du SMC_NCDecoders).

dEdgeDistance : LREAL

Cette variable d'entrée reprend la valeur qui, ajoutée à la valeur ToolRadius de l'objet SMC_GEOINFO, indique la distance par rapport à un coin, il s'agit de la valeur à partir de laquelle l'objet doit être recoupé et remplacé par une spline (voir ci-dessus). Cette valeur peut être modifiée en ligne. Ainsi, on dispose de la possibilité de saisir cette valeur hors ligne (via la structure SMC_GEOINFO) et de la moduler en ligne. Valeur par défaut : 0.

dAngleTol : REAL

Cette entrée décrit la valeur de tolérance d'angle jusqu'à laquelle un coude de la trajectoire ne doit pas être poncé.

nSizeOutQueue : UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structurels GEOINFO arrondis est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure SMC_GEOINFO, donc environ 2 KO. Si tel n'est pas le cas, le module SMC_SmoothPath ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

poqBufferOutQueue : POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure SMC_OUTQUEUE. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par nSizeOutQueue. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un array d'octet (p.ex. BUF: ARRAY[1..10000] OF BYTE; pour un espace mémoire d'une taille de 10000 octets). Cette valeur peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Sorties du module :

bDone : BOOL

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Si une erreur survient, cette valeur devient TRUE.

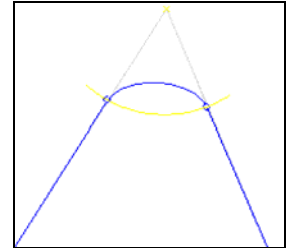
poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui gère les objets SMC_GEOINFO arrondis.

6.2.6 SMC_RoundPath

Le module *SMC_RoundPath* présente de nombreuses similitudes avec le module *SMC_SmoothPath*. Grâce à des **arcs de cercle**, il *arrondit les coins* générés lors du passage d'une ligne droite à l'autre.

Pour ce faire, la trajectoire est coupée à une distance *r* du coin et une **spline est insérée**. Le module calcule la grandeur de cette distance à partir d'une part de l'objet structurel *SMC_GEOINFO* du premier des objets à arrondir et d'autre part de l'entrée *dRadius*. Cette dernière est prépondérante, ce qui signifie que si sa valeur est 0, celle de l'objet est prise en compte. Si la valeur réglée est supérieure à la moitié de la longueur d'un des deux objets *SMC_GEOINFO*, cette demi longueur est utilisée.



Le module *SMC_RoundPath* fonctionne comme suit :

Tous les objets *SMC_GEOINFO* se trouvant dans l'entrée *SMC_OUTQUEUE* sont parcourus les uns après les autres. Si le bit5 de la variable *Intern_Mark* est activé dans un de ces objets, on commence en cet endroit à arrondir les coins jusqu'à ce que le bit3 de la variable *Intern_Mark* d'un des objets suivants soit activé. Le module *SMC_NCDecoder* active ces bits en réaction aux commandes *G50* et *G52* : **on arrondit les coins de tous les objets qui se situent au sein des commandes G50 et G52.**

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame la correction du rayon d'outil dès que cette entrée présente un front montant.

bAppend : BOOL

Si cette entrée est FALSE, la queue de sortie *DataOut* est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue *DataOut*.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel *SMC_OUTQUEUE* contenant les objets *SMC_GEOINFO* de la trajectoire non arrondie ; cet objet se trouve normalement sur la sortie *DataOut* du module précédent (p.ex. du *SMC_NCDecoders*).

dRadius : LREAL

Cette variable d'entrée reprend la valeur qui indique la distance par rapport à un coin, valeur à partir de laquelle l'objet doit être recoupé et remplacé par un arc de cercle (voir ci-dessus). Cette valeur peut être modifiée en ligne. Ainsi, on a la possibilité d'adapter cette valeurs aux caractéristiques de la trajectoire. Valeur par défaut : 0.

dAngleTol : REAL

Cette entrée décrit la valeur de tolérance d'angle jusqu'à laquelle un coude de la trajectoire ne doit pas être poncé.

nSizeOutQueue : UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structurels *GEOINFO* arrondis est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure *SMC_GEOINFO*. Si tel n'est pas le cas, le module *SMC_SmoothPath* ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation. Il est recommandé de créer la mémoire tampon effective comme décrit ci-dessous p.ex.dans `Exemple: Array of SMC_GeoInfo`. La taille appropriée de la mémoire tampon résulte de `sizeof(Exemple)`.

pbyBufferOutQueue : POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure OUTQUEUE. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par Size_SMC_OUTQUEUE. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un tableau SMC_GeoInfo (p.ex. BUF : ARRAY[1..50] OF SMC_GEOINFO; pour une mémoire tampon qui peut contenir 50 éléments de trajectoire). Cette valeur peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Sorties du module :

bDone : BOOL

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Si une erreur survient, cette valeur devient TRUE.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui gère les objets SMC_GEOINFO arrondis.

6.2.7 SMC_CheckVelocities

Ce module contrôle les vitesses des segments individuels de la trajectoire. Il doit toujours être appelé directement par l'interpolateur pour le cas où OutQueue ne serait pas créée par l'Éditeur mais bien au sein du programme IEC (p.ex. via SMC_NCDecoder).

La tâche principale de cette fonction est de vérifier si la trajectoire ne présente pas des coudes et de ramener la vitesse à 0 en de tels endroits.

Entrées du module :

bExecute : BOOL

Le contrôle commence avec un front montant.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire ; cet objet se trouve normalement sur la sortie DataOut du module précédent (p.ex. du SMC_NCDecoders).

dAngleTol : REAL

Cette entrée décrit la valeur de tolérance d'angle jusqu'à laquelle aucun arrêt ne doit être effectué au niveau d'un coude de la trajectoire (voir 0).

Sorties du module :

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Cette sortie d'énumération décrit le cas échéant une erreur.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique l'objet structurel SMC_OUTQUEUE qui contient la trajectoire avec ses vitesses admissibles et qui doit être transmis directement à l'interpolateur.

6.2.8 SMC_LimitCircularVelocities

Ce module contrôle chaque élément de OutQueue et limite les vitesses de trajectoire des éléments circulaires en fonction de leur rayon.

Lorsque l'on effectue un déplacement à vitesse constante (v) sur un passage d'une ligne droite à une courbe de rayon r , l'accélération de trajectoire saute en termes de chiffres de la valeur 0 à la valeur $\left[\frac{v^2}{r} \right]$.

Afin de limiter ce saut d'accélération à la valeur Acc , la vitesse de l'arc de cercle ne peut pas dépasser lors de la transition la valeur $\sqrt{Acc \cdot r}$.

Ce module contrôle le passage entre deux éléments (de ligne droite à arc de cercle, d'arc de cercle à ligne droite et d'arc de cercle à arc de cercle) et adapte la vitesse en fin du premier élément de telle sorte que le saut d'accélération ne dépasse pas la valeur $dMaxAccJump$.

En outre, le module limite l'accélération de trajectoire dans des cercles à la valeur $dMaxAcc$, cela en réduisant la vitesse de trajectoire à une valeur appropriée pour le cercle.

Entrées du module :

bExecute : BOOL

Le module procède à une réinitialisation et entame la correction du rayon d'outil dès que cette entrée présente un front montant.

bAppend : BOOL

Si cette entrée est FALSE, la queue de sortie DataOut est vidée à chaque réinitialisation. Si elle est TRUE, les nouvelles données sont écrites à la fin de la queue DataOut.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structuré SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire à modifier ; cet objet se trouve normalement sur la sortie DataOut du module précédent (p.ex. du SMC_NCDcoders).

dMaxAcc : LREAL

Cette variable d'entrée contient la valeur maximale d'accélération admissible pour des arcs de cercle. Une valeur égale à 0 signifie que le contrôle n'est pas exécuté.

dMaxAccJump : LREAL

Cette variable d'entrée contient la valeur maximale de saut d'accélération (a) pour la transition entre deux objets. Une valeur égale à 0 signifie que le contrôle n'est pas exécuté.

nSizeOutQueue : UDINT

Le module se voit communiquer ici la taille de la mémoire tampon dans laquelle la liste des objets structuré GEOINFO arrondis est écrite. Ce module doit être au moins cinq fois plus grand qu'une structure SMC_GEOINFO. Si tel n'est pas le cas, le module SMC_SmoothPath ne peut effectuer aucune action. La valeur peut être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation. Il est recommandé de créer la mémoire tampon effective comme décrit ci-dessous p.ex. dans Exemple: Array of SMC_GeoInfo. La taille appropriée de la mémoire tampon résulte de `sizeof(Exemple)`.

pbyBufferOutQueue : POINTER TO BYTE

Cette entrée doit indiquer le premier octet de l'espace mémoire créé pour la structure OUTQUEUE. Cet espace mémoire doit être au moins aussi grand que la valeur indiquée par Size_SMC_OUTQUEUE. Normalement, la création de cet espace mémoire s'effectue dans la partie déclaration du programme IEC au moyen d'un tableau SMC_GeoInfo (p.ex. BUF : ARRAY[1..50] OF SMC_GEOINFO; pour une mémoire tampon qui peut contenir 50 éléments de trajectoire). Cette valeur peut également être modifiée, puis elle ne peut encore l'être ultérieurement que lors d'une réinitialisation.

Sorties du module :**bDone : BOOL**

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

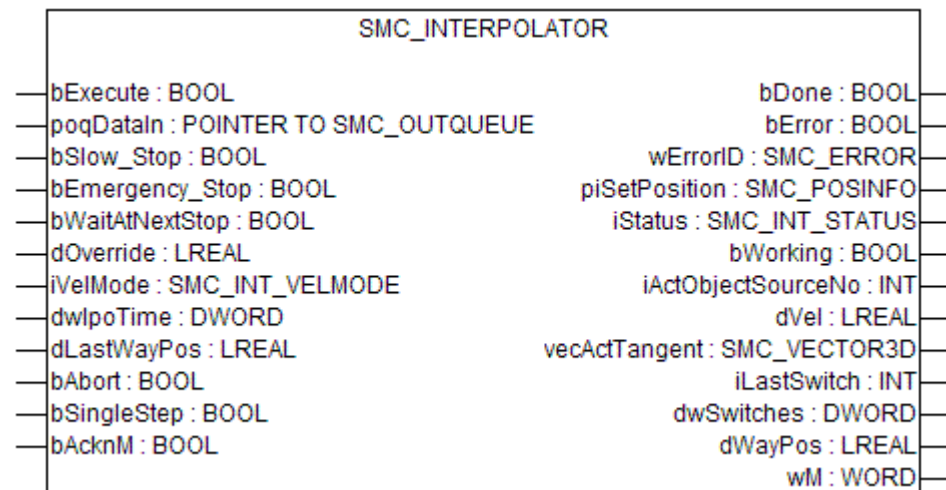
Si une erreur survient, cette valeur devient TRUE.

poqDataOut : POINTER TO SMC_OUTQUEUE

Cette sortie indique une structure SMC_OUTQUEUE qui gère les nouveaux objets SMC_GEOINFO.

6.2.9 SMC_Interpolator

Le module SMC_Interpolator a pour but de changer en **points de trajectoire discrets** une trajectoire continue présente, décrite par des objets GEOINFO, en tenant compte d'un profil de vitesse et d'un schéma temporel définis. Ces données relatives à la position sont normalement transformées par le programme IEC (p.ex. sur les positions d'axes d'entraînement) et envoyées aux entraînements par le biais de l'interface d'entraînement.

Entrées du module :**bExecute : BOOL**

Le module procède à une réinitialisation et entame l'interpolation dès que cette entrée présente un front montant.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structurel SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire qui doit être interpolée ; cet objet se trouve normalement sur la sortie poqDataOut du module précédent SMC_CheckVelocities.

bSlow_Stop : BOOL

Si cette entrée est sur FALSE (valeur par défaut), la trajectoire est parcourue sans interruption. Si l'entrée est sur TRUE, on oblige SMC_Interpolator à réduire la vitesse à 0, selon le profil de vitesse réglé (byVelMode, voir ci-dessous) et la décélération maximale de l'objet GEOINFO actuel (dDecel, voir ci-dessous) ; l'interpolateur doit ensuite attendre jusqu'à ce que Slow_Stop devienne à nouveau FALSE.

bEmergency_Stop : BOOL

Par défaut, cette entrée est sur FALSE. Si elle est TRUE, SMC_Interpolator procède à un arrêt immédiat, la position de réglage est ainsi maintenue. La vitesse est ainsi ramenée immédiatement à 0.

bWaitAtNextStop : BOOL

Si cette entrée est sur FALSE (valeur par défaut), la trajectoire est parcourue sans interruption. Si l'entrée est TRUE, on oblige SMC_Interpolator à maintenir la position de réglage au prochain arrêt naturel, à des points où la vitesse est de 0 – normalement aux coins de la trajectoire –, et à y rester jusqu'à ce que bWaitAtNextStop devienne à nouveau FALSE.

dOverride : LREAL

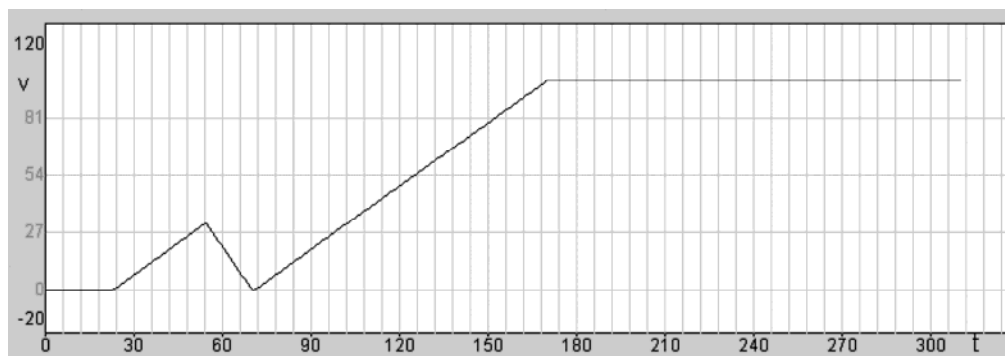
Cette variable permet de se servir de l'override. Les valeurs admissibles sont supérieures à 0,01. L'override est multiplié par la vitesse de consigne des objets individuels et permet ainsi d'augmenter ou de réduire la vitesse de consigne en ligne. Par exemple, un override de 1 (valeur par défaut) entraîne l'exécution des vitesses de consigne programmées alors qu'un override de 2 permet de doubler ces vitesses.

Même si l'override peut être modifié en tout temps, il faut noter qu'une modification ne sera reprise que si on n'est pas déjà dans une accélération ou une décélération.

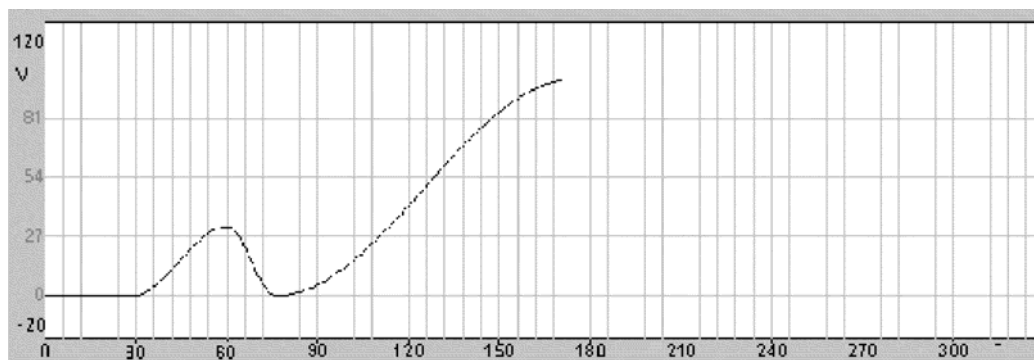
iVelMode : SMC_INT_VELMODE

Cette entrée permet de définir le profil de vitesse. La valeur „TRAPEZOID“ (valeur par défaut) entraîne un profil de vitesse trapézoïdal et la valeur „SIGMOID“ un profil en forme de S :

Exemple de profil de vitesse trapézoïdal (Vel_Mode = 0) :



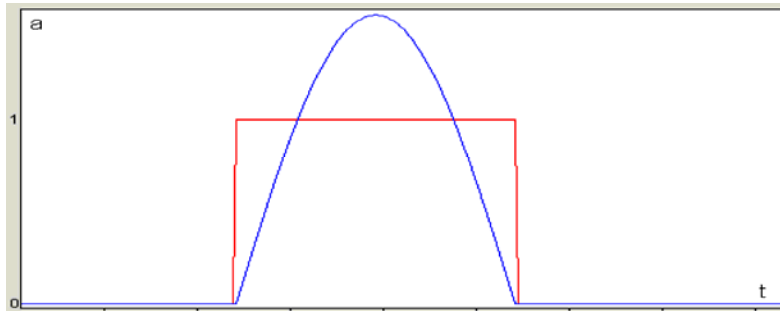
Exemple de profil de vitesse en forme de S (Vel_Mode = 1) :



Dans les exemples ci-dessus, l'accélération maximale (Accel) a été choisie comme étant inférieure à la décélération maximale (Decel). Ceci entraîne une progression différente de la vitesse selon qu'il s'agit d'une accélération ou d'une décélération.

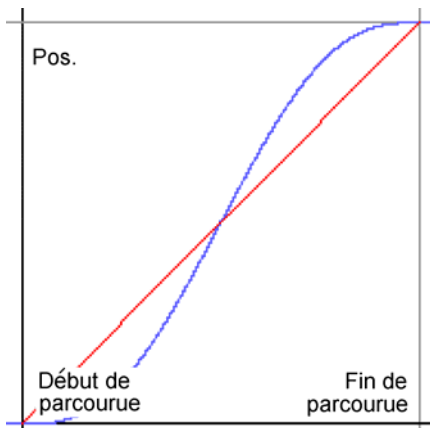
Le profil de vitesse en forme de S présente l'avantage selon lequel l'accélération qui s'y rapporte est constante – à l'inverse du profil trapézoïdal –, apportant ainsi un certain soulagement particulièrement pour des machines lourdes. Ceci est cependant compensé par un temps de calcul légèrement plus grand.

Comme le profil de vitesse en forme de S (bleu) est conçu de telle sorte qu'un passage à un profil trapézoïdal (rouge) n'entraîne aucun changement au niveau de la durée nécessaire au parcours de la trajectoire, la progression prudente de l'accélération au début et à la fin doit être compensée par une accélération plus élevée au centre. Il convient de noter ici que l'accélération et la décélération maximale programmée dans les objets GEO sont dépassées au maximum du facteur $p/2$:



Une modification en ligne – comme pour un *Override* – n'est reprise que quand une phase d'accélération ou de décélération est terminée.

Afin d'interpoler également les axes supplémentaires sous forme de S (bleu, voir esquisse) et non pas sous forme linéaire (rouge), les bits appropriés doivent être activés – indépendamment de l'entrée `byVelMode` de `SMC_Interpolators` – dans la variable `wSProfile` de `piStartPos` de l'objet actuel de trajectoire. Les axes supplémentaires ne sont alors plus interpolés de manière linéaire par rapport à la longueur de la trajectoire dans l'espace X,Y,Z, mais bien de manière polynomiale, de sorte à obtenir un profil en forme de S pour la position d'axe qui présente une vitesse et une accélération nulles en début et en fin d'un tronçon.



dwIpoTime : DWORD

Cette entrée qui doit être activée pour tout appel contient le temps de cycle en μsec .

dLastWayPos : LREAL

Grâce à cette entrée, l'utilisateur peut mesurer la distance parcourue par l'interpolateur. La sortie `dWayPos` est la somme de `dLastWayPos` et de la distance parcourue lors de ce cycle. Si `dLastWayPos=0`, `dWayPos` correspond à la longueur du tronçon effectif. Si `dLastWayPos` est identique à la sortie `dWayPos`, cette dernière est toujours incrémentée du tronçon effectif et on obtient ainsi le total de la distance parcourue. Cela dit, `dLastWayPos` peut toujours être remis à 0 ou sur une autre valeur.

bAbort : BOOL

Cette entrée interrompt l'exécution d'un contour.

bSingleStep : BOOL

Cette entrée provoque l'arrêt de l'interpolateur le temps d'un cycle à chaque transition entre des objets de trajectoire (même aux transitions présentant la même tangente). Si `bSingleStep` est réglé sur `TRUE` lors du processus, l'interpolateur s'arrête à la fin de l'objet suivant, objet qu'il peut atteindre sans dépasser la décélération définie.

Sorties du module :**bDone : BOOL**

Cette variable devient TRUE lorsque les données d'entrée de *DataIn* sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée *bExecute* est FALSE, *bDone* devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_INT_ERROR (INT)

Cette sortie d'énumération décrit le cas échéant une erreur survenue lors de l'interpolation. Suite à une erreur, l'exécution est interrompue jusqu'à une réinitialisation.

piSetPosition : SMC_POSINFO

Set_Position indique la position nominale calculée selon la définition. *Set_Position* présente une structure *SMC_POSINFO* et contient non seulement les coordonnées cartésiennes de point cible de la trajectoire, mais également la position des autres axes.

iStatus : SMC_INT_STATUS (INT)

Cette variable d'énumération indique le statut actuel du module. Voici les statuts possibles :

IPO_UNKNOWN	0	Statut interne. Ce statut ne peut pas survenir suite à l'exécution complète de <i>SMC_Interpolator</i> .
IPO_INIT	1	Statut d'initialisation ; <i>DataIn</i> n'est et n'était pas encore plein.
IPO_ACCEL	2	Le module se trouve en phase d'accélération.
IPO_CONSTANT	3	Le module se déplace à vitesse constante.
IPO_DECEL	4	Le module se trouve en phase de freinage.
IPO_FINISHED	5	L'exécution de la liste <i>GEOINFO</i> est terminée. Les objets <i>GEOINFO</i> survenant par après dans <i>DataIn</i> ne sont plus exécutés.
IPO_WAIT	6	Le module attend car un des cas de figure ci-dessous est survenu : <i>Emergency_Stop</i> = TRUE <i>Slow_Stop</i> = TRUE et <i>Vel</i> = 0 <i>Wait_At_Next_Stop</i> = TRUE et <i>Vel</i> = 0

bWorking : BOOL

La sortie devient TRUE si l'exécution de la liste est commencée mais pas encore terminée (IPO_ACCEL ou IPO_CONSTANT ou IPO_DECEL ou IPO_WAIT). Dans les autres cas, *Working* est sur FALSE.

iActObjectSourceNo: INT

Cette variable reprend l'entrée *SourceLine_Nr* de l'objet *GEOINFO* de *DataIn-Queue* actuellement parcouru. Si *SMC_Interpolator* ne fonctionne pas (plus) (*Working* = FALSE), cette variable affiche -1.

dVel : LREAL

Cette variable reprend la vitesse réelle résultant du déplacement d'un objet des coordonnées spatiales précédentes vers *Set_Position*, dans le temps *Ipo_Time*.

vecActTangent : SMC_VECTOR3D

Cette structure reprend le sens de la trajectoire au point *Set_Position*. Si *Vel* = 0, *Act_Tangent* affiche également des zéros.

iLastSwitch : INT

Cette variable reprend le numéro du point de commutation parcouru en dernier. Si plusieurs points de commutation sont traversés lors d'un cycle, celui parcouru en dernier lieu est toujours repris.

dwSwitches : DWORD

Cette variable DWORD contient le statut actuel de commutation de tous les points de commutation entre 1 et 32. À l'inverse de iLastHelpMark, on peut ainsi exclure toute perte de point de commutation.

dWayPos : LREAL

Pour une description, voir l'entrée dLastWayPos.

À chaque appel, SMC_Interpolator calcule et édite le point de trajectoire suivant en tenant compte des paramètres prédéfinis, de l'historique de la vitesse et de la dernière position de trajectoire. Dès que le premier objet GEOINFO est exécuté, il est supprimé de la structure poqDataIn-SMC_OUTQUEUE.

wM : WORD

Si l'interpolateur traverse un objet M (une ligne qui décrit une fonction auxiliaire), cette sortie est réglée sur la valeur appropriée et attend son acquittement via l'entrée bAcknM.

Il convient de noter qu'à la fin d'un contour, la variable SMC_OUTQUEUE est vide. Si on veut à nouveau effectuer un déplacement selon le même contour, il faut soit parcourir à nouveau le programme CNC via décodeur et modules de préparation de trajectoire au sein d'une structure SMC_OUTQUEUE, soit appliquer la fonction SMC_RESTOREQUEUE (voir 0). Cette dernière variante n'est possible que si la mémoire tampon OUTQUEUE a été sélectionnée suffisamment grande que pour pouvoir accueillir le contour complet.

6.2.10 SMC_GetMParameters

Ce module de la bibliothèque SM_CNC.lib permet de scruter des paramètres qui ont été fournis avec la fonction M (K, L, O, voir 3.2.2) alors que l'interpolateur se trouve sur une fonction M.

Entrées du module :

bEnable : BOOL

Le module est actif si cette entrée est réglée.

Entrées / sorties (VAR IN OUT) du module :

Interpolator : SMC_Interpolator

Interpolator-Instant

Sorties du module :

bMActive : BOOL (valeur par défaut : FALSE)

TRUE si une fonction M est actuellement présente.

dK, dL : LREAL (valeur par défaut : 0)

Paramètres M qui ont été définis par les mots K et L

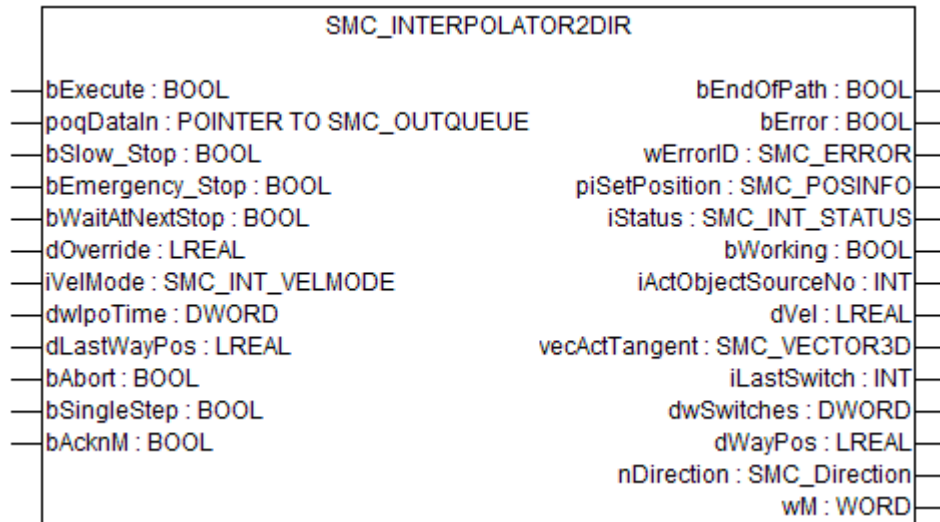
MParameters : SMC_M_PARAMETERS

Paramètres M qui ont été définis par des structures de données globales gSMC_MParameters ou par la variable transmise par O.

6.2.11 SMC_Interpolator2Dir

Dans sa fonction et l'affectation de ses entrées et sorties, ce module de la bibliothèque SM_CNC.lib correspond au module fonctionnel SMC_Interpolator, à cette différence près qu'il peut également interpoler une trajectoire en arrière.

Pour ce faire, l'entrée dOverride est affectée d'une valeur négative, suite à quoi SMC_Interpolator2Dir procède à l'interpolation en sens négatif. Cette entrée peut p.ex. accueillir l'entrée analogique de vitesse d'un bouton de réglage, de sorte que l'utilisateur puisse procéder à un déplacement en avant ou en arrière à son gré.



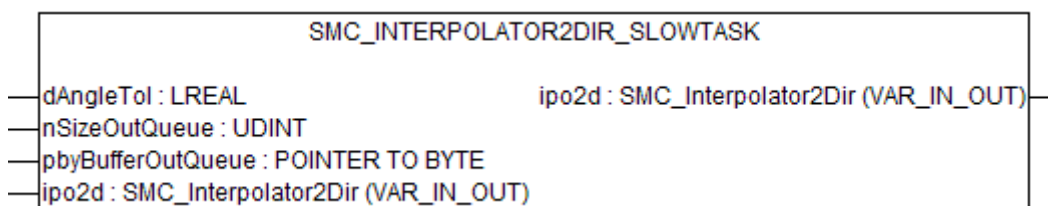
Entrées et sorties supplémentaires du module :

nDirection : SMC_Direction

Le module indique dans quel sens il se déplace actuellement. Les valeurs possibles sont IPO_positive, IPO_negative et IPO_standstill.

Les conditions suivantes doivent être réunies pour l'utilisation de ce module :

1. La trajectoire doit être entièrement comprise dans poqDataIn. Comme le module alterne les déplacements en avant et en arrière, la trajectoire complète doit être comprise dans la mémoire.
2. Un module supplémentaire, SMC_Interpolator2Dir_SlowTask, est appelé :



Ce module est responsable de la création du chemin en marche arrière. Il a été scindé par SMC_Interpolator2Dir afin qu'il puisse être déplacé dans une tâche moins prioritaire sur des systèmes cible fortement chargés et moins performants.

Entrées du module :

dAngleTol : LREAL

Tolérance d'angle pour le chemin en marche arrière. Valeur normalement identique à la tolérance d'angle du chemin original.

nSizeOutQueue : UDINT, pbyBufferOutQueue : POINTER TO BYTE

Taille et pointeur de mémoire tampon des données dans laquelle la trajectoire en arrière doit être enregistrée. Doit être suffisamment grande que pour accueillir la trajectoire complète.

lpo2d : SMC_Interpolator2Dir

Instance SMC_Interpolator2Dir pour laquelle le chemin en marche arrière doit être créé.

6.3 Fonctions et blocs fonctionnels auxiliaires

Les blocs fonctionnels SMC_RotateQueue2D et SMC_TranslateQueue3D contenus dans la bibliothèque SM_CNC.lib pivotent ou décalent la trajectoire enregistrée dans une mémoire SMC_OutQueue.

La variable d'entrée **DataIn** permet de transmettre la trajectoire à pivoter ou décaler à l'objet structurel SMC_OUTQUEUE, cela sous la forme d'un pointeur.

L'entrée **bEnable**, initialisée avec FALSE, empêche la rotation et la translation de la trajectoire jusqu'à ce qu'elle devienne TRUE. Tous les éléments GEOINFO sont alors traités par *poqDataIn*. Dès que **bEnable** devient FALSE, les modules ne procèdent plus à aucun changement.

L'entrée **bReset**, également initialisée avec FALSE, fait que ce ne sont pas les objets GEOINFO actuellement présents dans *poqDataIn* qui sont pivotés ou décalés mais bien ceux qui y rentrent à partir de maintenant.

- **SMC_ROTATEQUEUE2D**

La trajectoire contenue dans *poqDataIn* est pivotée autour de l'axe Z selon l'angle en ° transmis dans *dPhi*. Un angle positif entraîne une rotation dans le sens positif au sens mathématique (sens antihoraire).

- **SMC_TRANSLATEQUEUE3D**

La trajectoire contenue dans *poqDataIn* est décalée selon le vecteur transmis *vec* par le structure SMC_VECTOR3D (voir SMC_VECTOR3D).

- **SMC_SCALEQUEUE3D**

La trajectoire contenue dans *poqDataIn* est étendue du facteur *fScaleFactor*.

Comme une modification soudaine des grandeurs caractéristiques de rotation / translation (*dPhi*, *vec*) lors de l'exécution peuvent entraîner un saut au sein de la trajectoire, les modifications dans les entrées de ces grandeurs sont ignorées jusqu'à ce que la SMC_OUTQUEUE de *poqDataIn* soit vide ou qu'un **bReset** soit effectué.

Afin d'obtenir une rotation dans le niveau (XY) autour d'un autre point que (00), c.-à-d. autour d'un point (XpYp), on utilise successivement une translation selon le vecteur (-Xp-Yp°0), la rotation selon l'angle souhaité *dPhi* suivie d'une autre translation selon le vecteur (XpYp°0).

6.4 Paramétrage via variables globales

Des variables internes et constantes sont définies sous „SoftMotion_CNC_Globals“ (voir onglet Ressources en dessous de l'entrée de la bibliothèque SM_CNC.lib). Certaines de ces variables et constantes peuvent être modifiées :

Contrôle de point zéro (voir 3.6, Changer les valeurs epsilon pour zéro) :

- **g_fSMC_CNC_EPS** (valeur epsilon pour contrôle précis de point zéro)
- **g_fSMC_CNC_EPS_RELUCTANT** (valeur epsilon pour contrôle de tolérance de point zéro)

6.5 Structures de SM_CNC.lib

Voici une sélection des structures de SM_CNC.lib qui sont utilisées par les modules de SM_CNC.lib pour enregistrer des positions, sections de trajectoire (objets de trajectoire) et des vecteurs :

- SMC_POSINFO
- SMC_GEOINFO
- SMC_VECTOR3D
- SMC_VECTOR6D

La structure SMC_OUTQUEUE aide à gérer les objets GEOINFO au sein d'une liste de taille définie.

SMC_POSINFO

La structure **SMC_POSINFO** contenue dans la bibliothèque SM_CNC.lib permet l'enregistrement des coordonnées et de la position des axes supplémentaires.

```

TYPE SMC_POSINFO:
STRUCT
    iFrame_Nr:INT;
    wAuxData:WORD;
    wSProfile:WORD;
    dX:LREAL;
    dY:LREAL;
    dZ:LREAL;
    dA:LREAL;
    dB:LREAL;
    dC:LREAL;
    dA1:LREAL;
    dA2:LREAL;
    dA3:LREAL;
    dA4:LREAL;
    dA5:LREAL;
    dA6:LREAL;
END_STRUCT
END_TYPE

```

Les variables dX, dY et dZ contiennent la position dans l'espace, dA1, ..., dA6 la position des axes supplémentaires. iFrame_Nr donne à l'utilisateur la possibilité de créer des informations qui ne sont pas absolument nécessaires aux modules SoftMotion. dA, dB et dC ne sont momentanément pas utilisés.

wAuxData indique au moyen de bit les axes de positionnement qui doivent être calculés par SMC_Interpolator. wAuxData est initialisé avec 2#111, ce qui signifie que les axes X, Y et Z sont interpolés. Si le premier bit est activé, la position dX est calculée, le bit 7 entraîne par exemple le traitement de dA2.

De même, wSProfile définit pour les axes supplémentaires (tous sauf les axes X,Y) s'ils doivent être interpolés de manière linéaire (FALSE) ou en forme de S (TRUE) par l'interpolateur. Le bit2 correspond à l'axe Z, le bit6 au P, le bit7 au Q, le bit8 au U, le bit9 au V et le bit10 au W.

SMC_GEOINFO

La structure **SMC_GEOINFO** comprise dans la bibliothèque SM_CNC.lib permet d'enregistrer un objet de trajectoire. Par objet de trajectoire, on entend une partie de la trajectoire programmée qui peut être entièrement mémorisée dans la structure suivante en raison de sa forme géométrique.

```

TYPE SMC_GEOINFO:
STRUCT
    iObjNo:INT;
    iSourceLineNo:INT;
    iMoveType:MOVTYPE;
    piStartPos:SMC_POSINFO;
    piDestPos:SMC_POSINFO;
    dP1:LREAL;
    dP2:LREAL;
    dP3:LREAL;
    dP4:LREAL;
    dP5:LREAL;
    dP6:LREAL;
    dP7:LREAL;
    dP8:LREAL;
    dT1:LREAL;
    dT1:LREAL;
    dToolRadius:LREAL;
    dVel:LREAL;
    dVelEnd:LREAL;
    dAccel:LREAL;
    dDecel:LREAL;
    dLength:LREAL;
    byInternMark:BYTE;
    dHelpPos : ARRAY[0..MAX_IPOSWITCHES] OF LREAL;
    iHelpID : ARRAY[0..MAX_IPOSWITCHES] OF INT;
END_STRUCT
END_TYPE

```

iObjNo : INT

Cette valeur de nombre entier permet d'enregistrer un numéro d'objet au gré. Elle ne joue aucun rôle pour la description de la trajectoire.

iSourceLineNo : INT

Cette valeur de nombre entier reprend normalement le code source du numéro de ligne du programme CNC. Elle ne joue aucun rôle pour la description de la trajectoire.

iMoveType : MOVTYPE (INT)

Le type d'énumération MOVTYPE contient les valeurs admissibles ci-dessous et décrit le type d'objet :

LIN	1	Mouvement rectiligne (G01)
CLW	2	Rotation dans le sens horaire (G02)
CCLW	3	Rotation dans le sens antihoraire (G03)
SPLINE	5	Spline, parabole (G05, G06)
ELLCLW	8	Ellipse dans le sens horaire (G08)
ELLCCLW	9	Ellipse dans le sens antihoraire (G09)

LINPOS	100	Positionnement rectiligne (G00)
INITPOS	110	Positionnement aveugle (le point de départ ,est encore inconnu ; il est constamment complété par SMC_Interpolator)
MCOMMAND	120	Fonction auxiliaire, fonction M

piStartPos : SMC_POSINFO

Structure SMC_POSINFO qui décrit la position exacte de départ. (est ignorée si Move_Type = INITPOS).

piDestPos : SMC_POSINFO

Structure SMC_POSINFO qui décrit la position exacte de fin.

dP1, ..., dP8 : LREAL

Cette variable permet d'enregistrer des informations supplémentaires sur la trajectoire, quel que soit le Move_Type (voir ci-dessus) :

LIN LINPOS	Aucune signification car les informations complètes sont déjà contenues dans Start_Pos et Dest_Pos.
CLW CCLW	P1: coordonnées X du centre du cercle P2: coordonnées Y du centre du cercle P3: rayon de cercle
SPLINE	Paramètres de spline
ELLCLW, ELLCCLW	P1: coordonnées X du centre du cercle P2: coordonnées Y du centre du cercle P3: composants X du vecteur d'axe principal 1 P4: composants Y du vecteur d'axe principal 1 P5: longueur de l'axe principal P6: longueur de l'axe secondaire P7: direction de l'axe principal (°) P8: rapport P6/P5
INITPOS	Aucune signification

dT1, dT2 : LREAL

Cette variable reprend le début et la fin du paramètre de déplacement. Signification en fonction du Move_Type :

LIN LINPOS	Aucune signification car les informations complètes sont déjà contenues dans Start_Pos et Dest_Pos.
CLW	T1: angle de démarrage dans le sens mathématique, en degrés : (0 = est, 90 = nord, 180 = ouest, 360 = sud) T2: angle d'ouverture du cercle, longueur d'arc de cercle, en degrés : (par exemple : 90 = quart de cercle, 180 = demi-cercle)
CCLW	T1: angle de démarrage dans le sens mathématique, en degrés : (0 = est, 90 = nord, 180 = ouest, 360 = sud) T2: angle négatif d'ouverture du cercle : (par exemple : -90 = quart de cercle, -180 = demi-cercle)
SPLINE	Valeur de départ et de fin du paramètre t (voir description de spline). Valeurs standard 0 et 1.
INITPOS	Aucune signification.

dToolRadius : LREAL

Cette variable permet l'enregistrement d'informations nécessaires à la préparation de la trajectoire (voir les modules SMC_ToolCorr, SMC_RoundPath). L'entrée ne joue aucun rôle si des modules de préparation de trajectoire inappropriés (SMC_ToolCorr, SMC_RoundPath) sont parcourus.

dVel, dVelEnd : LREAL

Ces deux variables doivent toujours être affectées et contiennent des informations pour le profil de vitesse au sein de cet objet. dVel décrit la vitesse de consigne qui doit être atteinte, dVelEnd la vitesse de déplacement à la fin de l'objet (voir le module SMC_Interpolator) en unités de trajectoire/sec.

dAccel, dDecel : LREAL

dAccel et dDecel permettent l'enregistrement de l'accélération et de la décélération maximales admissibles en unités de trajectoire/sec². Les deux variables sont préalablement affectées de la valeur 100.

dLength : LREAL

Cette variable qui doit impérativement être affectée contient la longueur de l'objet en unités de trajectoire.

byIntern_Mark : BYTE

Le début et la fin de la préparation de trajectoire sont enregistrés ici comme suit :

Bit 0 affecté	Fin de la correction du rayon d'outil après cet objet
Bit 1 affecté	Début de la correction du rayon d'outil pour cet objet
Bit 2 affecté	Début de la correction du rayon d'outil à droite pour cet objet
Bit 3 affecté	Fin d'arrondi / de lissage de trajectoire après cet objet
Bit 4 affecté	Début de lissage de trajectoire pour cet objet
Bit 4 affecté	Début d'arrondi de trajectoire pour cet objet
Bit 6 affecté	Fin de l'évitement de boucles après cet objet
Bit 7 affecté	Début de l'évitement de boucles pour cet objet

dHelpPos : ARRAY[0..MAX_IPOSWITCHES] OF LREAL,**iHelpID : ARRAY[0..MAX_IPOSWITCHES] OF INT:**

Ces variables reprennent la position relative (0: début d'objet, 1:fin d'objet ; comme code G: O) et l'ID (voir Code G : H) de l'interrupteur auxiliaire.

S'il s'agit pour l'objet actuel d'une MCOMMAND, iHelpID[0] contient le numéro de la fonction M.

SMC_VECTOR3D

La structure **SMC_VECTOR3D3** comprise dans la bibliothèque **SM_CNC.lib** permet d'enregistrer un vecteur tridimensionnel.

```

TYPE SMC_VECTOR3D:
STRUCT
  dX:LREAL;
  dY:LREAL;
  dZ:LREAL;
END_STRUCT
END_TYPE

```


SMC_VECTOR6D

La structure **SMC_VECTOR6D** comprise dans la bibliothèque SM_CNC.lib permet d'enregistrer un vecteur à 6 dimensions.

```
TYPE SMC_VECTOR6D:
STRUCT
  dX:LREAL;
  dY:LREAL;
  dZ:LREAL;
  dA:LREAL;
  dB:LREAL;
  dC:LREAL;
END_STRUCT
END_TYPE
```

Structure SMC_OUTQUEUE et fonctions

La structure **SMC_OUTQUEUE** contenue dans la bibliothèque SM_CNC.lib aide à gérer les objets *GEOINFO* au sein d'une liste de taille définie.

```
TYPE SMC_OUTQUEUE :
STRUCT
  wOUTQUEUEStructID : WORD;
  pbyBuffer: POINTER TO BYTE;
  nSize : UDINT;
  nReadPos : UDINT;
  nWritePos : UDINT;
  bFull : BOOL;
  bEndOfList: BOOL;
  byGenerator : BYTE;
END_STRUCT
END_TYPE
```

Grâce à la variable `wOUTQUEUEStructID` qui contient une valeur fixe, les modules vérifient en interne si la variable structurelle transmise est de type `SMC_OutQueue`.

La variable `byGenerator` écrit le créateur de la Queue. Ces informations permettent à l'interpolateur de vérifier si le module `SMC_CheckVelocities` est parcouru en dernier comme cela est prescrit. Les valeurs suivantes sont définies.

La variable `byGenerator` écrit le créateur de la Queue. Ces informations permettent à l'interpolateur de vérifier si le module `SMC_CheckVelocities` est parcouru en dernier comme cela est prescrit. Les valeurs suivantes sont définies.

Créateur	Valeur
SMC_NCDecoder	1
SMC_AvoidLoop	10
SMC_LimitCircularVelocity	11
SMC_RoundPath	12
SMC_SmoothPath	13
SMC_ToolCorr	14
SMC_RotateQueue2D	30
SMC_ScaleQueue3D	31

Créateur	Valeur
SMC_TranslateQueue3D	32
SMC_CheckVelocities	254
Éditeur CNC	255

La bibliothèque SoftMotion propose les fonctions suivantes pour la manipulation d'un objet structurel SMC_OUTQUEUE :

BOOL **SMC_RESTOREQUEUE**(Enable: BOOL, POQ: POINTER TO SMC_OUTQUEUE)

Cette fonction restaure une structure déjà interpolée ou exécutée. Ceci n'est possible que si la liste peut déjà contenir la trajectoire complète dans POQ.

POINTER TO BOOL **SMC_APPENDOBJ**(POQ: POINTER TO SMC_OUTQUEUE, PGI: POINTER TO SMC_GEOINFO)

Cette fonction place dans *POQ* en fin de liste l'objet GEOINFO qui est dans *PGI*, pour autant que *OQ* soit correctement initialisé et ne soit pas encore complètement affecté. Si tel est le cas, la valeur de retour est l'apparition d'un pointeur en regard du nouvel élément de la liste ; dans le cas contraire, 0 apparaît.

BOOL **SMC_DELETEOBJ**(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Cette fonction supprime le *Ne* objet de la liste à *POQ*, le comptage commence à 0. Si *N-1* est supérieur au nombre d'objets GEOINFO enregistrés, il ne se passe rien et la valeur de retour est FALSE ; elle est TRUE dans les autres cas.

UINT **SMC_GETCOUNT**(POQ: POINTER TO SMC_OUTQUEUE)

La valeur de retour de cette fonction est composée du nombre d'objet enregistrés dans *POQ*.

POINTER TO **SMC_GEOINFO GETOBJ**(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Cette fonction fournit – pour autant que *POQ* soit correctement initialisé et qu'il existe un *Ne* objet – un pointeur sur le *Ne* objet *GEOINFO* de la liste chez *POQ*.

POINTER TO **SMC_GEOINFO GETOBJFROMEND**(POQ: POINTER TO SMC_OUTQUEUE, N: UINT)

Cette fonction fournit – pour autant que *POQ* soit correctement initialisé et qu'il existe au moins *N+1* éléments – un pointeur sur le *Ne* objet GEOINFO partant de la fin de la liste chez *POQ* ; si *N=0*, le dernier élément de liste est restitué.

Initialisation de structure :

Les modules SoftMotion *SMC_NCDecoder*, *SMC_SmoothPath*, *SMC_RoundPath*, *SMC_AvoidLoop* et *SMC_ToolCorr* éditent un pointeur sur une structure OutQueue gérée en interne et procèdent eux-mêmes à l'initialisation de la structure. Les modules *SMC_SmoothPath*, *SMC_RoundPath*, *SMC_ToolCorr*, *SMC_AvoidLoop* et *SMC_Interpolator* attendent comme entrée un pointeur sur la liste OutQueue correcte. Si cette liste est programmée et complétée „manuellement“, il faut procéder soi-même à une initialisation correcte. Pour ce faire, les deux premières variables (tampon, taille) doivent être réglées. Par ailleurs, il est fortement recommandé d'utiliser les fonctions ci-dessus lors de travaux avec une SMC_OUTQUEUE et d'éviter au possible – suite à une initialisation – toute modification des autres variables.

Composants de la structure :

pbyBuffer : POINTER TO BYTE

Ceci reprend l'adresse d'un espace mémoire continu prévu pour l'enregistrement des objets GEOINFO. Cette mémoire doit être affectée dans la programme IEC et son adresse doit être écrite dans cette variable. L'affectation dans la partie déclaration peut p.ex. se faire au moyen d'un array d'octet (BUF : ARRAY[1..10000] OF BYTE; pour un espace mémoire d'une taille de 10000 octets).

nSize : UDINT

Cette variable reprend la taille de l'espace mémoire réservé pour *Buffer*.

nReadPos : UDINT

Cette variable reprend l'adresse du premier élément de liste par rapport au premier octet de l'espace mémoire réservé.

nWritePos : UDINT

Cette variable reprend l'adresse du premier octet libre après la liste par rapport au premier octet de l'espace mémoire réservé.

bFull : BOOL

Si la liste atteint la capacité pour trois autres structures GEOINFO (mémoire tampon de sécurité), la fonction APPENDOBJ règle cette variable sur TRUE. DELETEOBJ la règle à nouveau sur FALSE dès que des éléments de la liste sont supprimés.

bEndOfList : BOOL

Les modules SoftMotion qui ont comme entrée une structure OutQueue attendent toujours que celle-ci soit pleine avant de procéder à son exécution, afin p.ex. d'éviter une sous-exécution de données (SMC_Interpolator). Comme dans le cas des derniers objets SMC_GEOINFO d'une trajectoire, la mémoire OutQueue n'est plus pleine, cette variable doit être TRUE après le placement du dernier objet SMC_GEOINFO, ceci afin que l'on puisse continuer le traitement. Si la liste est vide et doit par après être à nouveau remplie, EndOfList doit redevenir FALSE.

SMC_CNC_REF

Cette structure de données permet de gérer des fichiers G-Code parsés :

```

TYPE SMC_CNC_REF :
STRUCT
    wCNCREFStructID: WORD := 16#BA56;
    nelements: UDINT;
    diReadPos: UDINT := 0;
    udiBuffer: UDINT := 16#FFFFFFFF;
    pgc: POINTER TO SMC_GCODE_WORD := 0;
    piStartPosition: SMC_POSINFO;
    strProgramName: STRING := d'';
    bRestart: BOOL;
END_STRUCT
END_TYPE

```

Grâce à la variable wCNCREFStructID qui contient une valeur fixe, les modules vérifient en interne si la variable structurelle transmise est de type SMC_CNC_REF.

La variable pgc indique le premier SMC_GCODE_WORD.

La variable nelements contient le nombre de structures SMC_GCODE_WORD dans pgc.

La position de démarrage du programme CNC est enregistrée dans piStartPosition, sa désignation dans strProgramName.

Les variables diReadPos et udiBuffer sont à usage interne.

La variable bRestart est activée par le module SMC_NCDecoder lorsque des sauts doivent être utilisés dans le programme CNC (G20), elle signale au créateur de la structure de données que tous les pointeurs doivent être réinitialisés et que la structure de données doit à nouveau être créée.

SMC_GCODE_WORD

Cette structure de données permet d'enregistrer les mots G-Code :

```

TYPE SMC_GCODE_WORD :
STRUCT
    byLetter:BYTE:=0;
    fValue:LREAL:=0;
    diValue:DINT:=0;
    pAdr:POINTER TO BYTE:=0;
    byVarType:BYTE:=0;
END_STRUCT
END_TYPE

```

byLetter contient le code ASCII de la lettre du mot, fValue et diValue sa valeur sous la forme d'un nombre à virgule flottante et d'un nombre entier. Si ces variables sont utilisées à la place de valeurs fixes, un pointeur dans pADR indique ces variables ainsi que leur type dans byVarType :

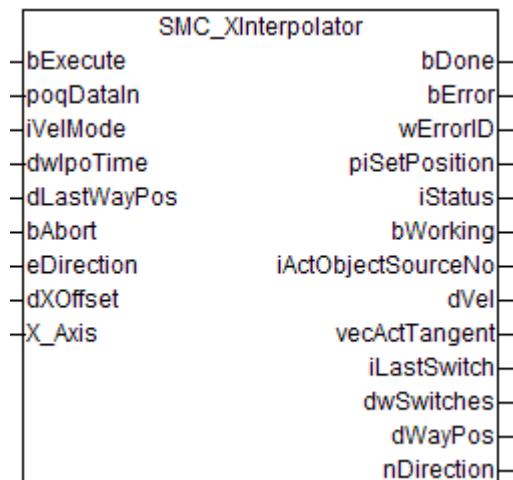
1	INT
2	BYTE
3	WORD
4	DINT
5	DWORD
6	REAL
14	SINT
15	USINT
16	UINT
17	UDINT
22	LREAL

6.6 Trajectoire CAM avec SMC_XInterpolator

Le module SMC_XInterpolator réalise un mélange entre CAM et CNC. Prenons le cas de figure suivant : une forme définie (via Code G) doit être découpée hors d'une pièce, cette pièce est commandée par un autre processus (p.ex. le long de l'axe X) et les autres axes (Y, Z, etc.) doivent être commandés en fonction de la position réelle de la pièce (X) et de la définition du contour de la trajectoire.

La pièce est toujours commandée dans le sens X ('autres cas de figure peuvent être visualisés par une rotation autour de l'axe X).

Le module SMC_XInterpolator possède les entrées et sorties ci-dessous :



Entrées du module :**bExecute : BOOL**

Le module procède à une réinitialisation et entame l'interpolation dès que cette entrée présente un front montant.

poqDataIn : POINTER TO SMC_OUTQUEUE

Cette entrée indique l'objet structuré SMC_OUTQUEUE contenant les objets SMC_GEOINFO de la trajectoire qui doit être interpolée.

dLastWayPos : LREAL

Grâce à cette entrée, l'utilisateur peut mesurer la distance parcourue par l'interpolateur. La sortie dWayPos est la somme de dLastWayPos et de la distance parcourue lors de ce cycle. Si dLastWayPos=0, dWayPos correspond à la longueur du tronçon effectif. Si dLastWayPos est identique à la sortie dWayPos, cette dernière est toujours incrémentée du tronçon effectif et on obtient ainsi le total de la distance parcourue. Cela dit, dLastWayPos peut toujours être remis à 0 ou sur une autre valeur.

bAbort : BOOL

Cette entrée interrompt l'exécution d'un contour.

eDirection : MC_Direction

Cette fonction permet de définir si la pièce est commandée le long de l'axe X dans le sens positif (positive) ou négatif (negative). d'autres valeurs ne sont pas admises.

dXOffset : LREAL

Décalage par rapport à la position de l'axe X.

X_Axis : AXIS_REF

Axe X, position de la pièce

Sorties du module :**bDone : BOOL**

Cette variable devient TRUE lorsque les données d'entrée de DataIn sont complètement exécutées. Sauf réinitialisation, le module n'effectue ensuite plus aucune action. Si l'entrée bExecute est FALSE, bDone devient à nouveau FALSE.

bError : BOOL

Si une erreur survient, cette valeur devient TRUE.

wErrorID : SMC_ERROR (INT)

Cette sortie d'énumération décrit le cas échéant une erreur survenue lors de l'interpolation. Suite à une erreur, l'exécution est interrompue jusqu'à une réinitialisation.

piSetPosition : SMC_POSINFO

Set_Position indique la position nominale calculée selon la définition. Set_Position présente une structure SMC_POSINFO et contient non seulement les coordonnées cartésiennes de point cible de la trajectoire, mais également la position des autres axes.

iStatus : SMC_INT_STATUS (INT)

Cette variable d'énumération indique le statut actuel du module. Voici les statuts possibles :

IPO_UNKNOWN	0	Statut interne. Ce statut ne peut pas survenir suite à l'exécution complète de SMC_Interpolator.
IPO_ACCEL	2	Le module se trouve en phase d'accélération.
IPO_CONSTANT	3	Le module se déplace à vitesse constante.
IPO_DECEL	4	Le module se trouve en phase de freinage.

IPO_FINISHED	5	L'exécution de la liste GEOINFO est terminée. Les objets GEOINFO survenant par après dans DataIn ne sont plus exécutés.
--------------	---	---

bWorking : BOOL

La sortie devient TRUE si l'exécution de la liste est commencée mais pas encore terminée (IPO_ACCEL ou IPO_CONSTANT ou IPO_DECEL ou IPO_WAIT). Dans les autres cas, Working est sur FALSE.

iActObjectSourceNo: INT

Cette variable reprend l'entrée SourceLine_Nr de l'objet GEOINFO de DataIn-Queue actuellement parcouru. Si SMC_Interpolator ne fonctionne pas (plus) (Working = FALSE), cette variable affiche -1.

dVel : LREAL

Cette variable reprend la vitesse réelle résultant du déplacement d'un objet des coordonnées spatiales précédentes vers Set_Position, dans le temps lpo_Time.

vecActTangent : SMC_VECTOR3D

Cette structure reprend le sens de la trajectoire au point Set_Position. Si Vel = 0, Act_Tangent affiche également des zéros.

iLastSwitch : INT

Cette variable reprend le numéro du repère parcouru en dernier. Si plusieurs repères sont traversés lors d'un cycle, celui parcouru en dernier lieu est toujours repris.

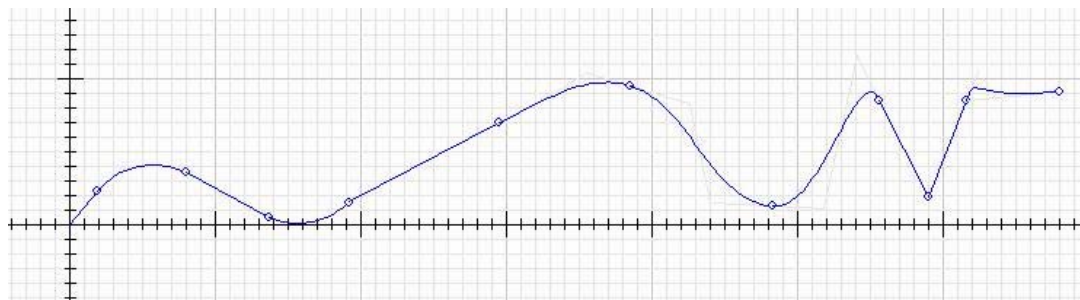
dwSwitches : DWORD

Cette variable DWORD contient le statut actuel de commutation de tous les repères entre 1 et 32. Bit0 de DWORD correspond au repère 1, Bit31 au repère 32. À l'inverse de iLastHelpMark, on peut ainsi exclure toute perte de repère.

dWayPos : LREAL

Pour une description, voir l'entrée dLastWayPos.

S'il est activé, XInterpolator recherche, par rapport à la position X réelle, la position sur la trajectoire définie et édite le point correspondant de la trajectoire en piSetPosition. Afin que ceci se passe sans saut, il faut toujours une position de trajectoire univoque correspondant à chaque position X. La trajectoire ci-dessous serait p.ex. admissible :



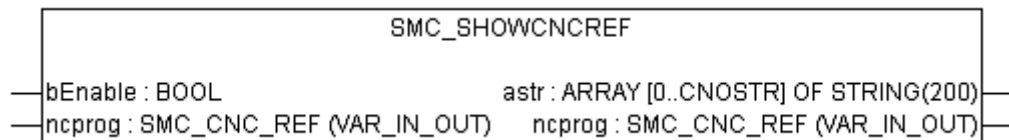
7 Bibliothèque SM_CNCDiagnostic.lib

Cette bibliothèque contient des adjuvants qui peuvent s'avérer utiles justement lors de la phase de mise en service car ils aident l'utilisateur à se faire une idée des données qui sont échangées entre les modules.

7.1 Modules pour l'analyse des données SMC_CNC_REF

7.1.1 SMC_ShowCNCREF

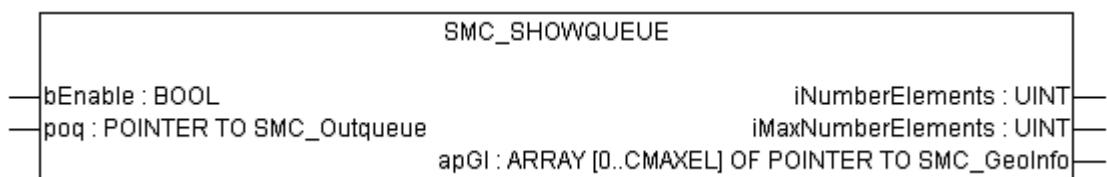
Ce bloc fonctionnel permet de représenter textuellement (Din66025) les dix premières lignes d'un programme NC présent sous la forme d'une structure de données SMC_CNC_REF (définie dans SM_DriveBasic.lib). Un Array of String est créé comme sortie (cnostr), contenant les lignes de texte. Le modèle de visualisation **VISU_SMC_ShowCNCRef** peut représenter ces tâches.



7.2 Modules pour l'analyse des données SMC_Outqueue

7.2.1 SMC_ShowQueue

Ce bloc fonctionnel (SM_CNCDiagnostic.lib) met les dix premiers objets SMC_GeoInfo d'une OutQueue à disposition sous la forme d'un ARRAY OF POINTER TO SMC_GeoInfo. Le modèle de visualisation **VISU_SMC_ShowQueue** peut en représenter quelques éléments importants. Parmi ceux-ci : numéro d'objet, numéro de ligne, type d'objet, position de départ (X/Y/Z), position de fin (X/Y/Z), vitesse de consigne et vitesse de fin.



8 Bibliothèque SM_Trafo.lib

8.1 Aperçu

Cette bibliothèque de module est une extension de SM_CNC.lib et propose des modules qui facilitent pour le programmeur IEC la transformation de coordonnées absolues en coordonnées machine ainsi que le contrôle des axes (position, position/vitesse, vitesse)..

Pour ce faire, elle contient deux modules qui contrôlent l'entraînement via des valeurs de consigne tout en surveillant ces dernières et en détectant d'éventuels sauts.

Elle contient en outre des modules de transformation mathématique vers l'avant et vers l'arrière pour quelques cinématiques courantes. Les instances des modules de transformation en avant peuvent être liés avec des modèles de visualisation également compris, permettant ainsi une représentation simple et immédiate.

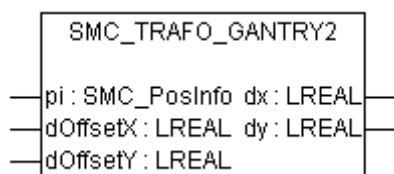
8.2 Blocs fonctionnels de transformation

Les modules de la bibliothèque SM_Trafo.lib appartenant à une cinématique spéciale sont combinés par paires, celui commençant avec SMC_TRAFO_<Cinématique> effectue un calcul en arrière et celui commençant avec SMC_TRAFOF_<Cinématique> un calcul en avant. Toute instance d'un module SMC_TRAFOF_<Cinématique> peut être reliée à un modèle de visualisation portant le nom de SMC_VISU_<Cinématique>.

8.2.1 Systèmes à portique

Les modules de la bibliothèque SM_Trafo.lib précisent en plus que pour des systèmes à portique, aucune transformation ne doit être effectuée, uniquement des décalages sur les axes x, y et z.

SMC_TRAFO_Gantry2



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

dOffsetX, dOffsetY : LREAL

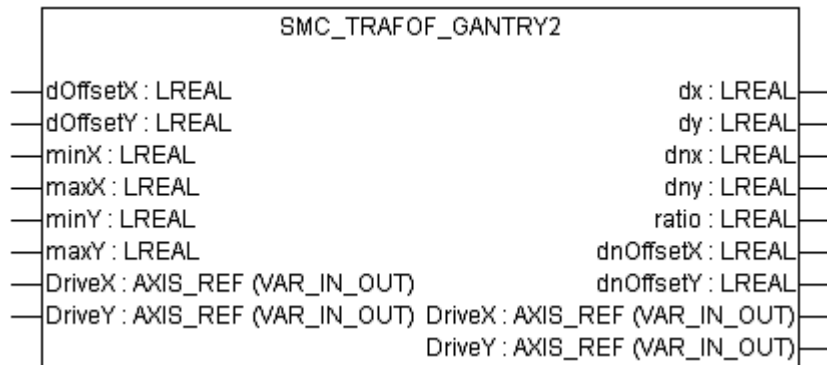
Décalage pour axes x et y.

Sorties du module :

dx, dy : LREAL

Valeurs de consigne pour axes x et y.

SMC_TRAFOF_Gantry2



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y. Valeurs identiques à celles de SMC_TRAFO_Gantry2.

minX, maxX, minY, maxY : LREAL

Plage de déplacement (pour la visualisation).

Entrées / sorties du module :

DriveX, DriveY : AXIS_REF

Axes x, y.

Sorties du module :

dx, dy : LREAL

Position x, y en coordonnées absolues.

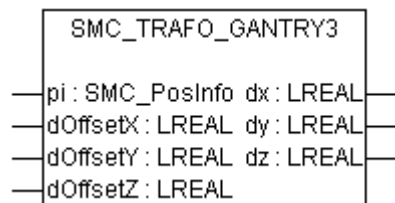
dnx, dny, dnOffsetX, dnOffsetY : LREAL

Position x et y normalisée [0..1] et décalages (pour la visualisation).

ratio : LREAL

Rapport entre l'intervalle x et l'intervalle y (pour la visualisation).

SMC_TRAFO_Gantry3



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

dOffsetX, dOffsetY, dOffsetZ : LREAL

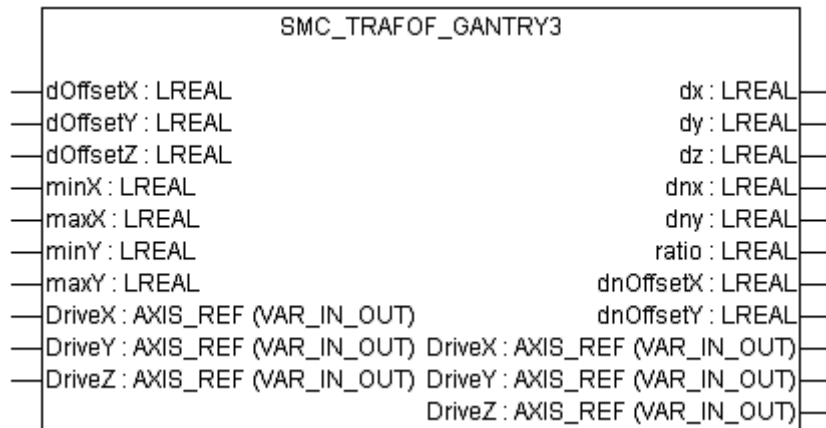
Décalage pour axes x, y et z.

Sorties du module :

dx, dy, dz : LREAL

Valeurs de consigne pour axes x, y et z.

SMC_TRAFOF_Gantry3



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

dOffsetX, dOffsetY, dOffsetZ : LREAL

Décalage pour axes x, y et z. Valeurs identiques à celles de SMC_TRAFO_Gantry3.

minX, maxX, minY, maxY : LREAL

Plage de déplacement (pour la visualisation).

Entrées / sorties du module :

DriveX, DriveY, DriveZ : AXIS_REF

Axes x, y, z

Sorties du module :

dx, dy, dz : LREAL

Position x, y, z en coordonnées absolues.

dnx, dny, dnOffsetX, dnOffsetY : LREAL

Position x et y normalisée [0..1] et décalage (pour la visualisation).

ratio : LREAL

Rapport entre l'intervalle x et l'intervalle y (pour la visualisation).

GantryCutter



Les modules SMC_TRAFO<n>_Gantry<n> sont également disponibles dans la bibliothèque SM_Trafo.lib en tant que SMC_TRAFO<n>_GantryCutter<n>.

Ces modules contiennent des transformations pour systèmes de portique à axe de rotation unique qui est dirigé dans le même sens que la tangente de la trajectoire actuelle. Ils contiennent des entrées supplémentaires : Axe rotatif (DriveR) qui en tant que tel doit être défini avec la période 360, Décalage (dOffsetX) et Sens de rotation (iDirectionR). Le module de transformation en marche arrière nécessite en outre le vecteur de la tangente actuelle (v) qui est présent comme sortie auprès de l'interpolateur.

SMC_TRAFOV_Gantry

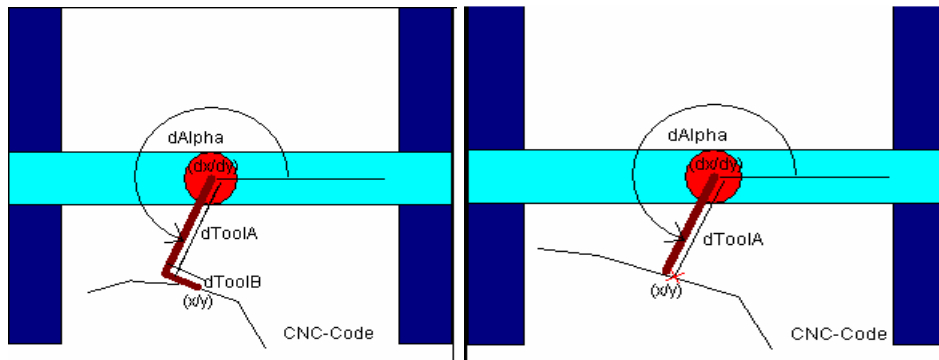
Quelques unes des transformations en marche arrière décrites ci-dessus et pour lesquelles des modules sont disponibles dans la bibliothèque SM_Trafo.lib sont à disposition dans une version dans laquelle la vitesse et la direction de la trajectoire sont utilisés comme grandeur de commande pour les axes. Elles commencent avec „SMC_TRAFOV_“ à la place de „SMC_TRAFO_“. L'interpolateur doit fournir comme entrées supplémentaires la tangente de trajectoire (v) et sa vitesse ($dVel$). Outre les positions de consigne, ces entrées donnent les vitesses de consigne ($dv_x/dv_y/dv_z$). Les avantages de cette méthode sont que l'erreur de poursuite au niveau de l'entraînement, due à la commande de vitesse, peut être minimisée – pour autant que l'entraînement supporte ce mode. Chaque axe doit de ce fait être commandé via le module SMC_ControlAxisByPosVel.

8.2.2 Systèmes à portique avec décalage d'outil

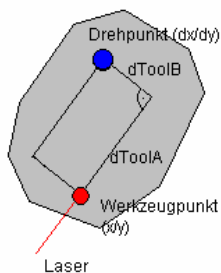
Il existe des systèmes de machines à portique dont les positions xyz ne coïncident pas avec les points de contact des outils : ces derniers ne sont en effet pas placés de manière axiale sur l'axe z mais sont décalés. Si l'axe z ne peut pas être pivoté, il s'agit d'un décalage constant xy qui peut être intégré en tant que tel dans la transformation standard gantry.

Par contre, si z présente une axe rotatif, il ne s'agit plus d'un décalage constant ; le déplacement est fonction de la position de l'axe C.

Il convient ici de distinguer un outil rectiligne (ce qui est le cas lorsque le vecteur entre d'une part le point de contact de l'outil et l'axe et d'autre part la direction souhaitée pour l'outil coïncident) (->SMC_TRAFO_Gantry2Tool1) d'un outil ressemblant à un parallélogramme ou à un triangle rectangle (SMC_TRAFO_Gantry2Tool2) :

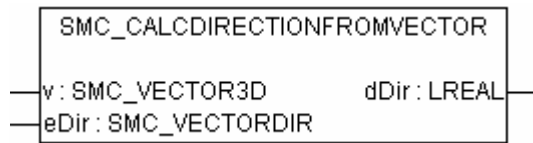


Dans cet exemple, l'outil ne peut pas être considéré comme rectiligne mais bien comme triangle rectangle.



En principe et à la place d'une transformation unidimensionnelle avec décalage d'outil (l'outil peut être considéré comme rectiligne), il est également possible de moduler la trajectoire en conséquence via SMC_Toolcorr. La différence entre les deux méthodes réside dans la vitesse du point d'outil. Alors qu'avec SMC_ToolCorr, la vitesse du centre de rotation est commandée en fonction des données du programme CNC (F, E) (le point d'outil peut donc varier), la vitesse du point d'outil est définie par le programme CNC si on utilise cette transformation.

La fonction ci-dessous peut être utilisée pour le calcul de l'alignement de l'outil ($dAlpha$) :

SMC_CalcDirectionFromVector**v : SMC_VECTOR3D**

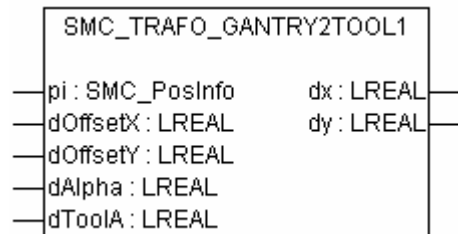
Le vecteur d'entrée v est normalement la sortie de l'interpolateur vecActTangent.

eDir : SMC_VECTORDIR

eDir permet de spécifier si la direction est calculée tangentielle par rapport à la trajectoire (SMC_tangential), y est opposée (SMC_opp_tangential) ou orthogonale (SMC_orthogonal_r (à droite de la tangente de trajectoire) et SMC_orthogonal_l (à gauche de la tangente de trajectoire)).

dDir : LREAL

La sortie dDir est donnée en degrés et reste constante pour les phases dans lesquelles se trouve l'interpolateur (v est le vecteur nul). eDir est la plupart du temps utilisé comme valeur de consigne (SMC_ControlAxisByPos) et sert de direction d'axe comme entrée dAlpha pour la transformation.

SMC_TRAFO_Gantry2Tool1**pi : SMC_PosInfo**

Vecteur de position de consigne. Sortie de l'interpolateur.

dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y.

dAlpha : LREAL

Alignement de l'outil en degrés.

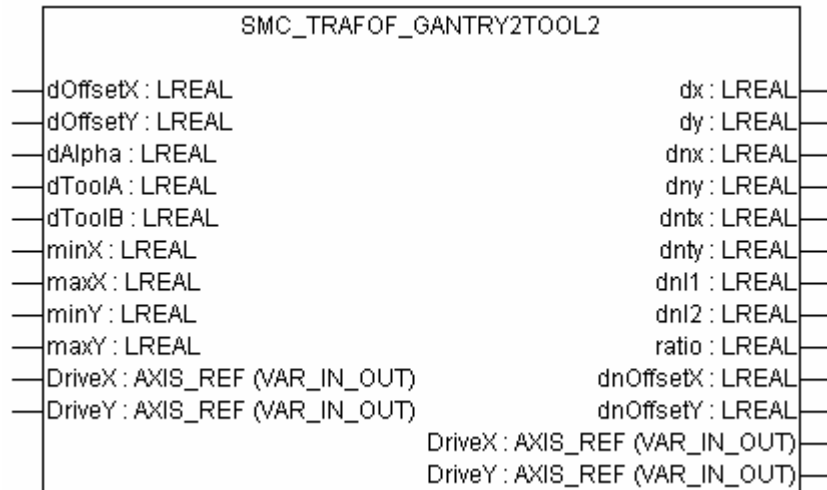
dToolA : LREAL

Longueur de l'outil ; distance entre le centre de rotation et le centre d'outil.

dx, dy : LREAL

Valeurs de consigne pour axes x et y.

SMC_TRAFOF_Gantry2Tool1



dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y. Valeurs identiques à celles de SMC_TRAFO_Gantry2.

dAlpha : LREAL

Alignement de l'outil en degrés.

dToolA : LREAL

Longueur de l'outil ; distance entre le centre de rotation et le centre d'outil.

minX, maxX, minY, maxY : LREAL

Plage de déplacement (pour la visualisation).

DriveX, DriveY: AXIS_REF

Axes x, y.

dx, dy : LREAL

Position x, y en coordonnées absolues.

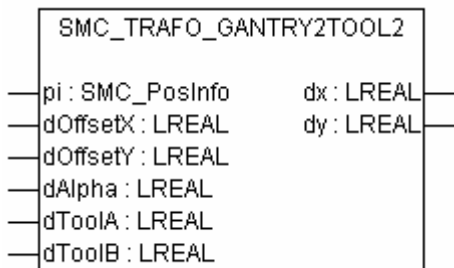
dnx, dny, dnl, dnOffsetX, dnOffsetY : LREAL

Position x et y normalisée [0..1], longueur d'outil et décalages (pour la visualisation).

ratio : LREAL

Rapport entre l'intervalle x et l'intervalle y (pour la visualisation).

SMC_TRAFO_Gantry2Tool2



pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y.

dAlpha : LREAL

Alignement de l'outil en degrés.

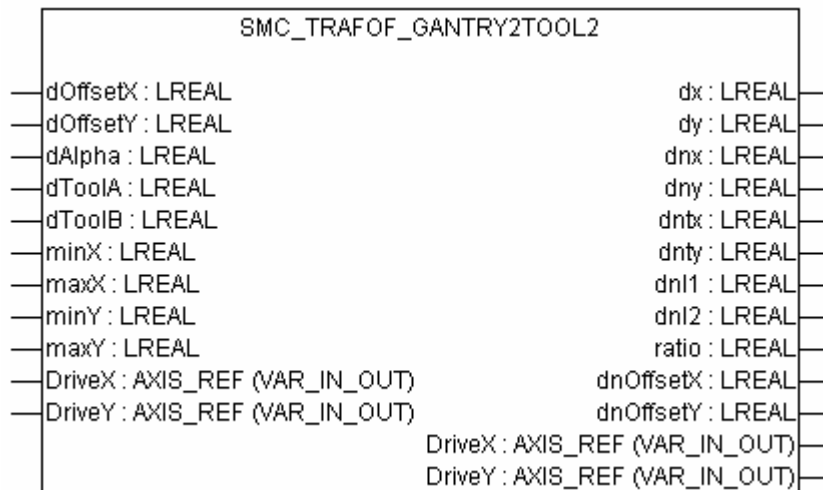
dToolA, dToolB : LREAL

Longueur de côté de l'angle droit du triangle rectangle situé entre le centre de rotation et le centre d'outil. dToolA est ici la longueur du côté de l'angle droit tangentiel par rapport à la trajectoire, dToolB la longueur du côté de l'angle droit orthogonal par rapport à la trajectoire.

Si dToolB est positive, le centre d'outil (x/y) est déplacé vers la gauche vu dans le sens de l'outil, il est sinon déplacé vers la droite.

dx, dy : LREAL

Valeurs de consigne pour axes x et y.

SMC_TRAFOF_Gantry2Tool2**dOffsetX, dOffsetY : LREAL**

Décalage pour axes x et y. Valeurs identiques à celles de SMC_TRAFO_Gantry2.

dAlpha : LREAL

Alignement de l'outil en degrés.

dToolA, dToolB : LREAL

Longueur de l'outil ; distance entre le centre de rotation et le centre d'outil.

minX, maxX, minY, maxY : LREAL

Plage de déplacement (pour la visualisation).

DriveX, DriveY : AXIS_REF

Axes x, y.

dx, dy : LREAL

Position x, y en coordonnées absolues.

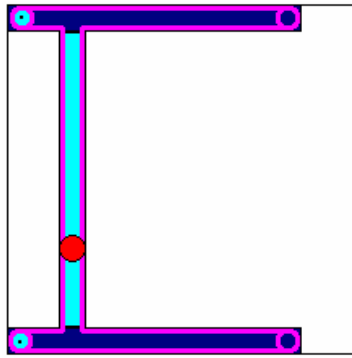
dnx, dny, dnl1, dnl2, dnOffsetX, dnOffsetY : LREAL

Position x et y normalisée [0..1], longueur d'outil et décalages (pour la visualisation).

ratio : LREAL

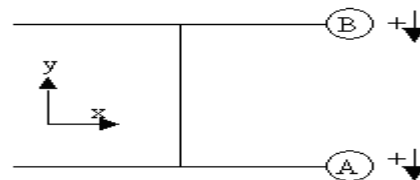
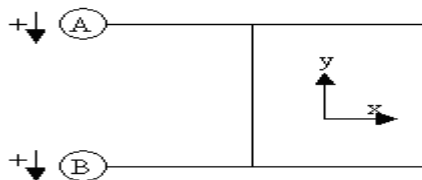
Rapport entre l'intervalle x et l'intervalle y (pour la visualisation).

8.2.3 Système de portiques H avec entraînements stationnaires



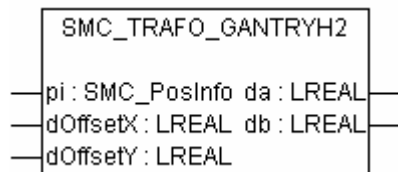
Ce système cinématique ressemble aux Systèmes de portiques décrits précédemment, à ces différences près que les entraînements sont de type stationnaire et qu'ils déplacent les chariots et l'axe Y par le biais de courroies à plusieurs renvois (roses dans l'image).

La transformation est conçue pour les configurations d'entraînement ci-dessous ; d'autres configurations peuvent être obtenues par la permutation de x et y :



Il convient de noter qu'une course de référence spéciale est nécessaire pour cette transformation. Si on souhaite effectuer un déplacement dans le sens Y, les entraînements A et B doivent être tourner dans le même sens ; pour un déplacement dans le sens x strict, les entraînement doivent tourner dans le sens opposé. Dès que la position de référence est trouvée, on utilise les valeurs x et y obtenues par la transformation en marche avant FB comme valeurs de translation (dOffsetX et dOffsetY).

SMC_TRAFO_GantryH2



Module pour système de portiques H avec entraînements stationnaires (SM_Trafo.lib).

Entrées du module :

pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

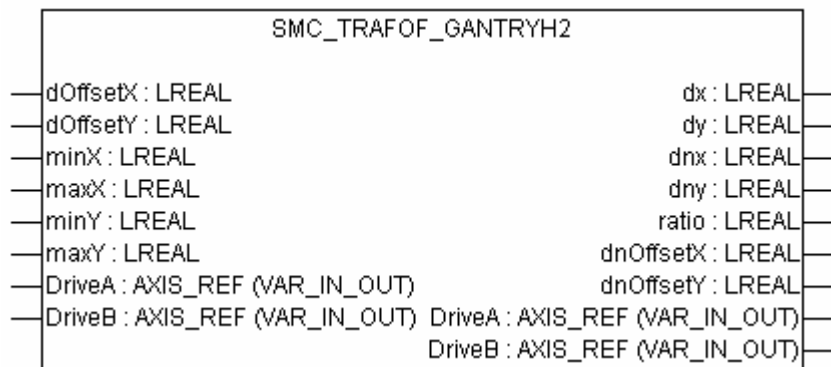
dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y.

Sorties du module :

da, db : LREAL

Valeurs de consigne pour axes A et B.

SMC_TRAFOF_GantryH2

Module pour système de portiques H avec entraînements stationnaires (SM_Trafo.lib).

Entrées du module :

dOffsetX, dOffsetY : LREAL

Décalage pour axes x et y. Valeurs identiques à celles de SMC_TRAFO_GantryH2.

minX, maxX, minY, maxY : LREAL

Plage de déplacement (pour la visualisation).

DriveA, DriveB : AXIS_REF

Axes A, B.

Sorties du module :

dx, dy : LREAL

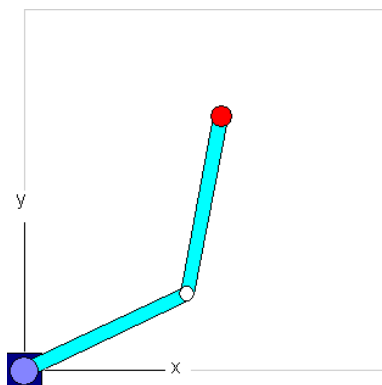
Position x, y en coordonnées absolues.

dnx, dny, dnOffsetX, dnOffsetY : LREAL

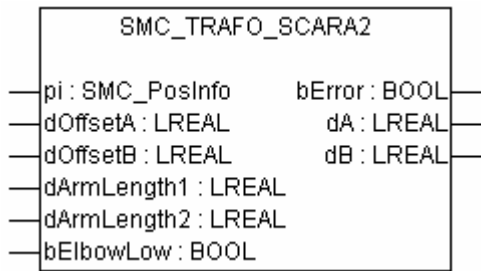
Position x et y normalisée [0..1] et décalages (pour la visualisation).

ratio : LREAL

Rapport entre l'intervalle x et l'intervalle y (pour la visualisation).

8.2.4 Systèmes Scara à double articulation

SMC_TRAFO_Scara2



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

dOffsetA, dOffsetB : LREAL

Décalage pour axes A et B.

dArmLength1, dArmLength2 : LREAL

Longueur des premier et second bras.

bElbowLow : BOOL

Coude vers le bas (TRUE) ou vers le haut (FALSE)

Sorties du module :

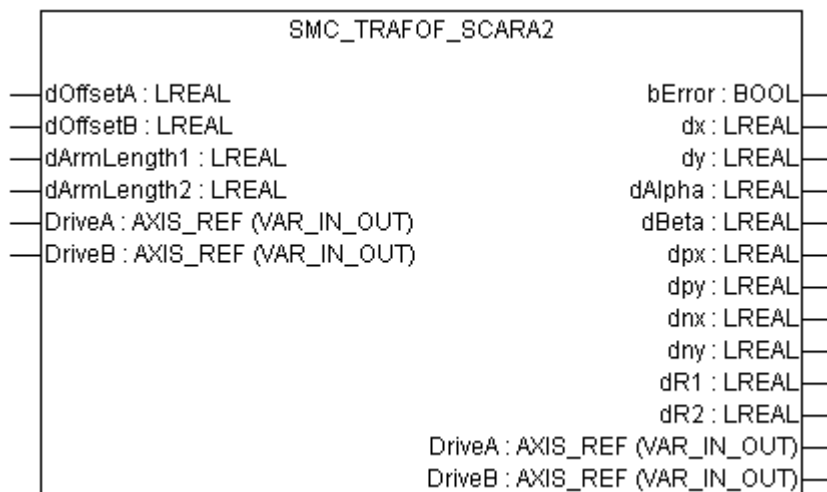
bError BOOL

TRUE : Valeurs inadmissibles.

dA, dB : LREAL

Position des axes A et B.

SMC_TRAFOF_Scara2



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

dOffsetA, dOffsetB : LREAL

Décalage pour axes A et B. Valeurs identiques à celles de SMC_TRAFO_Scara2.

dArmLength1, dArmLength2 : LREAL

Longueur des premier et second bras.

Entrées / sorties du module :

DriveA, DriveB : AXIS_REF

Axes A, B.

Sorties du module :

bError BOOL

TRUE : Valeurs inadmissibles.

dx, dy : LREAL

Position x, y en coordonnées absolues.

dAlpha, dBeta : LREAL

Angle d'articulation (positions d'axe sans décalage). (pour la visualisation)

dpx, dpy : LREAL

Position normalisée de la 1re articulation.]-1..1[(pour la visualisation)

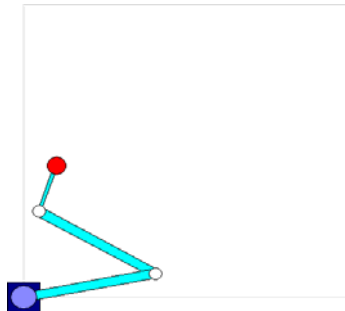
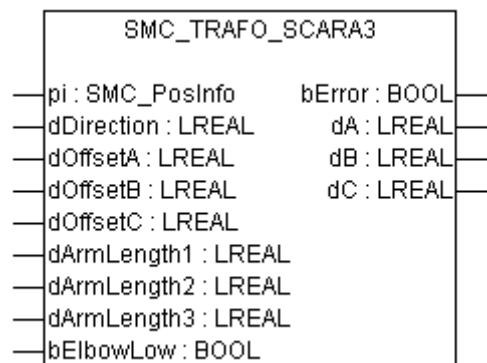
dnx, dny : LREAL

Position normalisée du manipulateur.]-1..1[(pour la visualisation)

dR1, dR2 : LREAL

Longueurs relatives des bras. $dR1 + dR2 = 1$.]0..1[(pour la visualisation)

8.2.5 Systèmes Scara à triple articulation

**SMC_TRAFO_Scara3**

Module de la bibliothèque SM_Trafo.lib

Entrées du module :

pi : SMC_PosInfo

Vecteur de position de consigne. Sortie de l'interpolateur.

Direction : LREAL

Angle de direction de la dernière articulation en degrés. (0° O, 90° N)

dOffsetA, dOffsetB, dOffsetC : LREAL

Décalage pour axes A, B. et C

dArmLength1, dArmLength2, dArmLength3 : LREAL

Longueur des bras

bElbowLow : BOOL

Coude (1re et 2e articulation) vers le bas (TRUE) ou vers le haut (FALSE)

Sorties du module :

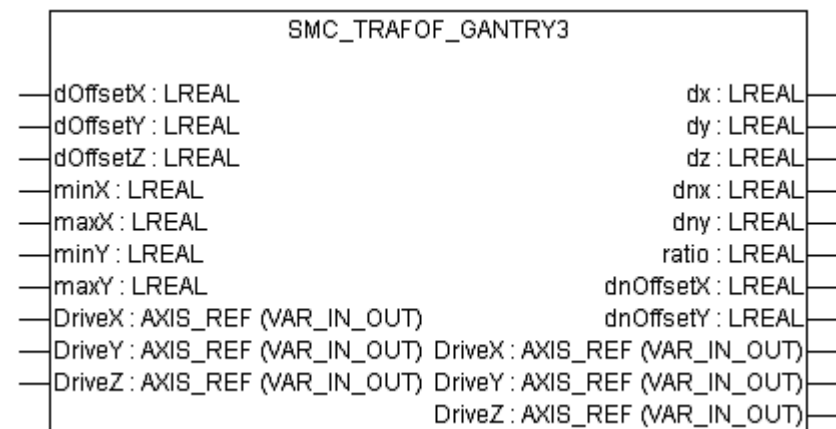
bError : BOOL

TRUE : Valeurs inadmissibles.

dA, dB, dC : LREAL

Position des axes A, B et C.

SMC_TRAFOF_Scara3



Module de la bibliothèque SM_Trafo.lib

Entrées du module :

dOffsetA, dOffsetB, dOffsetC : LREAL

Décalage pour axes A, B. et C Valeurs identiques à celles de SMC_TRAFO_Scara3.

dArmLength1, dArmLength2, dArmLength3 : LREAL

Longueur des bras

Entrées / sorties du module :

DriveA, DriveB, DriveC : AXIS_REF

Axes A, B. et C

Sorties du module :

bError : BOOL

TRUE : Valeurs inadmissibles.

dx, dy : LREAL

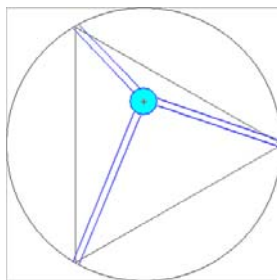
Position x, y en coordonnées absolues.

dAlpha, dBeta, dGamma : LREAL

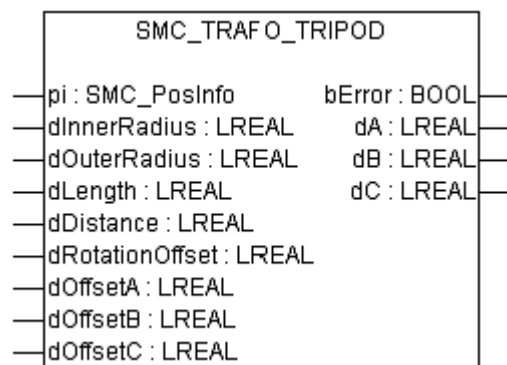
Angle d'articulation (positions d'axe sans décalage). (pour la visualisation)

dpx, dpy : LREALPosition normalisée de la 1re articulation. $]-1..1[$ (pour la visualisation)**dppx, dppy : LREAL**Position normalisée de la 2e articulation. $]-1..1[$ (pour la visualisation)**dnx, dny : LREAL**Position normalisée du manipulateur. $]-1..1[$ (pour la visualisation)**dR1, dR2, dR3 : LREAL**Longueurs relatives des bras. $dR1+dR2+dR3 = 1$. $]0..1[$ (pour la visualisation)

8.2.6 Cinématique parallèle



SMC_TRAFO_Tripod

**pi : SMC_PosInfo**

Vecteur de position de consigne. Position du centre de l'anneau intérieur. Sortie de l'interpolateur.

dInnerRadius : LREAL

Rayon de l'anneau intérieur

dOuterRadius : LREAL

Rayon de l'anneau extérieur

dLength : LREAL

Longueur de barre

dDistance : LREAL

Distance entre deux barres conjuguées sur l'anneau intérieur et l'anneau extérieur

dRotationOffset : LREAL

Direction en degrés (sens mathématique) dans laquelle l'axe A est disposé, vu de l'origine (0/0)

dOffsetA, dOffsetB, dOffsetC : LREAL

Décalage des axes individuels

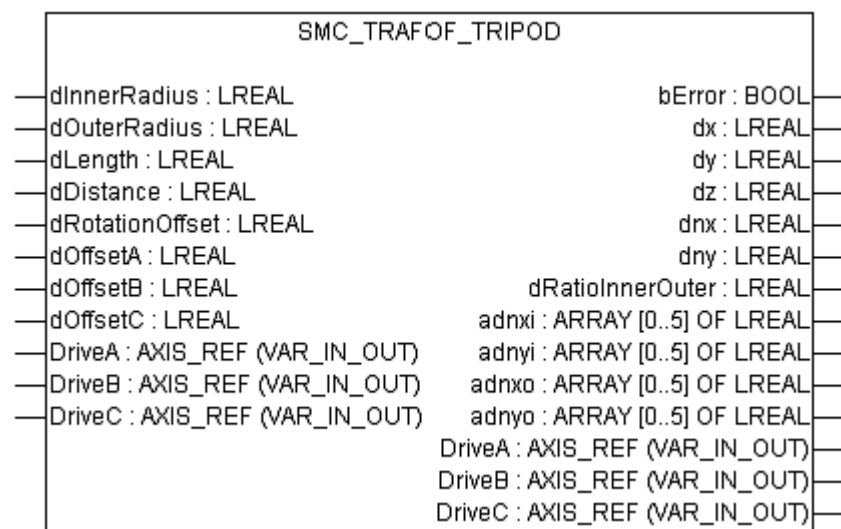
bError BOOL

TRUE : Valeurs inadmissibles. On quitte la plage de travail

dA, dB, dC : LREAL

Position des axes A, B et C.

SMC_TRAFOF_Tripod



dInnerRadius, dOuterRadius, dLength, dDistance, dRotationOffset, dOffsetA, dOffsetB, dOffsetC : LREAL

voir SMC_TRAFOF_TRIPOD

DriveA, DriveB, DriveC : AXIS_REF

Axes A, B et C

bError : BOOL

TRUE : Valeurs inadmissibles.

dx, dy, dz : LREAL

Position x, y, z du centre de l'anneau intérieur en coordonnées absolues.

dnx, dny : LREAL

Position normalisée du manipulateur. (pour la visualisation)

dRatioInnerOuter : LREAL

Rapport du rayon de l'anneau intérieur au rayon de l'anneau extérieur. (pour la visualisation)

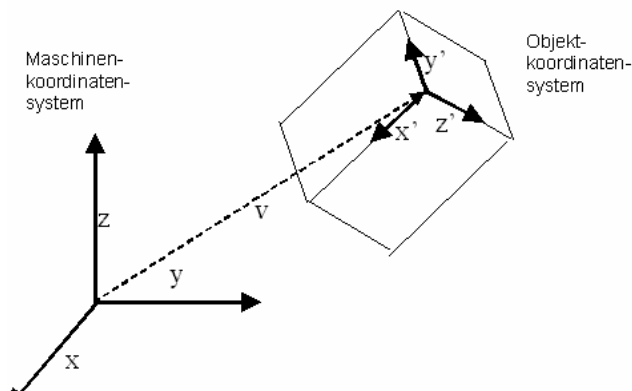
adnxi, adnyi, adnxo, adnyo : ARRAY[0..5] OF LREAL

Position normalisée de début et de fin des barres. (pour la visualisation)

8.3 Transformation dans l'espace

Il arrive que le système de coordonnées de la machine diffère de celui de la pièce à usiner. Les modules de transformation ci-dessous convertissent les coordonnées cartésiennes du point d'outil en coordonnées machine. Il peut cependant être au préalable nécessaire (si la pièce à usiner n'est pas alignée précisément conformément au programme CNC) de modifier les coordonnées de trajectoire calculées par l'interpolateur avant leur transmission à la transformation de machine.

Prenons un système à portique usuel (X/Y/Z). Il faut déplacer le point d'outil de ce système à portique sur la surface d'une pièce à usiner qui est placée de travers dans la zone.



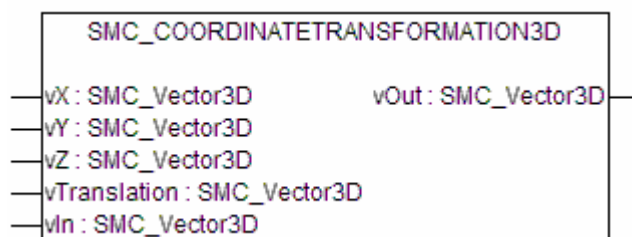
Il existe plusieurs possibilités pour décrire le rapport entre deux systèmes de coordonnées. Une transformation de coordonnées consiste toujours en une translation dans l'espace et une rotation dans l'espace. La translation est décrite au moyen d'un vecteur tridimensionnel ; la rotation est décrite soit par trois angles (p.ex. YawPitchRoll), soit par les trois vecteurs d'unité du nouveau système de coordonnées (objet) x' , y' , z' .

Si on choisit la méthode des trois angles de rotation, ceux-ci peuvent être définis p.ex. selon la convention Roll/Pitch/Yaw (RPY). Le nouveau système de coordonnées résulte alors de l'ancien système par une rotation autour de différents axes. On peut s'imaginer le procédé RPY (α , β , γ) comme étant des moyens différents qui fournissent le même résultat :

1. Partant du système de coordonnées (x, y, z) , on pivote le système de coordonnées autour de l'axe z selon un angle γ et dans le sens mathématique positif. On obtient ainsi un nouveau système de coordonnées $(x_1, y_1, z_1=z)$. Partant de ce dernier, on garde l'axe y_1 tel quel et on pivote le système de coordonnées de β , ce qui donne $(x_2, y_2=y_1, z_2)$. On pivote enfin ce système de coordonnées autour de x_2 selon l'angle α . On obtient alors $(x'=x_2, y', z')$.
2. Partant du système de coordonnées (x, y, z) , on pivote le système de coordonnées autour de l'axe x selon un angle α . Le système de coordonnées ainsi obtenu $(x_a=x, y_a, z_a)$ est pivoté autour de l'axe y original (pas y_a !!) d'un angle β (x_b, y_b, z_b) , puis enfin autour de l'axe original z d'un angle γ , ce qui donne (x', y', z') .

SMC_CoordinateTransformation3D

Ce module de la bibliothèque SM_Trafo.lib calcule les coordonnées d'une position (présente dans l'ancien système de coordonnées) par rapport au nouveau système de coordonnées. Pour ce faire, la transformation des coordonnées du nouveau système de coordonnées par rapport à l'ancien doit être définie via un vecteur de translation et les nouveaux vecteurs d'unité.



vX, vY, vZ : SMC_Vector3D

Vecteurs d'unité du nouveau système de coordonnées par rapport à l'ancien.

vTranslation : SMC_Vector3D

Vecteur de translation. Vecteur de l'ancienne origine de coordonnées par rapport au nouveau système de coordonnées.

vIn : SMC_Vector3D

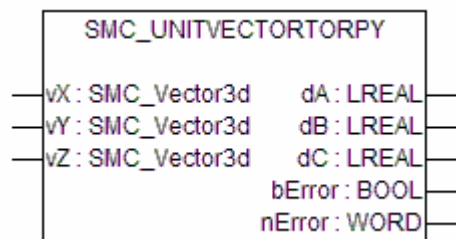
Position à transformer.

vOut : SMC_Vector3D

Position transformée.

SMC_UnitVectorToRPY

Module de la bibliothèque SM_Trafo.lib servant à calculer l'angle RPY de l'ancien système de coordonnées à partir des vecteurs d'unité du nouveau système de coordonnées.



vX, vY, vZ : SMC_Vector3D

Vecteurs d'unité du nouveau système de coordonnées par rapport à l'ancien.

dA, dB, dC : LREAL

Angle RPY en radian.

bError : BOOL

Valeurs de saisie improbables.

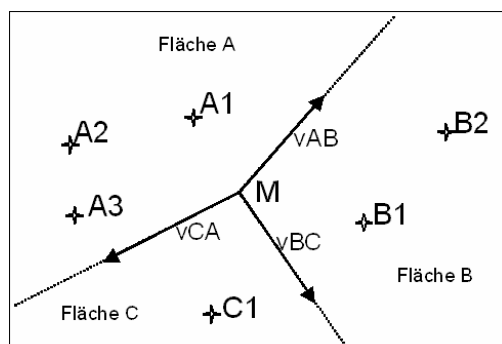
nError : WORD

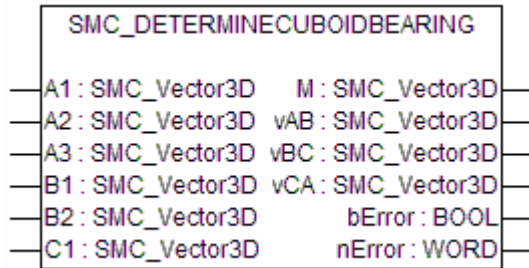
Description d'erreur :

- 0 Pas d'erreur (bError = FALSE)
- 1 Les vecteurs ne présentent pas la longueur 1
- 2 Les vecteurs ne sont pas perpendiculaires les uns par rapport aux autres
- 3 Pas de système droit

SMC_DetermineCuboidBearing

Module de la bibliothèque SM_Trafo.lib permettant de définir la position d'un parallélépipède (coin, sens des côtés) dans l'espace, via la saisie de 6 (3/2/1) points :



**A1, A2, A3 : SMC_Vector3D**

Trois points sur une surface latérale d'un parallélépipède ne pouvant pas se trouver sur une ligne droite.

B1, B2 : SMC_Vector3D

Deux points sur une autre surface latérale du parallélépipède, dont la projection sur la surface A ne peut pas être identique.

C1 : SMC_Vector3D

Points sur une autre surface latérale du parallélépipède.

M : SMC_Vector3D

Coin du parallélépipède.

vAB, vBC, vCA : SMC_Vector3D

Vecteurs d'unité sur les lignes latérales du parallélépipède

bError : BOOL

Valeurs de saisie improbables.

nError : WORD

Description d'erreur :

- 0 Pas d'erreur (bError = FALSE)
- 1 A1, A2, A3 se trouvent sur une ligne droite
- 2 La projection de B1 et celle de B2 sur la surface A sont identiques

9 Bibliothèque SM_Error.lib

9.1 Aperçu

Cette bibliothèque doit être présente dans tous les projets car elle contient toutes les définitions d'erreur. Elle contient toute les erreurs générées par les modules fonctionnels SoftMotion et peut les représenter sous la forme de chaînes.

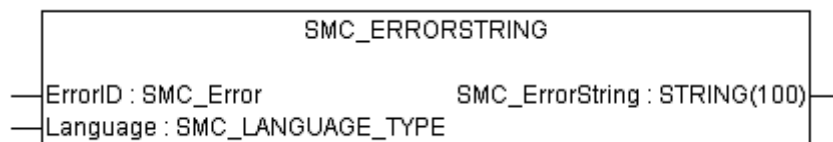
L'utilisateur doit noter que deux sortes d'erreurs peuvent en principe survenir dans son programme. d'une part des **erreurs d'entraînement**, il s'agit d'erreurs qui se produisent au niveau des entraînements (p.ex. erreur de poursuite, pas d'alimentation tension etc.). d'autre part des **erreurs de module**, il s'agit d'erreurs qui sont signalées par les modules via les sorties Error et ErrorID, elles sont souvent dues à un paramétrage incorrect.

Les **erreurs d'entraînement** doivent être lues via MC_ReadAxisError et MC_ReadParameter et le cas échéant supprimées avec MC_Reset. Les erreurs d'entraînement sont spécifiques aux entraînements et ne sont pas normalisées.

Les **erreurs de module** peuvent au besoin être converties en chaînes avec les fonctions de la bibliothèque SM_Error.lib. Comme ces erreurs peuvent survenir avec tous les modules SoftMotion et doivent être rassemblées au sein de l'application, une fonctionnalité supplémentaire a été implémentée dans la structure de données AXIS_REF, afin d'enregistrer une liste des erreurs survenues en dernier lieu. La sortie **FBEErrorOccured** de MC_ReadStatus permet de vérifier si des erreurs de modules sont survenues en dernier et de quel type d'erreur il s'agissait. Le bloc fonctionnel **SMC_ReadFBEError** reprend le numéro d'erreur de celle survenue en dernier lieu. La fonction **SMC_ClearFBEError** permet de supprimer l'erreur la plus ancienne.

9.2 Blocs fonctionnels

9.2.1 SMC_ErrorString



Selon les saisies ErrorID (SMC_Error) et Langage (SMC_LANGUAGE_TYPE (anglais, allemand)), la fonction SMC_ErrorString (SM_Error.lib) donne une représentation du chaîne de l'erreur.

9.3 L'énumération SMC_Error

L'énumération SMC_Error (SM_Error.lib) contient tous les numéros d'erreur qui sont générés par les blocs fonctionnels SoftMotion.

Voir également le module fonctionnel SMC_ErrorString.

Numéro d'erreur	Module	Valeur Enum	Description
0	tous	SMC_NO_ERROR	Pas d'erreur
1	DriveInterface	SMC_DI_GENERAL_COMMUNICATION_ERR OR	Erreur de communication (p.ex. rupture d'anneau Sercos)

L'énumération SMC_Error

Numéro d'erreur	Module	Valeur Enum	Description
10	DriveInterface	SMC_DI_SWLIMITS_EXCEEDED	Position en dehors de la plage admissible (SWLimit)
20	Tous les modules générant un mouvement	SMC_REGULATOR_OR_START_NOT_SET	Pas de validation de régulateur ou frein activé
30	DriveInterface	SMC_FB_WASNT_CALLED_DURING_MOTION	Le module générant un mouvement n'a plus été appelé avant la fin du mouvement
31	Tous les modules	SMC_AXIS_IS_NO_AXIS_REF	La variable AXIS_REF saisie n'est pas de type AXIS_REF
32	Tous les modules générant un mouvement	SMC_AXIS_REF_CHANGED_DURING_OPERATION	La variable AXIS_REF saisie a été remplacée alors que le module était en cours
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	Le mode prescrit l'utilisation de fins de course (à des fins de sécurité)
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	Mode non supporté
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	Axe pas en mode correct de régulation
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	Erreur lors du lancement du groupe d'axes
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	Valeurs incorrectes
91	SMC_ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	Les paramètres d'entraînement ne peuvent pas être modifiés tant que ce dernier est en cours de régulation
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	L'axe ne contient aucune mention du temps de cycle (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	L'axe était exempt d'erreur
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	L'axe ne procède pas à la réinitialisation de l'erreur
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	Impossible de réinitialiser l'erreur.
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	La communication avec l'axe n'a pas fonctionné.
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Numéro de paramètre inconnu
131	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Erreur lors de la transmission aux entraînements. Voir numéro d'erreur dans l'instance de module ReadDriveParameter (SM_DriveBasic.lib)
140	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	Numéro de paramètre inconnu ou écriture interdite

Numéro d'erreur	Module	Valeur Enum	Description
141	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	Erreur lors de la transmission aux entraînements. Voir numéro d'erreur dans l'instance de module WriteDriveParameter (SM_DriveBasic.lib)
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	L'axe n'était pas en statut „Arrêt“
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	Échec du démarrage de l'action Homing.
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	Erreur de communication.
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	Erreur lors de l'arrêt après Homing. Décélération définie ?
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	Erreur inconnue lors de l'arrêt
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	Direction=shortest pas applicable
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	L'entraînement est en statut de errorstop. L'arrêt ne peut pas être exécuté.
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	Erreur de direction
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	Erreur de direction
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	Erreur de direction
276	MC_MoveAdditive	SMC_MSI_INVALID_VELACC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
277	MC_MoveAdditive	SMC_MSI_INVALID_DIRECTION	Erreur de direction
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	Valeurs incorrectes de vitesse ou d'accélération
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest pas applicable
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	Taille de tableau incorrecte
326	MC_PositionProfile	SMC_PP_STEP0MS	Temps de pas = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	Taille de tableau incorrecte
351	MC_VelocityProfile	SMC_VP_STEP0MS	Temps de pas = t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	Taille de tableau incorrecte
376	MC_AccelerationProfile	SMC_AP_STEP0MS	Temps de pas = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Déclencheur déjà activé

L'énumération SMC_Error

Numéro d'erreur	Module	Valeur Enum	Description
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	La DriveInterface ne supporte la fonction de fenêtre
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Erreur de communication
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	Déclencheur déjà validé
426	SMC_MoveContinuousRelative	SMC_MCR_INVALID_VELACC_VALUES	Erreur de direction
427	SMC_MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	Valeurs de vitesse ou d'accélération incorrectes
451	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_VELACC_VALUES	Erreur de direction
452	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	Direction= plus rapide non applicable
453	SMC_MoveContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Erreur de direction
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	LA CAM ne contient pas de Tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	La Tappet-GroupID dépasse le MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	Plus de 32 accès à une CAM_REF
625	MC_CamIn	SMC_CI_NO_CAM_SELECTED	Pas de CAM sélectionnée
626	MC_CamIn	SMC_CI_MASTER_OUT_OF_SCALE	Axe maître en dehors de la plage admissible
627	MC_CamIn	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	Des valeurs de vitesse et d'accélération doivent être spécifiées pour la fonction ramp_in
628	MC_CamIn	SMC_CI_SCALING_INCORRECT	Variables de graduation fEditor/TableMasterMin/Max incorrectes
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	Accélération incorrecte
677	MC_GearIn	SMC_GI_INVALID_DEC	Décélération incorrecte
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	Valeurs de vitesse, d'accélération ou de décélération incorrectes
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	Axe rotatif avec fPositionPeriod = 0
750	Tous les modules qui utilisent MC_CAM_REF comme entrée	SMC_NO_CAM_REF_TYPE	La CAM saisie n'est pas du type MC_CAM_REF
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	Axe maître change pendant le couplage de la direction de rotation
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Trajectoire impraticable car vitesse de consigne = 0.
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Le tout dernier objet de la trajectoire a une Vel_End > 0.

Numéro d'erreur	Module	Valeur Enum	Description
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Attention: La liste GEOINFO est exécutée dans DataIn, mais la fin de la liste n'est pas définie. Cause : omission de la définition de EndOfList de la Queue de DataIn ou SMC_Interpolator plus rapide que les modules générant la trajectoire.
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	Vitesse au point d'arrêt > 0.
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	Trop de récurrences de SMC_Interpolator. Erreur SoftMotion.
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	La OutQueue DataIn d'entrée n'a pas parcouru SMC_CheckVelocities en dernier
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Erreur interne / numérique
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	Mémoire tampon des données trop petite
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	La trajectoire n'est pas complètement contenue dans la Queue
1080	SMC_Interpolator	SMC_WAR_INT_OUTQUEUE_TOO_SMALL	Avertissement : OutQueue DataIn trop faiblement dimensionnée. Le respect des arrêts n'est pas garanti.
1081	SMC_Interpolator	SMC_WAR_END_VELOCITIES_INCORRECT	Avertissement : vitesses finales contradictoires.
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	Valeurs de vitesse, d'accélération ou de décélération incorrectes
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Valeur d'accélération incorrecte
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Valeur de décélération incorrecte
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	Data underrun. La Queue a été vidée.
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	Mémoire tampon trop petite
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	Les éléments de tampon présentent un type incorrect.
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	Impossible de trouver la ligne actuelle de l'interpolateur
1500	Tous les modules fonctionnels qui utilisent SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	Le programme CNC saisi n'est pas de type SMC_CNC_REF
1501	Tous les modules fonctionnels qui utilisent SMC_OUTQUEUE	SMC_NO_OUTQUEUE_TYPE	La OutQueue saisie n'est pas de type SMC_OUTQUEUE
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	Le fichier n'existe pas
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	Pas de mémoire tampon créé

L'énumération SMC_Error

Numéro d'erreur	Module	Valeur Enum	Description
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	Mémoire tampon trop petite
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Data underrun. La mémoire tampon a été complètement lue.
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	Impossible de remplacer la variable de caractère de remplacement
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	L'entrée pvl n'indique pas l'objet SMC_VARLIST
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	Impossible d'ouvrir le fichier
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	Pas de mémoire tampon mentionné
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	Mémoire tampon trop petite.
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	Fin de fichier inattendue
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	Impossible d'ouvrir le fichier.
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	Débordement de mémoire tampon ; appeler WriteToFile plus fréquemment
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	Impossible d'ouvrir le fichier.
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	CAM enregistrée trop grande
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	Mode de compilation incorrect
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	Le fichier présente une version incorrecte
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	Fin de fichier inattendue
3001	SMC_WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	Canal de lecture des paramètres occupé
3002	SMC_WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	Impossible de créer fichier
3003	SMC_WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	Erreur lors de la lecture des paramètres
3004	SMC_WriteDriveParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	Timeout lors de la préparation de la liste des paramètres
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	Dénominateur du facteur de conversion (dwRatioTechUnitsDenom) de référence d'encodeur 0.
5001	SMC_Encoder	SMC_ENC_AXISUSED BY OTHER FB	Exécuter mouvement sur axe encodeur.

10 Bibliothèque SM_FileFBs.lib

10.1 Aperçu de la bibliothèque SM_FileFBs.lib

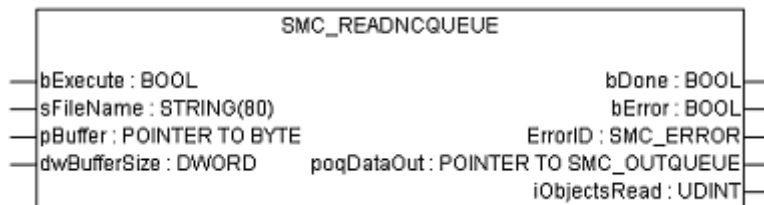
Cette bibliothèque propose des modules en rapport avec la fonctionnalité de fichier.

Leur utilisation suppose les bibliothèques du système 3S SysLibFile.lib et Standard.lib.

10.2 Blocs fonctionnels CNC

SMC_ReadNCQueue

Ce module lit un fichier OutQueue qui a été créé par l'Éditeur CNC (voir 3.3) du système de fichiers de la commande et met une structure OutQueue à disposition qui doit normalement être parcourue par l'interpolateur.



Entrées du module :

bExecute : BOOL

Le module débute la lecture de la Queue lors d'un front montant.

sFileName : STRING(80)

Chemin et nom du fichier.

pBuffer : POINTER TO BYTE

Pointeur sur une page de données libre et suffisamment grande, réservée au sein de l'application IEC.

dwBufferSize : DWORD

Taille de la page de données en octets.

Sorties du module :

bDone : BOOL

Cette sortie est activée si la queue a été entièrement lue.

bError : BOOL

TRUE : une erreur est survenue

ErrorID : SMC_ERROR

Numéro d'erreur

poqDataOut : POINTER TO SMC_OUTQUEUE

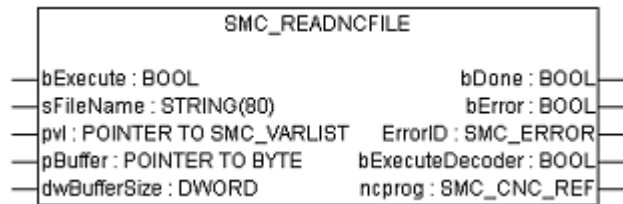
Pointeur sur la queue lue.

iObjectsRead : UDINT

Nombre d'objets SMC_GeoInfo qui ont été lus jusqu'à présent et sont écrits dans la queue.

SMC_ReadNCFile

Ce module peut lire un fichier NC-ASCII du système de fichiers de la commande afin de le mettre à la disposition du module SMC_NCDecoder. Ainsi, il est possible de lire et de convertir un programme NC en cours d'exécution.



Entrées du module :

bExecute : BOOL

Le module débute la lecture du programme lors d'un front montant.

sFileName : STRING(80)

Chemin et nom du fichier.

pvl : POINTER TO SMC_VARLIST

Pointeur sur un objet SMC_VARLIST. (voir ci-dessous) Si aucune variable n'est utilisée dans le programme CNC, cette entrée n'est pas affectée.

pBuffer : POINTER TO BYTE

Pointeur sur une plage de données libre et suffisamment grande, réservée au sein de l'application IEC.

dwBufferSize : DWORD

Taille de la plage de données en octets.

Sorties du module :

bDone : BOOL

Cette sortie est activée si le programme a été entièrement lu.

bError : BOOL

TRUE : une erreur est survenue

ErrorID : SMC_ERROR

Numéro d'erreur

bExecuteDecoder : BOOL

Signal avec lequel l'entrée Execute du module SMC_NCDecoder doit être déclenchée.

ncprog : SMC_CNC_REF

Programme CNC. Entrée du module suivant SMC_NCDecoder.

SMC_VARLIST-Struktur

IEC1131-3 ne prévoit pas de possibilité pour obtenir la valeur d'une variable à partir du nom symbolique de cette variable disponible par exemple sous forme de chaîne.

Cela devrait cependant être possible pour pouvoir utiliser la fonctionnalité des variables (voir 3.2) dont l'utilisateur dispose via la variante de compilation „variable programme“ (voir 3.7) lors de la lecture du programme CNC via des fichiers.

On dispose pour ce faire de la structure SMC_VARLIST.

Celle-ci se compose de la variable *wNumberVars* qui contient le nombre de variables utilisées, et d'un pointeur *psvVarList* sur le premier élément d'un tableau de **SMC_SingleVar**, lequel contient à son tour le type et un pointeur sur la variable.

Un objet SMC_SingleVar contient la chaîne de caractères *strVarName* qui à son tour contient le nom de la variable en majuscules, tel qu'il apparaît dans le programme NC. Il comprend en outre la valeur de la variable, pouvant être transmise selon le type en DINT (*dValue*) ou en REAL (*fValue*).

Exemple :

Le programme NC lu avec SMC_ReadNCFile à partir d'un fichier, contient deux variables *_rTestX* (REAL) et *g_byCommand* (BYTE) :

```
N0 G$g_byCommand$ X$g_fTestX$
```

On crée alors les variables suivantes :

```
g_byCommand: BYTE;
g_rTest: REAL;
asv: ARRAY[0..1] OF SMC_SingleVar :=
  (strVarName:='G_BYCOMMAND', eVarType:=SMC_TYPE_BYTE),
  (strVarName:='G_rTESTX', eVarType:=SMC_TYPE_REAL);
vl: SMC_VarList:=(wNumberVars:=2);
```

Avant l'appel du module SMC_ReadNCFile dont l'entrée pvl est ensuite écrite avec ADR(vl), il faut placer le pointeur sur la variable :

```
asv[0].pAdr:=ADR(g_byCommand);
asv[1].pAdr:=ADR(g_rTest);
```

Il faut également établir le lien entre SMC_VarList et SMC_SingleVar :

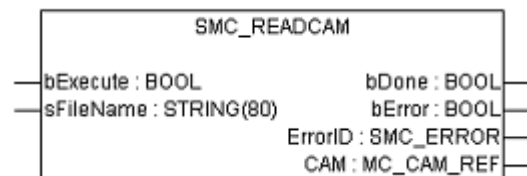
```
vl.psvVarList := ADR(asv[0]);
```

Si une variable ne peut pas être remplacée, une erreur est générée et le module est interrompu.

10.3 Blocs fonctionnels CAM

SMC_ReadCAM

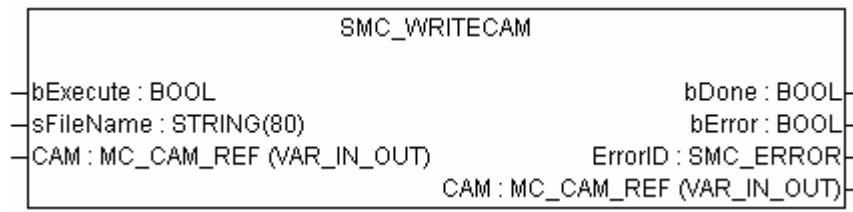
Ce module de la bibliothèque SM_FileFBs.lib est utilisé pour charger en cours d'exécution une CAM créée via l'Éditeur CAM et enregistrée dans un fichier *.CAM (voir 4.4.3) et pour mettre les modules MC_CamTableSelect et MC-CAMIn à disposition.



La taille d'une CAM pouvant être chargée est limitée par les constantes globales *gc_SMC_FILE_MAXCAMEL* (nombre d'éléments) et *gc_SMC_FILE_MAXCAMTAP* (nombre d'actions de came).

SMC_WriteCAM

Ce module de la bibliothèque SM_FileFBs.lib est utilisée pour enregistrer, dans un fichier *.CAM, une came créée avec l'éditeur CAM. Voir aussi: SMC_ReadCAM.



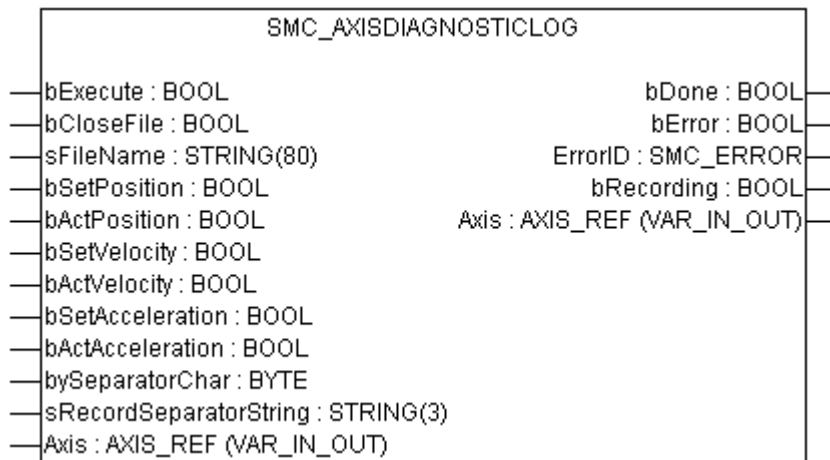
10.4 Blocs fonctionnels de diagnostic

SMC_AxisDiagnosticLog

Ce module peut être utilisé pour écrire les valeurs sélectionnées d'un axe de manière cyclique dans un fichier. Le fichier ainsi obtenu s'avère idéal pour un diagnostic.

Comme l'écriture de données sur un support prend normalement beaucoup de temps, ce module enregistre les données récoltées dans une mémoire tampon d'une taille de 5 ko, et ce n'est qu'à l'appel de l'action de module **WriteToFile** que les données sont écrites. Cet appel d'action doit de ce fait être inscrit dans une tâche plus lente (env. 50 ms) et moins prioritaire afin de ne pas interrompre la tâche Motion effective et de perturber le comportement de l'entraînement. Si la mémoire tampon est trop petite, le module génère une erreur.

Il convient de noter que l'utilisation de ce bloc fonctionnel peut influencer fortement sur le comportement dans le temps de l'application.



Entrées du module :

bExecute : BOOL

Le module débute lors d'un front montant. Il écrase le cas échéant les données existantes du nom sélectionné.

bCloseFile : BOOL

Le module ferme le fichier dès que cette entrée est TRUE.

sFileName : STRING(80)

Chemin et nom du fichier.

bSetPosition, bActPosition, bSetVelocity, bActVelocity, bSetAcceleration, bActAcceleration : BOOL

Ces entrées permettent de décider si les valeurs correspondantes des axes doivent être enregistrées dans le fichier.

bySeparatorChar : BYTE (valeur par défaut : TAB)

Code ASCII de la lettre qui doit être écrite entre deux valeurs de la même date.

sRecordSeparatorString : STRING(3) (valeur par défaut : ,R\$N')

Chaîne qui doit être écrite à la fin d'une date.

Axis : AXIS_REF;

Axe qui doit être scruté.

Sorties du module :

bDone BOOL

TRUE : Fin de session, fichier fermé.

bError : BOOL

TRUE : une erreur est survenue

ErrorID : SMC_ERROR

Numéro d'erreur

bRecording : BOOL

Le module enregistre.

11 Exemples de programmation

11.1 Aperçu

Pour commander le matériel hardware d'un entraînement au sein d'un projet CoDeSys au moyen de la fonctionnalité SoftMotion, il faut tenir compte des points suivants :

- La fonction SoftMotion doit être activée dans la **configuration du système cible** sous l'onglet „Généralités“.
- La bibliothèque **SM_DriveBasic.lib** et la bibliothèque spécifique au fabricant **<Désignation BusInterface>Drive.lib** doivent être intégrées au projet CoDeSys pour pouvoir utiliser l'interface d'entraînement à des fins de communication avec les entraînements.
- Dans l'**interface d'entraînement** (configuration de la commande), la structure du matériel hardware de l'entraînement doit être représentée et paramétrée ; pour ce faire, des variables globales sont automatiquement créées à la compilation du projet.
- Une **configuration de tâche** doit être définie.
- Un **programme IEC** avec éditeur CoDeSys doit être créé afin d'exécuter les mouvements souhaités via l'appel des modules appropriés. Pour pouvoir disposer des fonctions SoftMotion pertinentes pour ce traitement, les bibliothèques **SM_CNC.lib** et **SM_PLCOpen.lib** doivent être intégrées . Dans l'éditeur CNC ou CAM, des mouvements d'axes multiples ou des cames de commande des entraînements doivent être programmés de manière graphique et textuelle ; sur cette base, CoDeSys crée automatiquement les structures globales de données (données CNC, données CAM) avec lesquelles le programme IEC peut travailler.

Voir les exemples de programmation ci-dessous :

- Créer une configuration de commande pour des entraînements, chap.11.2
- Commande d'axes individuels (single-axis), chap.11.3
- Commande d'axe unique en CFC avec modèle de visualisation (single-axis), chap. 11.4
- Commande d'entraînement CAM via axe de temps virtuel, chap. 11.5
- CAM alternées, chap. 11.6
- Commande à l'aide de l'éditeur CNC, chap. 11.7
 - Exemple 1 : Création directe de la Queue, chap. 11.7.1
 - Exemple 2 : Décodage en ligne, utilisation de variables, chap. 11.7.2
 - Exemple 3 : Préparation de trajectoire en ligne, chap. 11.7.3
- Programmation dynamique SoftMotion, chap. 11.8

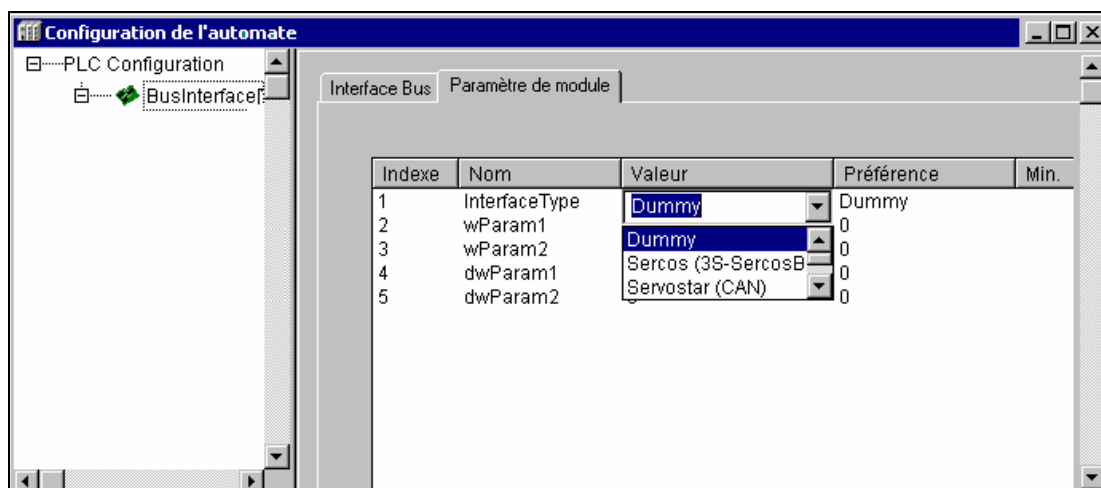
11.2 Drive Interface : Créer une configuration de commande pour des entraînements

*(Voir également le projet modèle fourni avec SoftMotion **DriveInterface.pro**, reposant sur le fichier de configuration **softmotion.cfg**)*

Cet exemple décrit la façon de représenter une structure physique d'entraînement dans le programme IEC. Grâce à cette configuration, le programme IEC dispose de structures de données sur lesquelles les modules fonctionnels SoftMotion travaillent et peuvent ainsi créer des mouvements.

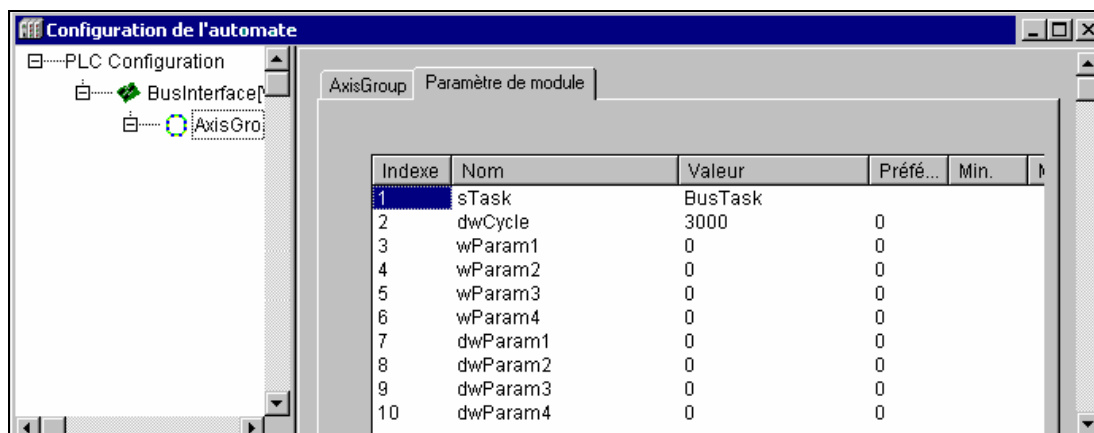
- Commencer par ouvrir la **Configuration de l'automate** et sélectionner dans le menu „Extras“ „Configuration par défaut“.

- Le bus de terrain utilisé est Sercos. On utilise par exemple pour ce faire une carte enfichable ISA Bus de la société Automata. C'est pourquoi il faut sélectionner dans le menu contextuel obtenu par un clic du bouton droit sur „Configuration de l'automate“ : „Ajouter sous-élément (BusInterface)“.
- Il faut maintenant régler pour cette BusInterface les **Paramètres de module** appropriés. Tout d'abord InterfaceType. Dans l'exemple, on ne dispose pas de connexion hardware et on peut sélectionner une sorte de mode de simulation grâce à l'option „Dummy“.



Les paramètres suivants à régler dépendent de cette saisie. Avec „Sercos (Automata)“ il s'agit p.ex. pour wParam1 du numéro Interrupt, pour wParam2 du type hardware et pour dwParam1 de l'adresse de base de la carte.

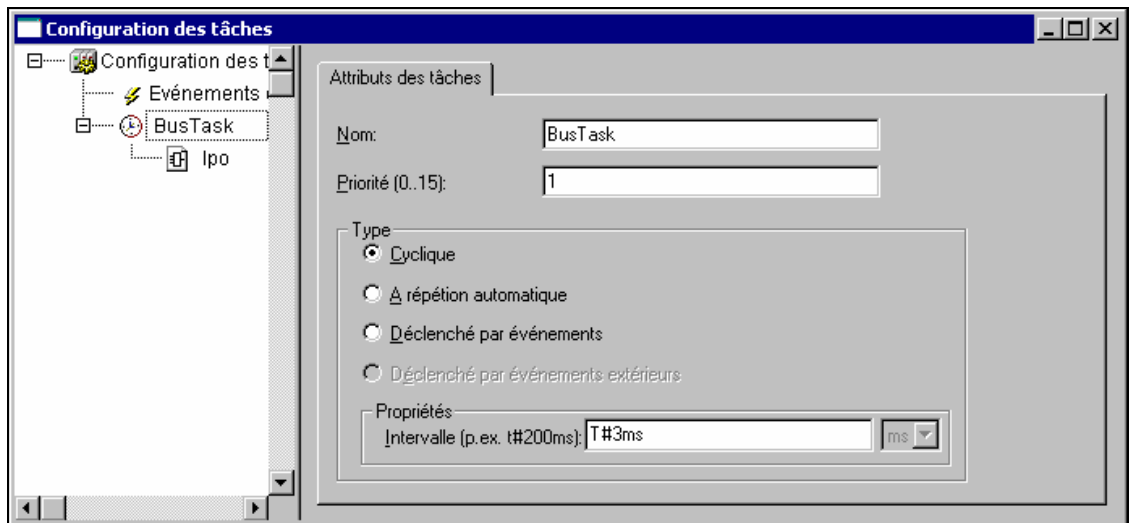
- Cette carte présente une boucle à laquelle sont connectés quatre entraînements à exploiter. C'est pourquoi on insère d'un clic du bouton droit de la souris un „Groupe d'axes“ sur le Businterface.



Il faut ici également saisir les **Paramètres de module**. Ensuite, le nom de la **tâche** est saisi dans „sTask“, tâche à partir de laquelle les entraînements doivent être exploités (p.ex. BusTask), la ligne en dessous reprend le temps de cycle de la tâche en µsec (p.ex. 3000). (Le paramétrage de la tâche est décrit dans le point suivant.)

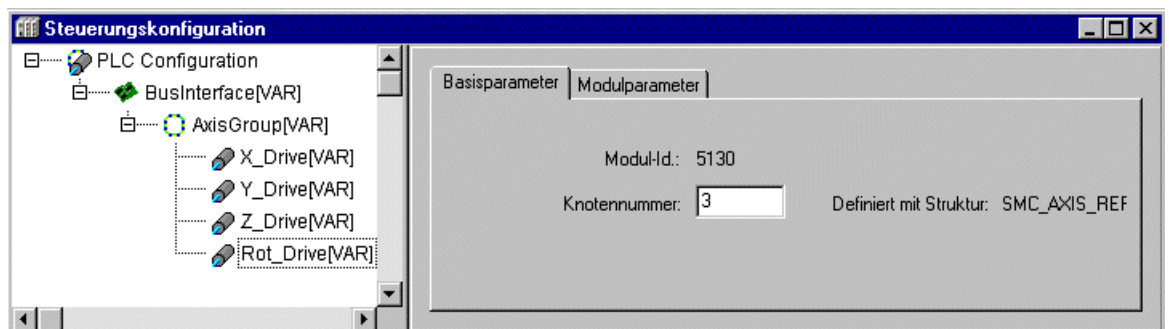
Les autres paramètres du groupe d'axe sont fonction du choix de la Businterface. Dans l'exemple, il s'agit pour le wParam1 du taux de baud en MBit (p.ex. 2) et pour wParam2 de l'intensité des LED.

- Ouvrir la **Configuration des tâches** et créer une tâche avec ces propriétés:



Il faut maintenant **incorporer les entraînements**. On dispose de quatre entraînements dont trois linéaires, commandant un système à portique X, Y, Z, et un quatrième qui fait tourner l'outil dans l'axe Z.

Si l'entrée „Groupe d'axes“ est sélectionnée dans l'arborescence de configuration (cadre pointillé), un clic du bouton droit de la souris suivi de la sélection de „Insérer entraînement“ permet d'incorporer successivement quatre entraînements. Les noms des entraînements peuvent être affectés de tout identificateur IEC, pour autant que suite à un clic sur l'entrée, un champ d'édition du nom apparaisse à l'écran. Dans l'exemple, on saisit la désignation suivante :



- Les entraînements doivent à présent **être paramétrés** l'un après l'autre :

Il faut tout d'abord régler pour tous les entraînements et selon leur configuration l'ID „wDriveld“. On décrit ci-dessous la manière de procéder aux réglages sous l'onglet „paramètre de module“ ; cette procédure est beaucoup plus simple et claire via le dialogue.

Les entraînements de portique peuvent tous être déplacés de -50cm et +50cm. Leur pondération est configurée par translation. Un incrément pour toutes les données de position et de vitesse et de 10-7m ou 10-7m/s. Si on veut à la place que toutes les données IEC de position et de vitesse se rapportent à des mm ou des mm/sec, il faut saisir dans dwRatioTechUnitsDenom „10000“, et „1“ dans iRatioTechUnitsNum. Comme il s'agit d'un entraînement linéaire, fPositionPeriod n'a aucune signification. Il reste maintenant à spécifier les données qui sont envoyées et reçues de manière cyclique, p.ex. lorsque l'on souhaite se rapporter à un télégramme de priorité préférentielle, il suffit de rechercher dans „wControlType“ „POS, VEL -> POS, VEL“. Ceci permet d'envoyer de manière cyclique des valeurs de consignes pour la position et la vitesse et de maintenir les valeurs réelles de position et de vitesse. Pour contrôler en plus tout dépassement de la plage admissible (-50cm = -5000mm, 50 cm = 5000mm) (l'application doit être programmée de telle sorte qu'un dépassement de cette plage soit impossible), on active une fonction de contrôle en réglant SWLimitEnable = TRUE, SWLimitNegative = -5000 et SWLimitPositive = 5000.

Sur les entraînements rotatifs utilisés, une rotation se compose de 65536 incréments. Il faut dès lors saisir „65536“ sous dwRatioTechUnitsDenom et „360“ sous iRatioTechUnitsNum pour obtenir une unité interne en degrés. Cet entraînement est conçu par exemple pour pouvoir visser un capuchon sur une bouteille. La position et le couple doivent être envoyés de manière cyclique afin de pouvoir passer ultérieurement d'une régulation du positionnement à une régulation du couple en modifiant le mode de fonctionnement. Pour ce faire, wControlType est réglé sur „CONFIGURABLE“ et on saisit sous wCyclicDataS1/2 „fSetPosition“ et „fSetTorque“. Afin d'obtenir la valeur réelle de positionnement, on saisit sous wCyclicDataR1 „fActPosition“.

- Afin de pouvoir compiler le programme correctement, il faut encore ajouter un appel de programme à la tâche „BusTask“. Par exemple, on crée un programme „lpo“ dans lequel la commande des mouvements aura lieu ultérieurement, et on l'appelle de „BusTask“.

Un **processus de compilation** correct doit maintenant être possible et le programme peut être **chargé et démarré sur la commande**. Les variables et structures suivantes sont automatiquement créées.

→ Le dossier des variables globales „Drive Configuration Data“ comprend maintenant trois instances des modules „SercosDriveExecute_Start“, „SercosDriveExecute_End“ et „SercosDriveInit“ provenant de la bibliothèque Sercos portant les noms „AxisGroupStartCycle“, „AxisGroupEndCycle“ et „AxisGroupInit“, lesquels sont responsables de la communication avec les entraînements.

→ Le dossier des variables globales „Drive_Globale_Variablen“ de la bibliothèque „SM_DriveBasic.lib“ contient une variable structurelle g_DRIVESTRUCT qui à son tour contient toutes les entrées de la configuration de commande, c.-à-d. toutes les interfaces de bus, tous les groupes d'axes et tous les entraînements.

→ En outre, une variable structurelle est créée pour chaque entraînement de manière globale, p.ex. „X_Drive“, quel'on peut p.ex. consulter via le gestionnaire espion et de recettes. Les modules SoftMotion travaillent sur ces structures et la DriveInterface veille à la mise à jour des structures.

Les modules Motiondu programme IEC travaillent sur ces structures d'entraînement qui ont pu être créées dans **lpo**.

11.3 Commande d'axes individuels (single-axis)

(Voir également le projet modèle fourni avec SoftMotion *PLCopenSingle.pro*, reposant sur le fichier de configuration *softmotion.cfg*)

Commande d'axes individuels (single-axis)

En plus des **bibliothèques** de Interface entraînement, la bibliothèque SM_PLCopen.lib doit être intégrée au projet.

Un entraînement linéaire dénommé "Drive" est défini dans la **Configuration de l'automate** :



Dans la **Configuration des tâches**, appeler le programme "lpo" qui permet de générer un mouvement sur cet axe.

Ce **programme** doit être créé par après, c'est pourquoi un programme est créé et complété en langage ST dans l'Organisateur d'objets.

Avant de pouvoir procéder à la commande de l'entraînement, il faut veiller à ce que le pilote ait trouvé et initialisé cet entraînement. Si tel est le cas, l'entraînement doit recevoir la validation du régulateur et il faut le cas échéant desserrer le frein. Cette fonction est prise en charge par le module MC_Power :

```

PROGRAM Ipo
VAR
  Init: BOOL := FALSE;
  Power: MC_Power;
END_VAR
IF NOT Init THEN
  Power(Enable:=TRUE, bRegulatorOn:=TRUE, bDriveStart:=TRUE, Axis:=Drive);
  Init:= Power.Status;
ELSE
END_IF

```

Il est donc possible de commander l'entraînement dans la branche ELSE de la première condition IF. Dans cet exemple, le module de positionnement **MC_MoveAbsolute** doit prendre cette commande en charge. Déclarer pour ce faire une instance de ce module ainsi qu'une position cible p qui doit être initialisée à 100. Il faut maintenant appeler chaque cycle de cette instance de module avec les valeurs nécessaires. Lorsque la position programmée est atteinte, la sortie Done du module est activée et l'entrée Execute doit devenir FALSE avant de pouvoir procéder à un nouveau déplacement : en effet, le module ne procède à un nouveau déplacement qu'avec un front montant :

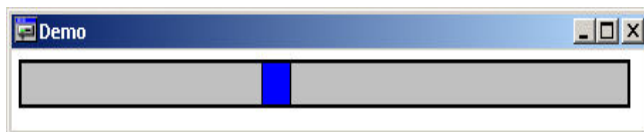
```

... (* Suite du programme illustré ci-dessus *)
ELSE
MoveAbsolute(Execute:=TRUE, Position:=p, Velocity:=100, Acceleration:=100,
Deceleration:=100, Axis:=Drive);
IF MoveAbsolute.Done THEN
  MoveAbsolute(Execute:=FALSE, Axis:=Drive);
END_IF
END_IF

```

Il est maintenant possible de compiler correctement le programme, d'ouvrir une session en ligne et de le démarrer. Si on observe la position réelle Drive.fActPosition à l'aide d'une liste d'espion ou de l'histogramme, on voit le mode d'accès de l'entraînement à cette position. Si on force la valeur de p, l'axe se déplace vers cette nouvelle cible après avoir atteint la cible précédente.

Des modèles de visualisation d'entraînement (voir 0) sont disponibles dans la bibliothèque SM_DriveBasic.lib afin de suivre le déplacement sur un graphique. Pour utiliser ces modèles, il faut tout d'abord se rendre hors ligne, créer une nouvelle visualisation, y insérer un objet de visualisation et sélectionner "LinDrive" dans la liste apparaissant à l'écran. Double-cliquer ensuite sur le nouvel objet et saisir sous "Visualisation", "Caractère de remplacement", "AXISREF" pour remplacer le nom de la structure d'entraînement (ici : Drive). La visualisation ainsi obtenue représente la position de l'entraînement :



11.4 Commande d'axe unique en CFC avec modèle de visualisation (single-axis)

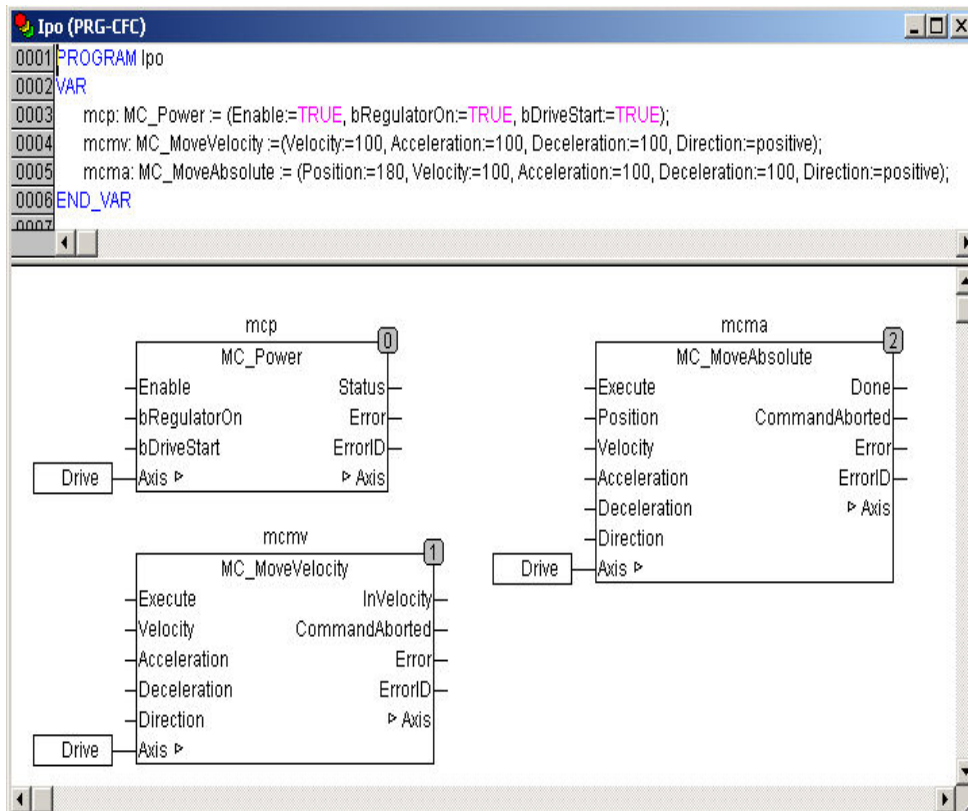
(Voir également le projet modèle fourni avec SoftMotion **PLCopenSingle2.pro**, reposant sur le fichier de configuration **softmotion.cfg**)

À la place du langage ST, on peut utiliser tout autre langage de programmation IEC comme l'exemple suivant le montre.

Cet exemple a pour but d'illustrer le mécanisme de démarrage et d'interruption des blocs fonctionnels. En outre, différents modes de démarrage peuvent être essayés pour le module MC_MoveAbsolute, pour des axes rotatifs.

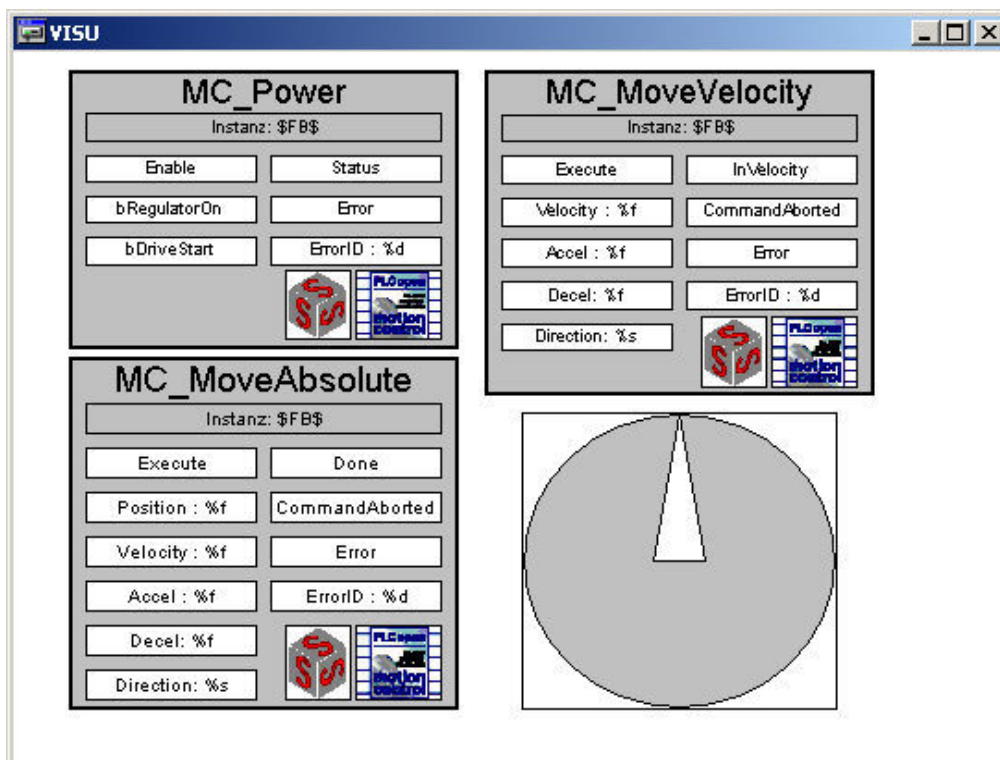
Imaginons une configuration de commande et de tâche comme dans l'exemple précédent, sauf qu'il s'agit d'un entraînement rotatif avec période 360. Le programme „Ipo“ est écrit en CFC et ne contient

que trois appels d'instances des modules MC_Power (nécessaire pour l'activation des axes), MC_MoveAbsolute et MC_MoveVelocity :



L'initialisation des entrées de module est recommandée afin de ne pas devoir saisir à nouveau ces valeurs lors des démarrages ultérieurs de cette application de test.

Créons en plus un système de visualisation de commande pour lequel nous utilisons les modèles de modules disponibles dans la bibliothèque et que nous relient aux instances de module via le concept des caractères de remplacement.



Nous pouvons alors compiler correctement le projet, ouvrir une session sur la commande et démarrer. Une pression sur l'entrée Exécute de MoveVelocity met l'entraînement en marche. Une pression sur Exécute de MoveAbsolute déplace l'entraînement sur la position réglée, le déplacement s'effectuant uniquement dans les sens positif conformément à la mention Direction :positive. Le module MoveVelocity s'en trouve interrompu. Il est recommandé de se familiariser avec ces modules, d'essayer différentes vitesses et accélérations et de tester les modes de direction (positive/negative/current/shortest/fastest) de MoveAbsolute.

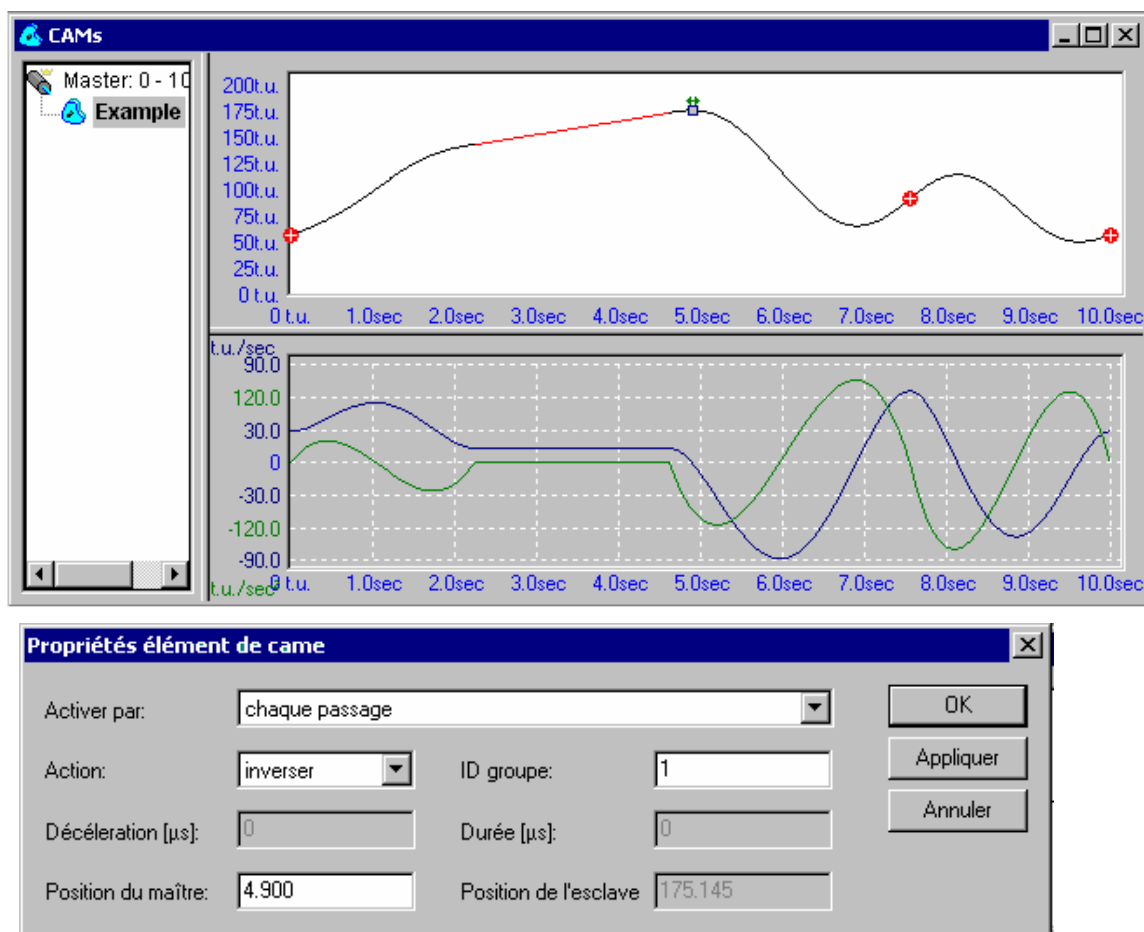
11.5 Commande d'entraînement CAM via axe de temps virtuel

(Voir également le projet modèle fourni avec SoftMotion *PLCopenMulti.pro*, reposant sur le fichier de configuration *softmotion.cfg*)

(Condition : les bibliothèques DriveBasic.lib et SM_PLCopen.lib sont reliées au projet.)

L'exemple ci-dessous montre que l'on peut convertir une CAM périodique en un entraînement linéaire. On y démontre en plus la fonction de came.

1. Pour ce faire, il faut tout d'abord créer dans l'éditeur CAM CoDeSys une CAM périodique au gré qui se rapporte à un axe maître entre 0 et 10 sec et qui contient au moins une came d'inversion avec ID1. Par exemple :



2. Un entraînement linéaire dénommé „Drive“ est défini dans l'interface de l'entraînement (Configuration de l'automate) :



3. Le programme Ipo est créé en langage FBD et contient les appels de ces modules :

```

PROGRAM Ipo
VAR
    Power: MC_Power;
    TimeAxis : SMC_TimeAxisFB;
    TableSelect: MC_CamTableSelect := (SlaveAbsolute:=TRUE);
    CamIn: MC_CamIn:=(StartMode:=ramp_in, VelocityDiff:=100, Acceleration:=100, Deceleration:=100);
    Tappet: SMC_GetTappetValue;
END_VAR
  
```

Après le module d'alimentation (Power) pour l'axe esclave, on appelle d'abord le module d'axe de temps (TimeAxis). Ce dernier obtient comme période 10 sec car la CAM est conçue pour cette durée. Le temps de cycle de la tâche doit être saisi manuellement. TableSelect permet de sélectionner la CAM souhaitée, CamIn convertit cette dernière. Le module Tappet vérifie la position de la came. Comme ce dernier a été programmé en mode inverse, il est commuté toutes les 10 secondes.

Le programme peut alors être compilé et lancé sur la commande.

Afin de contrôler la position de consigne et la position réelle, on génère à nouveau une visualisation à l'aide de laquelle on peut contrôler

Il convient de noter que le maître CAM peut être non seulement un axe de temps virtuel mais également toute structure de données AXIS_REF. À cet égard, les valeurs de consigne sont prises en compte pour les entraînements en phase de paramétrage alors que les valeurs réelles le sont pour les entraînements libres.

11.6 CAM alternées

(Voir également le projet modèle fourni avec SoftMotion *PLCopenMultiCAM.pro*, reposant sur le fichier de configuration *softmotion.cfg*)

(Condition : les bibliothèques DriveBasic.lib et SM_PLCopen.lib sont reliées au projet.)

Cet exemple montre comment un mouvement de CAM combinant deux comes peut être réalisé. Il a été programmé en langage ST et exécute les mêmes actions que dans l'exemple précédent. À la fin de la première CAM, le module MC_CamIn active la sortie bEndOfProfile, sur base de laquelle l'autre MC_CamTableSelect intervient et MC_CamIn est redémarré.

11.7 Commande à l'aide de l'éditeur CNC

Vous trouverez ci-après une description de la structure de base d'un programme IEC réalisable sous CoDeSys et qui peut convertir des trajectoires définies dans l'éditeur CNC.

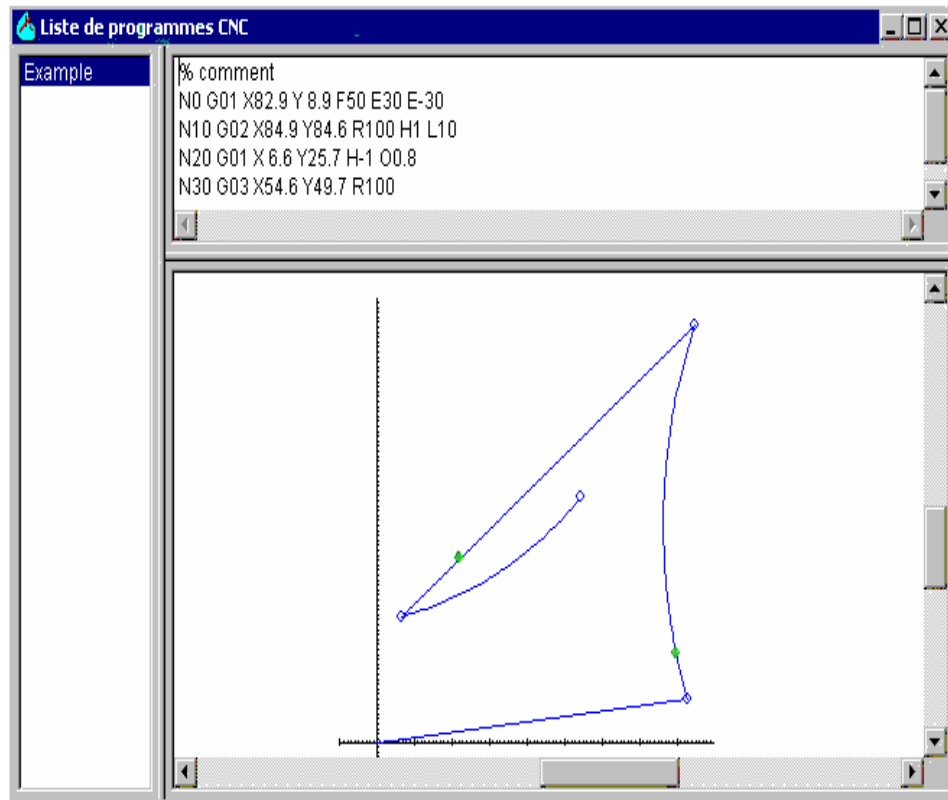
Comme l'exemple 1 et l'exemple 2 l'indiquent, vous avez deux possibilités pour compiler et appliquer des programmes CNC. Le premier exemple illustre la création directe d'une OutQueue, le second décode le programme en ligne en utilisant des variables. L'exemple 3 applique en plus en ligne un module de préparation de trajectoire.

11.7.1 Exemple CNC 1 : Création directe de OutQueue

(voir l'exemple de projet fourni avec SoftMotion CNCdirect.pro)

1. Création du programme NC dans l'éditeur CNC :

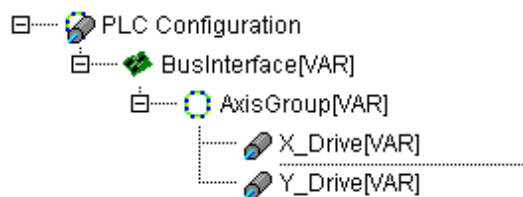
Comme décrit sous 0, on crée un exemple de programme qui se déroule entre x [0,100] et y [0,100]. On définit en outre des vitesses et accélérations pour la trajectoire et on définit deux points de coupure sur ladite trajectoire. Par exemple :



Le mode de compilation choisi est "Créer OutQueue en compilant".

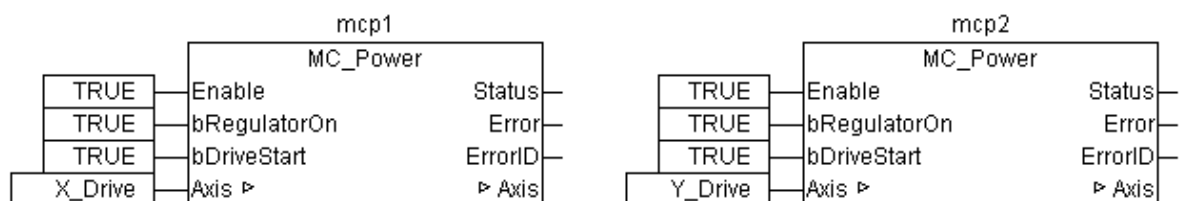
2. Interface d'entraînement, configuration de commande :

Une structure d'entraînement contenant deux entraînements linéaires est définie ; il faut également définir la vitesse maximale etc.

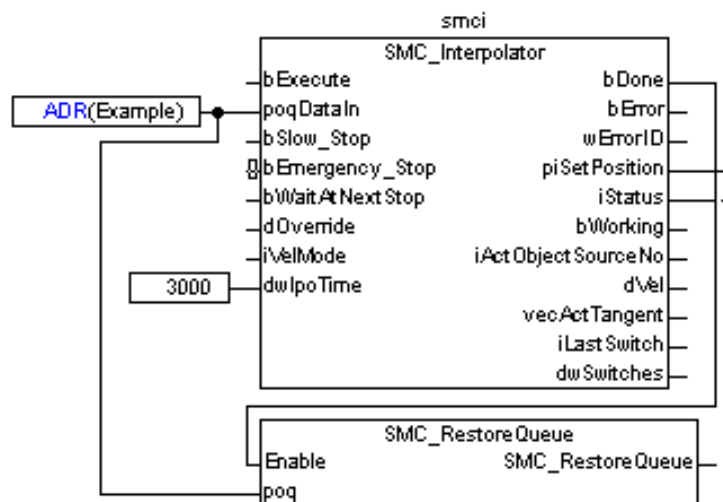


3. Création du programme IEC :

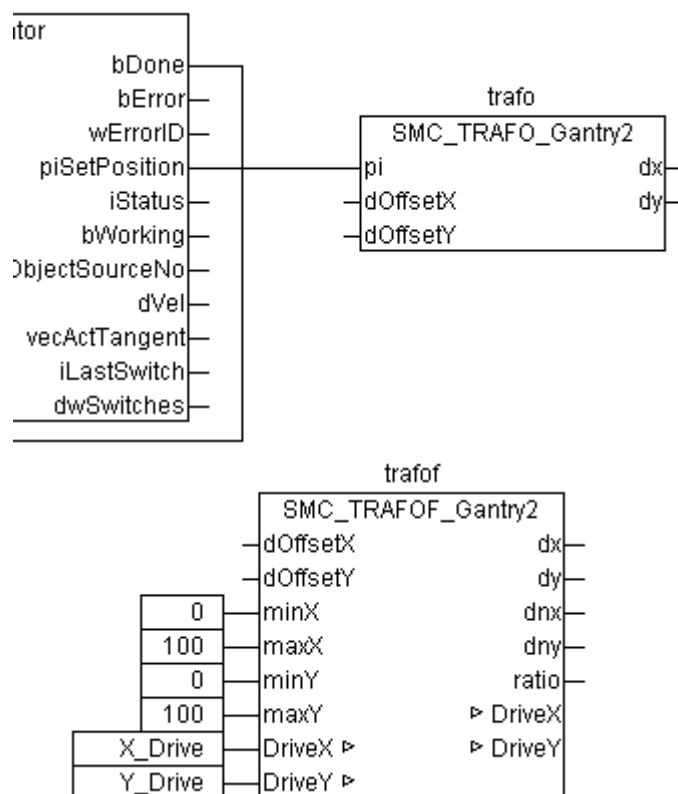
Il faut tout d'abord activer les entraînements avec le module MC_Power :



Un autre élément important est le module SMC_Interpolator. Ce dernier obtient comme entrée poqDataIn l'adresse du programme CNC créé. De plus, le temps de cycle de la tâche IEC doit être écrit dans dwlpoTime. À cet égard, il peut être saisi comme valeur constante pour l'entrée dwlpoTime, on utilise la variable dwCycle de la structure du groupe d'axes de la configuration de commande ; cette dernière possibilité s'avère avantageuse si le temps de cycle de la tâche change, la durée correcte est automatiquement utilisée comme entrée d'interpolateur.

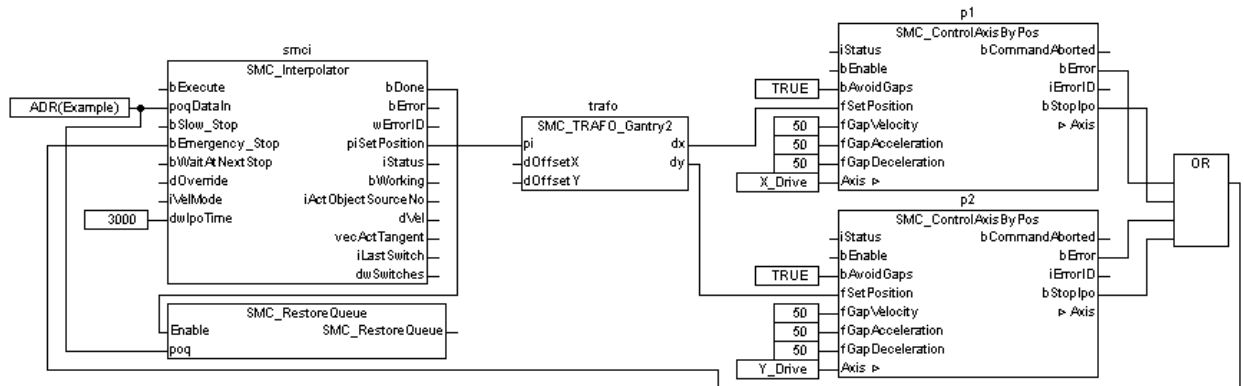


Dans l'exemple, on souhaite commander un système de portique. Il faut pour ce faire insérer une instance de modules de transformation marche arrière et marche avant provenant de la bibliothèque SM_Trafo.lib. Le module de transformation marche avant obtient comme entrées les entraînements (l'entraînement Z est affecté d'une variable fictive normalement inutilisée et de type AXIS_REF) ; le module de transformation marche arrière doit obtenir la position de consigne de l'interpolateur :



Les sorties du modules - les coordonnées des axes - doivent alors être écrites sur les entraînements. Utiliser pour ce faire les blocs fonctionnels SMC_ControlAxisByPos. Comme l'application ne peut pas

garantir que les sorties de l'interpolateur restent constantes, (par exemple, la trajectoire se termine sur un autre point que celui de départ), il est recommandé d'activer l'évitement de sauts (bAvoidGaps, fGapVelocity, fGapAcceleration, fGapDeceleration), de relier la sortie StopIpo avec celle bEmergency_Stop de l'interpolateur et de relier la sortie de l'interpolateur iStatus avec les entrées correspondantes des modules de contrôle d'axe.



Il faut veiller à respecter la succession correcte des modules lors de la programmation avec CFC.

4. Création d'une surface de service et de test :

Incorporer deux objets de visualisation dans une nouvelle visualisation. Le premier est le modèle d'interpolateur, le second est le modèle de transformation. Ceux-ci sont reliés avec les instances de modules ad hoc via des caractères de remplacement (ici : lpo.smci et lpo.trafof).

5. Mise en service :

Le programme ainsi créé peut être correctement compilé et lancé ; il exécute le programme CNC dès que l'entrée **Execute** de l'interpolateur est activée. Dès que le programme est complètement exécuté, on peut à nouveau le parcourir via un nouveau front montant.

Tenir également compte de la fonction des interrupteurs de course ; il sont également représentés dans la visualisation du module d'interpolation.

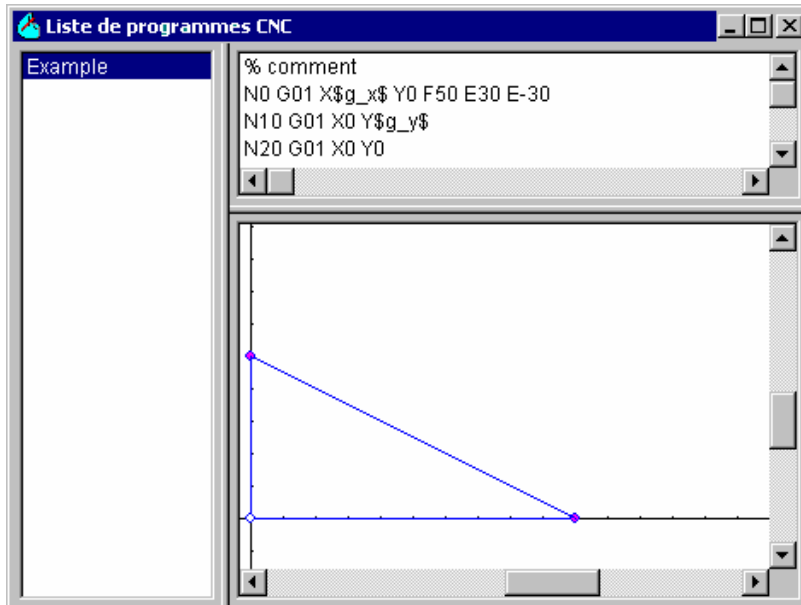
11.7.2 Exemple CNC 2 : Décodage en ligne à l'aide de variables

(voir l'exemple de projet fourni avec SoftMotion CNCOnline.pro)

1. Création du programme NC dans l'éditeur CNC :

Comme dans l'exemple précédent, création d'un programme CNC utilisant deux variables globales **g_x** et **g_y**. Par exemple :

```
VAR_GLOBAL
  g_x : REAL := 100 ;
  g_y : REAL := 50 ;
END_VAR
```



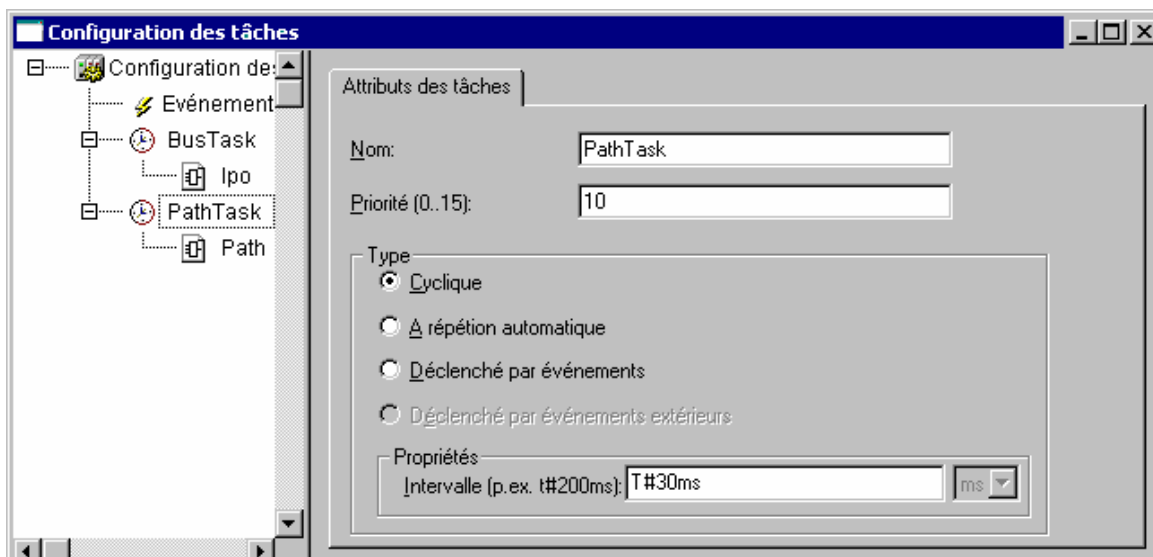
Le mode de compilation est cette fois ci „Créer variable programme en compilant“ puisque le programme utilise des variables.

2. Interface d'entraînement, configuration de commande :

Interface d'entraînement, configuration de commande :

3. Création du programme IEC :

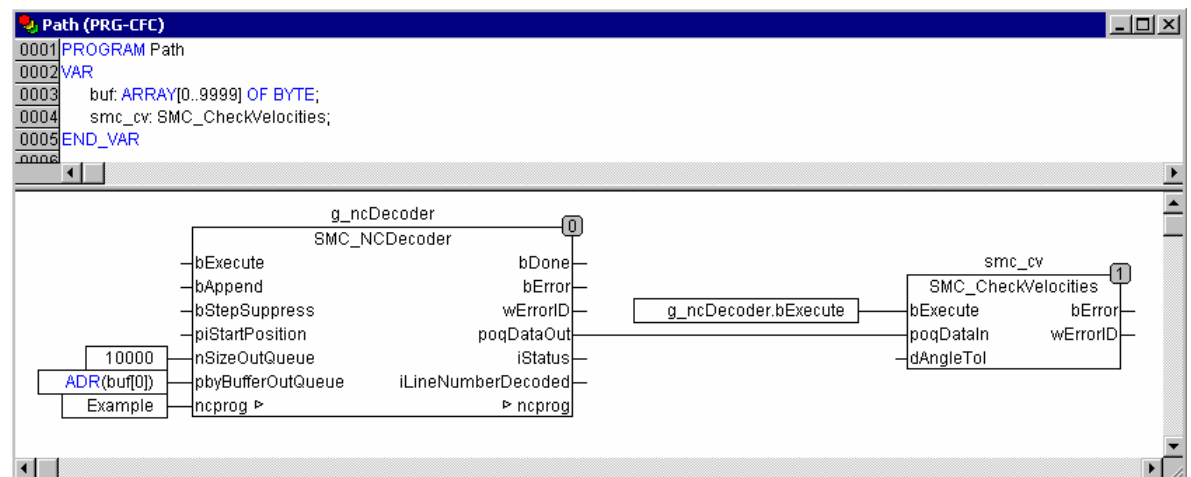
Comme le mode de compilation est différent, il faut procéder au décodage et à la préparation de trajectoire dans le programme IEC. Comme ce processus nécessitant un long temps d'exécution ne peut pas être exécuté à la cadence de l'interpolateur (un objet de trajectoire est créé par appel du décodeur, cet objet suffit normalement pour de nombreux appels d'interpolateur), il est souvent réservé à des tâches moins importantes et appelées plus rarement :



Le mécanisme de base est le suivant : pour une tâche plus lente, un objet GEOINFO est créé en début et par cycle, il est enregistré dans la structure OUTQUEUE du module de décodeur. Si cette mémoire OUTQUEUE est pleine, les modules de la tâche lente marquent une pause jusqu'à ce que l'OUTQUEUE ne soit plus pleine, ou encore jusqu'à ce que la tâche plus rapide ait exécuté le premier objet GEOINFO et supprimé ce dernier de la OUTQUEUE. Les modules de la tâche plus lente sont alors à nouveau activés et remplissent la structure OUTQUEUE. Avec la tâche rapide, un point de

trajectoire provenant de la structure OUTQUEUE et indiqué par l'entrée DataIn de l'interpolateur est calculé et exécuté par cycle. Comme un objet GEOINFO se compose en général de plusieurs points de trajectoire, il faut quelques cycles jusqu'à ce que le premier objet GEOINFO soit exécuté et automatiquement supprimé par l'interpolateur. Comme l'exécution d'un objet GEOINFO nécessite plus de cycles que sa création, la tâche lente est effectivement appelée plus rarement que la tâche rapide. La durée des tâches doit de ce fait être choisie de telle sorte que la dernière OUTQUEUE de la tâche lente contienne toujours suffisamment d'objets GEOINFO que pour ne pas entraîner de sous-exécution de données. Ceci se produit lorsqu'il n'y a plus d'objets GEOINFO à la disposition de l'interpolateur DataIn et que la fin de la trajectoire n'est pas encore atteinte. Dans un tel cas, l'interpolateur freine et reste à l'arrêt jusqu'à ce que des nouveaux éléments de données soient disponibles.

Le décodage du programme NC pour OUTQUEUE et le contrôle de vitesse ont lieu dans Programm Path :



La partie du programme IEC à interpoler reste quasi identique, sauf que les entrées de données de l'interpolateur ne correspondent pas comme précédemment au programme CNCnamen (ADR(exemple)), mais se composent de la sortie OutQueue des modules de préparation de trajectoire (g_CheckVel.poqDataOut).

4. Création d'une surface de service et de test :

Afin de visualiser l'exemple précédent, il est recommandé d'insérer des modèles de nouveaux modules (SMC_NCDecoder et SMC_CheckVelocities). En outre, on doit pouvoir modifier les variables globales *g_x* et *g_y* afin de pouvoir en contrôler le fonctionnement par après lors de la mise en service.

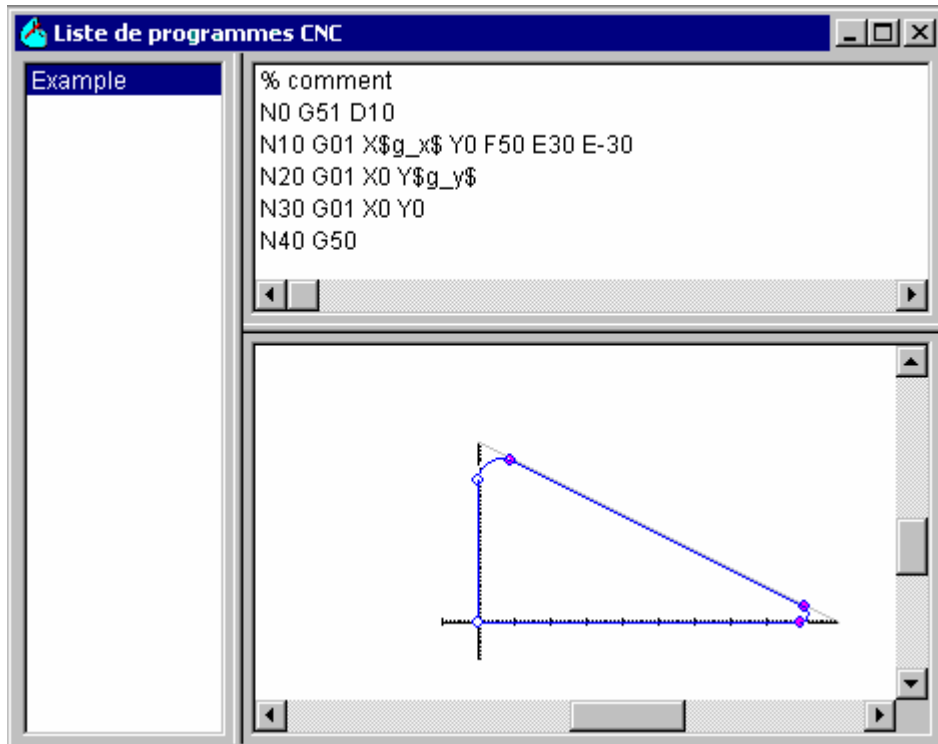
5. Mise en service :

Le programme ainsi créé peut être correctement compilé et lancé ; il exécute le programme CNC dès que les entrées Execute du décodeur et de l'interpolateur sont activées. Si on modifie les valeurs des variables globales, celles-ci sont reprises lors d'un redémarrage du décodeur et la trajectoire est modifiée en conséquence. Il faut également tenir compte de la fonction de l'entrée Append du décodeur.

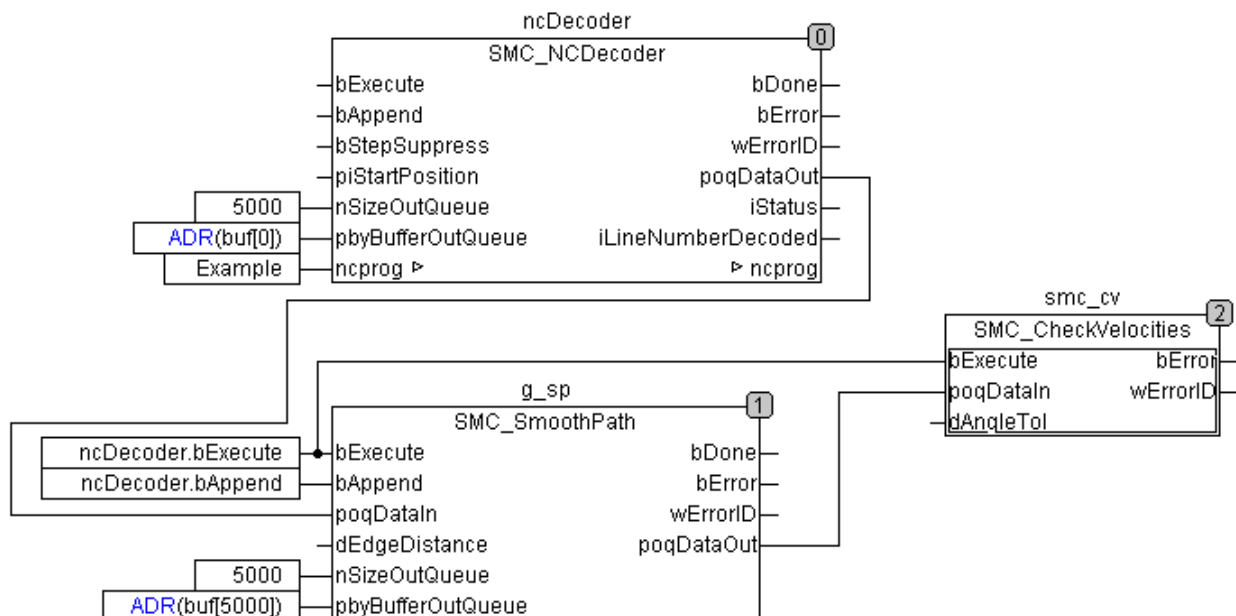
11.7.3 Exemple CNC 3 : Préparation de trajectoire en ligne

(voir à cet égard l'exemple de projet fourni avec SoftMotion [CNCprepro.pro](#))

L'exemple 2 doit être complété d'une préparation de trajectoire : les coins du dernier programme doivent être arrondis à l'aide de spline. Ceci est exécuté par le module SMC_SmoothPath. Le programme CNC doit être entouré des mots G51/G50, ce qui donne :



Si on n'utilise pas de variables, le programme peut être directement compilé sous cette forme en tant que queue et saisi directement dans l'interpolateur ; cependant, comme des variables sont utilisées, on doit effectuer soi-même le décodage et le ponçage des coins. C'est pourquoi il faut déclarer un nouveau module de type SMC_SmoothPath et l'appeler suite au décodeur :



L'entrée des données du module d'interpolateur doit comme toujours être réglée sur la sortie poqDataOut du module CheckVelocities.

Ce programme peut être correctement compilé et contrairement au précédent, il ne s'arrête plus aux coins du programme NC vu que les coins de la trajectoire ont été arrondis via la préparation de trajectoire.

11.8 Programmation dynamique SoftMotion

(Voir également le projet modèle fourni avec SoftMotion *CNCDynamicPath.pro*, reposant sur le fichier de configuration *softmotion.cfg*)

Un des avantages de SoftMotion est que tant le programmeur que l'utilisateur peuvent non seulement influencer sur l'exécution d'une trajectoire, mais qu'ils peuvent également générer ou modifier une trajectoire lorsque le Programme est en cours d'exécution. À cette fin, le programmeur doit seulement **remplacer le module de décodeur par un propre générateur de trajectoire** ; il peut malgré cela se servir de la préparation de trajectoire et surtout de l'interpolateur.

Pour remplacer le module de décodeur, il faut créer d'une autre manière **un objet structurel OUTQUEUE** rempli d'objets GEOINFO représentant la trajectoire souhaitée et qui est transmis au module ad hoc suivant (p.ex. l'interpolateur).

Étapes préparatoires :

- Dans la partie déclaration, il faut déclarer un objet structurel OUTQUEUE, un objet structurel GEOINFO ainsi qu'une mémoire tampon de la taille souhaitée :

```
QUEUE: SMC_OUTQUEUE;
BUF: ARRAY[0..49] OF SMC_GEOINFO;
GEO: SMC GEOINFO:=(dT1:=0, dT2:=1, dToolRadius:=0, dVel:=100, dVel_End:=100,
dAccel:=200, dDecel:=500, iObj_Nr:=0);
```

- La structure OUTQUEUE doit être initialisée dans une étape Init dans le corps de programme :

```
QUEUE.nSize := SIZEOF(BUF);
QUEUE.pbyBuffer := ADR(BUF);
```

Programmation dynamique de trajectoire

Pour chaque objet GEOINFO de la trajectoire, les étapes suivantes doivent être effectuées dans le corps du programme, au point où la trajectoire doit être créée :

- Définir la position de départ (premier objet)

```
GEO.piStartPos.dX := 0;
...
...ou copier à partir de l'objet précédent.
GEO.piStartPos := GEO.piDestPos;
```

- Définir le type de mouvement. Par exemple :

```
GEO.iMoveType := CCWL; ou
GEO.iMoveType := LIN;
```

- Définir les paramètres correspondant au type de mouvement. Si vous utilisez un arc de cercle (p.ex. CCWL), notez que les positions suivantes doivent être réglées (voir structure SMC_GEOINFO) :

```
GEO.dP1 := 200;
GEO.dP2 := 100;
GEO.dP3 := 50;
GEO.dT1 := 0;
GEO.dT2 := 90;
```

- Régler le cas échéant dans `InternMark` le bit de départ ou de fin pour la préparation de trajectoire (voir structure SMC_GEOINFO)

- Calculer la position de fin :

```
SMC_CalcEndPnt (ADR (GEO)) ;
```

- (Faire) calculer la longueur de l'objet :

```
SMC_CalcLengthGeo (ADR (GEO)) ;
```

- Enregistrer l'objet dans la mémoire OUTQUEUE :

```
SMC_AppendObj (POQ:=ADR (QUEUE), PGI:=ADR (GEO)) ;
```

- Si la trajectoire est complètement générée, la liste OUTQUEUE doit être clôturée :

```
QUEUE.bEndOfList := TRUE;
```

Il convient de noter que si la mémoire OUTQUEUE est pleine (si QUEUE.bFULL = TRUE), il ne faut pas tenter d'y annexer d'autres objets. La création de la trajectoire doit rester interrompue jusqu'à ce que le premier objet de la mémoire OUTQUEUE soit exécuté. C'est seulement à ce moment que l'on peut ajouter un autre objet. Si on veut éviter cela, la taille de la mémoire OUTQUEUE doit être choisie de telle sorte que tous les objets GEOINFO de la trajectoire souhaitée puissent y trouver place.

La liste d'objets Queue est tout d'abord transmise aux modules CheckVelocities puis à l'Interpolateur, ce dernier exécutant les objets.

Cet exemple montre également la manière de programmer une transformation cinématique qui n'est même pas comprise dans la bibliothèque SM_Trafo.lib mise à disposition par 3S. Les modules SMC_TRAFO et SMC_TRAFOF incorporés à ce projet le montrent sur base d'un exemple de système cartésien X/Y.

12 Index

A

- accélération 5-16, 5-23
- Accélération de trajectoire 6-11
- Accoupler CAM 5-21
- Activation de modules 5-1
- Adapter grandeur 3-13
- Afficher accélération 4-8
- Afficher extrêmes 4-7
- Afficher vitesse 4-8
- Afficher vitesse/accélération 4-1
- Analyse SMC_CNC_REF 7-1
- Analyse SMC_QutQueue 7-1
- Angle de rotation 3-10
- Angle RPY 8-16
- Arborescence CAM 4-3
- Arc de cercle 3-5
- Arrêter 5-5
- Arrondi 6-7, 6-9
- Arrondir coins 3-14, 6-9
- Arrondir les coins de trajectoire 6-7
- Axe
 - Arrêt 5-5
 - Contrôler position et vitesse 2-16
 - Contrôler vitesse 2-16
 - Couple 5-4
 - Force 5-4
 - Position 6-19
 - Positions de consigne 2-15
 - Puissance 5-4
 - Statut de décélération 5-4
- Axe 1-1
- Axe de temps 2-16
- Axe esclave 4-1
- Axe individuel
 - Exemple de programme 11-5
- Axe individuel 5-1
- Axe maître 4-1
- Axe rotatif 5-7
- Axe temps virtuel
 - Exemple de programme 11-7
- Axe temps virtuel 2-16
- AXIS_REF 2-22
- AXIS_REF 2-20
- Axisgroup
 - Configuration 2-2
- Axisgroup 2-2

B

- Bibliothèque
 - SM_CNC.lib 6-1
 - SM_Error.lib 9-1
 - SM_FileFBs.lib 10-1
 - SM_PLCopen.lib 5-1
 - SM_Trafo.lib 8-1
- Bibliothèque CNC 6-1, 6-19
- Bibliothèque CNC SoftMotion 6-1
- Boucle 6-6
- BusInterface 2-2
- BusInterface Drive library 2-21

C

- Calcul valeur de consigne 5-23
- Calcul valeur de consigne esclave 5-23
- CAM
 - Accoupler esclave et maître 5-21
 - Axe esclave 4-1
 - Axe maître 4-1
 - Communtation 4-12
 - Configuration 4-2
 - Éditer CAM 4-3
 - Périodique 4-2
 - Sélectionner 5-18
 - Séparer esclave et maître 5-20, 5-21
- CAM 1-1
- CAM alternées
 - Exemple de programme 11-8
- CAM data 4-13
- CAM générée manuellement 4-15
- CAM Phasing 5-22
- CAM structures 4-13
- CAM XYVA 4-15
- Came 4-4
- Came mécanique 5-25
- Cames comme tableau 4-1, 4-8
- CCLW 6-20
- Changer les valeurs epsilon pour zéro 3-15
- Charger fichier de l'automate 4-10
- Chronométrage de cycle 2-8
- Cinématique parallèle 8-13, 8-14
- CLW 6-20
- CNC - Exemple de programme
 - Programmation dynamique 11-15
- CNC - Exemple de programme 11-8
- CNC structure 3-2
- Code CNC comme tableau de chaînes 6-3
- Code CNC dans visualisation 6-3
- Code G 6-3
- Code visualisation 6-3
- Commande d'axes individuels 11-4
- commandes de déplacement 3-4
- Commandes de l'éditeur CNC 3-12
- Compléter courbe 4-6
- Composants SoftMotion PLCopen SoftMotion 1-1
- Concept SoftMotion 1-1
- Configuration BusInterface 2-2
- Configuration CAM 4-2
- Configuration de commande pour entraînements -
 - Exemple de programme 11-1
- Configuration de commande pour SoftMotion 2-2
- Configuration d'encodeur 2-7
- Configuration d'entraînement 2-4
- Configuration d'entraînement Sercos 2-4
- Contrôle d'axe 8-1
- Contrôle de vitesse 6-10
- Contrôle PackProfile 2-4
- Conversion pour entraînement 2-4
- Convertir CAM 5-19
- Correction de rayon d'outil 6-4
- Correction d'outil 6-4
- Correction du rayon d'outil 3-14
- Couple 5-4, 5-29

Couple de consigne 5-29
 courbe 4-4
 Courbe comme tableau 4-1, 4-8
 Course de référence 2-17, 5-5
 Course de référence d'entraînement 5-5
 Course de référence des axes 5-5
 Créer CAM 4-2
 Créer éditeur CAM 5-24
 Créer OutQueue en compilant 3-11
 Créer programme CNC 3-7
 Créer trajectoire sans boucle 6-6
 Créer variable de programme en compilant 3-11

D

Débit en bauds de groupe d'axes CAN 2-2
 Décalage de phase dans CAM 5-22
 Décalage d'outil 8-4
 Déclencheur 5-17
 Décodage en ligne 11-11
 Définir CAM pour SoftMotion 4-1
 Définir grandeur queue 3-9
 Définir position de démarrage 3-9
 Définir position de démarrage dans le programme CNC 3-9
 Définir tolérance d'angle pour les arrêts 3-10
 Définitions d'erreurs SoftMotion 9-1
 Déplacer point zéro 5-16
 Déplacer programme 3-10
 Déplacer programme CNC 3-10
 Description de variables 10-2
 Diagnostic accélération et décélération 2-19
 Diagnostic CNC 7-1
 Diagnostic d'axe 10-4
 Diagnostic de vitesse 2-19
 Diagnostic erreur de poursuite 2-19
 Diagnostic vitesse 2-19
 Dialogue de configuration d'entraînement 2-4
 DIN 66025 3-2
 DIN66025 3-1
 Distance maître-esclave 5-22
 Diviser l'objet 3-10
 Diviser l'objet CNC 3-10
 Données CAM 4-15
 Données de communication 2-4
 Drive 2-2, 2-4
 Drive ID 2-4
 Drive Interface 2-1, 3-1
 Drive Interface SoftMotion 2-1
 DriveInterface 1-1
 Durée de déplacement 3-9

E

Écrire axe dans fichier 10-4
 Écrire CAM dans un fichier 4-8
 Écrire fichier dans l'automate 4-10
 Écrire OutQueue dans fichier 3-11
 Écrire positions de consigne 2-15
 Écrire programme CNC dans fichier 3-11
 Écriture des paramètres d'entraînement 5-3
 Éditeur CAM
 Afficher extrêmes 4-7
 Afficher vitesse/accélération 4-1, 4-8
 Arborescence CAM 4-3
 Cames comme tableau 4-8
 Compléter courbe 4-6

Configuration générale 4-3
 Courbe comme tableau 4-8
 Définition d'une CAM pour SoftMotion 4-1
 Écrire CAM dans un fichier 4-8
 Éditer CAM 4-3
 Extras
 Configuration 4-6
 Insérer
 Sélectionner élément 4-8
 Insérer came 4-9
 Insérer ligne droite 4-9
 Insérer point 4-9
 Lire CAM du fichier 4-8
 Nouvelle CAM 4-2
 Options de compilation 4-7
 propriétés des éléments de courbe 4-4
 Éditeur CAM 4-1
 Éditeur CAM - Démarrage 4-1
 Éditeur CAM en mode en ligne 4-9
 Éditeur CNC
 Adapter grandeur 3-13
 Appel 3-7
 Arrondir coins 3-14
 Changer les valeurs epsilon pour zéro 3-15
 Correction du rayon d'outil 3-14
 Créer OutQueue en compilant 3-11, 3-16
 Créer programme 3-7
 Créer variable de programme en compilant 3-11, 3-16
 Définir grandeur queue 3-9
 Définir position de démarrage 3-9
 Définir tolérance d'angle pour les arrêts 3-10
 Déplacer programme 3-10
 Diviser l'objet 3-10
 Écrire OutQueue dans fichier 3-11
 Éditeur graphique 3-12
 Éditeur littéral 3-7
 Effacer programme CNC 3-9
 Étendre programme 3-10
 Extras
 Éviter les boucles 3-15
 Importer un fichier DXF 3-11
 Info 3-9
 Inverser la direction 3-10
 Mode d'insertion pour les cercles à droite 3-15
 Mode d'insertion pour les cercles à gauche 3-16
 Mode d'insertion pour les lignes 3-15
 Montrer grille 3-14
 Montrer interpolation 3-15
 Ne pas compiler 3-11, 3-16
 Nouvelle numérotation du programme 3-14
 Pivoter programme 3-10
 Poncer coins 3-14
 Remplacer les splines et ellipses par des droites 3-14
 Remplissage automatique de structure 3-16
 Renommer programme CNC 3-8
 Sélectionner 3-15
 Supprimer le jeu 3-15
 Éditeur CNC 1-1, 3-1
 Éditeur CNC 3-14
 Éditeur CNC.lib
 MC_GearOut 5-21
 Éditeur graphique 3-12
 Effacer 3-9
 Effacer programme CNC 3-9
 Ellipse 3-6
 Encoder 2-20
 Enregistrement de position 6-19

Enregistrement de vecteur 6-22, 6-23
 Enregistrer position d'axe 5-17
 Enregistrer position d'entraînement 5-17
 entraînement
 Arrêt 5-5
 Couple 5-4
 Force 5-4
 Puissance 5-4
 Statut de décélération 5-4
 entraînement 2-2, 2-4
 Entraînement CAN 2-4
 Entraînement SERCOS 2-4
 Entraînements d'axes 5-1
 Entrée de déclenchement 5-17
 Erreur de module 9-1
 Erreur de poursuite 2-19, 8-4
 Erreur d'entraînement 9-1
 Erreur SoftMotion 9-1
 Étendre programme 3-10
 Étendre programme CNC 3-10
 Éviter les boucles 6-6
 Exemple CAM 4-15
 Exemple de CAM 4-15
 Exemple de configuration Drive Interface 11-1
 Exportation d'une CAM 4-8
 Exporter CAM comme tableau ASCII 4-8
 Extras
 Adapter grandeur 3-13
 Afficher extrêmes 4-7
 Afficher vitesse/accélération 4-8
 Arrondir coins 3-14
 Cames comme tableau 4-8
 Changer les valeurs epsilon pour zéro 3-15
 Compléter courbe 4-6
 Configuration 4-2
 Correction du rayon d'outil 3-14
 Courbe comme tableau 4-8
 Écrire CAM dans un fichier 4-8
 Exporter CAM comme tableau ASCII 4-8
 Extras
 Éviter les boucles 3-15
 Importer CAM du tableau ASCII 4-8
 Lire CAM du fichier 4-8
 Mode d'insertion pour les cercles à droite 3-15
 Mode d'insertion pour les cercles à gauche 3-16
 Mode d'insertion pour les lignes 3-15
 Montrer grille 3-14
 Montrer interpolation 3-15
 Nouvelle CAM 4-2
 Nouvelle numérotation du programme 3-14
 Options de compilation 4-7
 Poncer coins 3-14
 Propriétés CAM 4-6
 Remplacer les splines et ellipses par des droites 3-14
 Sélectionner 3-15
 Supprimer le jeu 3-15
 Extras
 Éviter les boucles 3-15

F

Facteur d'extension 3-10
 FErrorOccured 9-1
 Fichier ASCII dans CNC 3-11
 Fichier de diagnostic pour axe 10-4
 Fichier pour CAM 10-3
 Fichier texte dans CNC 3-11

Fichiers code G 6-25
 Fin de course logicielle pour entraînements 2-4
 Fonction auxiliaire 3-5
 Fonction de point de commutation 3-4
 Fonction H 3-4
 Fonction M 3-5, 6-16
 Fonctionnalité de fichier 10-1
 Fonctionnalité de variable pour programme NC 10-2
 Force 5-4

G

G00..G99 3-4
 GantryCutter 8-3
 gc_SMC_FILE_MAXCAMEL 10-3
 gc_SMC_FILE_MAXCAMTAP 10-3
 Générateur SYNC 2-2
 Générer CAM manuellement 4-15
 GEOINFO 3-1, 6-1, 6-20, 6-23
 Groupe d'axes
 CAN 2-2
 Configuration 2-2
 SERCOS 2-2
 Groupe d'axes 2-2
 Groupe d'axes 2-2
 Groupe d'axes CAN
 Réglages spécifiques 2-2
 Groupe d'axes CAN 2-2

I

ID d'entraînement 2-4
 identificateur de mot 3-2, 3-3
 Importation d'une CAM 4-8
 Importer CAM du tableau ASCII 4-8
 Importer fichier DXF 3-11
 Importer fichier DXF dans l'éditeur CNC 3-11
 Info 3-9
 Informations sur le programme CNC 3-9
 Insérer
 Insérer came 4-9
 Insérer ligne droite 4-9
 Insérer point 4-9
 Insérer came 4-9
 Insérer ligne droite 4-9
 insérer point 4-9
 Intensité des LED 2-2
 Interpolateur 6-17
 Interpolation 6-12
 Interpolation circulaire 3-5
 Interpolation de spline 3-6
 Interpolation d'ellipse 3-6
 Interrupteur de puissance 5-4
 Interrupteur de référencement 2-17
 Inverser la direction 3-10

J

Jerk 2-4

L

langage DIN 66025 3-2
 Lecture d'erreur 5-2
 Lecture des paramètres d'entraînement 5-3
 Lecture position de consigne 5-29

ligne 4-4
 Ligne de programme CNC 3-9
 LinDrive 2-20
 LinDrive_V 2-20
 Lire CAM à partir de fichier 10-3
 Lire CAM du fichier 4-8
 Lire et écrire paramètres CAN 2-21
 Lire et écrire paramètres Sercos 2-21
 Lire fichier programme NC 10-2
 Lire position 5-3
 Lire programme CNC du fichier 3-11
 Liste d'objets 3-9
 Lire fichier OutQueue 10-1
 Longueur totale de trajectoire 3-9

M

Marche / arrêt de l'entraînement 5-4
 MasterAbsolute 4-11
 MasterOffset 4-11
 MasterScaling 4-11
 MC_AbortTrigger 5-17
 MC_AccelerationProfile 5-16
 MC_CAM_REF 4-14
 MC_CamIn 4-11
 MC_CamIn 5-19
 MC_CamOut 5-20
 MC_CamTableSelect 5-18
 MC_CAMTableSelect 4-11
 MC_DigitalCamSwitch 5-27
 MC_GearIn 5-21
 MC_GearOut 5-21
 MC_Halt 5-6
 MC_Home 5-5
 MC_MoveAbsolute 5-7
 MC_MoveAdditive 5-9
 MC_MoveRelative 5-10
 MC_MoveSuperImposed 5-12
 MC_MoveVelocity 5-13
 MC_Phasing 5-22
 MC_PositionProfile 5-14
 MC_Power 5-4
 MC_ReadActualPosition 5-3
 MC_ReadActualTorque 5-4
 MC_ReadActualVelocity 5-4
 MC_ReadAxisError 5-2
 MC_ReadBoolParameter 5-3
 MC_ReadParameter 5-3
 MC_ReadStatus 5-2
 MC_Reset 5-2
 MC_SetPosition 5-16
 MC_Stop 5-5
 MC_TA_REF 5-16
 MC_TouchProbe 5-17
 MC_TP_REF 5-14
 MC_TV_REF 5-15
 MC_VelocityProfile 5-15
 MC_WriteBoolParameter 5-3
 MC_WriteParameter 5-3
 Mode de compilation d'éléments optimisés 4-7
 Mode de compilation équidistant 4-7
 Mode de compilation polynomiale 4-7
 Mode de sélection 3-12
 Mode de vitesse 2-4
 Mode de vitesse parabolique 2-4
 Mode de vitesse sigmoïdal 2-4
 Mode de vitesse trapézoïdal 2-4

Mode d'insertion de lignes dans l'éditeur CNC 3-15
 Mode d'insertion pour les cercles à droite 3-15
 Mode d'insertion pour les cercles à gauche 3-16
 Mode Insertion 3-12
 Modèle de visualisation 11-5
 Modes de sélection dans l'éditeur CNC 3-15
 Modes de traitement dans l'éditeur CNC 3-12
 Modifier type d'entraînement 2-10
 Modifier valeurs de variables 3-7
 Module de décodeur 6-1
 Module d'interpolateur 6-12
 Modules de groupes d'axes
 SMC_DisableAllDrives 2-10
 SMC_EnableAllDrives 2-10
 SMC_GetAxisGroupState 2-9
 SMC_IsAxisGroupREady 2-8
 SMC_ResetAxisGroup 2-9
 Modules de groupes d'axes 2-8
 Modules de transformation 8-1
 Modules mathématiques 2-8
 Montrer grille 3-14
 Montrer interpolation 3-15
 mot 3-2
 Motion 1-1
 Mots code G 6-26
 Mouvement absolu d'axe 5-7
 Mouvement absolu d'entraînement 5-7
 Mouvement additionnel d'axe 5-9
 Mouvement additionnel d'entraînement 5-9
 Mouvement relatif d'axe 5-10
 Mouvement relatif d'entraînement 5-10
 Mouvement supplémentaire d'axe 5-12
 Mouvement supplémentaire d'entraînement 5-12
 Mouvements des axes
 absolu 5-7
 additionnel 5-9
 relatif 5-10
 supplémentaire 5-12
 Multi-axe 5-1

N

N° de contrôleur 2-2
 Ne pas compiler 3-11
 Nom de programme CNC 3-7, 3-9
 Nouveau programme CNC 3-7
 Nouvelle CAM 4-2
 Nouvelle numérotation de programme 3-14
 numéro de jeu 3-2
 Numéro de ligne 7-1
 Numéro d'objet 7-1
 Numéros d'erreur 9-1

O

Objet de trajectoire 3-10, 6-19, 6-20
 Options de compilation CAM 4-7
 Options de l'éditeur CNC 3-12
 Options d'importation DXF 3-11
 OUTQUEUE 3-1, 6-19, 11-9
 OutQueue dans le fichier 3-11

P

PackProfile 2-4
 Parallélépipède

- Définition de position 8-16
- Paramètres de module 2-2, 2-4
- Paramètres d'entraînement 5-3
- Période 4-10
- Périodique 4-10, 4-11
- Périodique 4-2
- Pivoter programme 3-10
- Pivoter programme CNC 3-10
- Plage modulo pour entraînements rotatifs 2-4
- PLCopen
 - SM_PLCopen.lib 5-1
 - Spécification 5-1
- point 4-4
- Point d'appui 4-7
- Point zéro d'axe 5-16
- Point zéro d'entraînement 5-16
- Points de trajectoire 6-12
- Poncer coins 3-14, 6-7
- Position 2-12, 2-13, 2-15, 2-16, 5-16, 5-17, 5-23, 5-29
- Position d'axe 2-12, 2-13, 5-3
- Position de consigne 5-23
- Position de consigne d'axe 5-23
- Position de démarrage 3-9, 7-1
- Position de fin 7-1
- Position d'entraînement 5-3, 5-17, 5-29
- Préparation de trajectoire 6-7
- Préparation de trajectoire en ligne 11-14
- Profil de position 5-14
- Profil de vitesse 5-15
- Programmation dynamique
 - Exemple de programme 11-15
- Programme CNC
 - Convertir en objets structuraux SMC_GEOINFO 6-1
 - Créer OutQueue en compilant 3-11
 - Créer variable de programme en compilant 3-11
 - Définir grandeur queue 3-9
 - Définir position de démarrage 3-9
 - Définir tolérance d'angle pour les arrêts 3-10
 - Déplacer programme 3-10
 - Diviser l'objet 3-10
 - Écrire OutQueue dans fichier 3-11
 - Effacer 3-9
 - Étendre programme 3-10
 - Importer un fichier DXF 3-11
 - Info 3-9
 - Inverser la direction 3-10
 - Ne pas compiler 3-11
 - Nouveau programme CNC 3-7
 - Pivoter programme 3-10
 - Renommer programme CNC 3-8
- Programme NC 3-7
- Programme NC à partir de fichier 10-2
- Propriétés CAM 4-10
- Propriétés CAM 4-6
- Propriétés des éléments de courbe 4-4
- Puissance de l'entraînement 5-4

R

- Rapport de transmission 2-10
- Réglages dans l'éditeur CAM 4-6
- Réglages Sercos 2-2
- Réinitialisation du statut d'axe 5-2
- Réinitialiser statut d'axe 5-2
- Remplacer les splines et ellipses par des droites 3-14
- Remplissage de structure dans l'éditeur CNC 3-16
- Renommer 3-8

- Renommer programme CNC 3-8
- Représenter objets GEOINFO 7-1
- Représenter programme NC 7-1
- Rotation 6-18
- Rotation de trajectoire 6-18
- RotDrive 2-20

S

- Saisie de profil pour déplacement 5-14
- Saut d'accélération 6-11
- Sauts conditionnels 3-7
- Scruter paramètres de fonction M 6-16
- Section de trajectoire 6-19, 6-20
- Sélectionner CAM 5-18
- Séparer CAM 5-20, 5-21
- SercosDrive.lib 2-21
- SlaveAbsolute 4-11
- SlaveOffset 4-11
- SlaveScaling 4-11
- SM_CAN.lib 2-21
- SM_CNC.lib
 - OUTQUEUE 6-23
 - Paramétrage via variables globales 6-18
 - SMC_AvoidLoop 6-6
 - SMC_GCodeViewer 6-3
 - SMC_GEOINFO 6-19, 6-20
 - SMC_GetMParameters 6-16
 - SMC_Interpolator 6-12
 - SMC_Interpolator2Dir 6-17
 - SMC_NCDECODER 6-1
 - SMC_POSINFO 6-19
 - SMC_ROTATEQUEUE2D_2D 6-18
 - SMC_RoundPath 6-9
 - SMC_SmoothPath 6-7
 - SMC_ToolCorr 6-4
 - SMC_TRANSLATEQUEUE3D_2D 6-18
 - SMC_VECTOR3D 6-19, 6-22
 - SMC_VECTOR6D 6-23
- SM_CNC.lib 1-1, 6-1
- SM_CNCDiagnostic.lib
 - SMC_ShowCNCREF 7-1
 - SMC_ShowQueue 7-1
- SM_Drive_Basic.lib 2-1
- SM_DriveBasic.lib
 - Modules de configuration 2-10
 - Modules de diagnostic 2-19
 - Modules de groupes d'axes 2-8
 - Modules de régulation 2-11
 - Modules mathématiques 2-8
 - Référencement via entrées numériques 2-17
- SM_DriveBasic.lib 2-7
- SM_Error.lib
 - SMC_Error 9-1
- SM_Error.lib 9-1
- SM_FileFBs.lib
 - SMC_ReadCAM 10-3
 - Structure SMC_VARLIST 10-2
- SM_FileFBs.lib 1-1, 10-1
- SM_PLCopen.lib 5-1
- SM_Trafo.lib 1-1, 8-1
- SMC_APPENDOBJ 6-23
- SMC_AvoidLoop 6-6
- SMC_AxisDiagnosticLog 10-4
- SMC_CAMBounds 5-23
- SMC_CAMEditor 5-24
- SMC_CAMRegister 5-25

- SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_1 4-15
- SMC_CAMTable_<Type de variable>_<Nombre d'éléments>_2 4-15
- SMC_CAMVisu 5-24
- SMC_CAMXYVA 4-15
- SMC_ChangeGearingRatio 2-10
- SMC_CheckLimits 2-14
- SMC_CheckVelocities 6-10
- SMC_ClearFBError 9-1
- SMC_CNC_REF 6-25
- SMC_ControlAxisByPos 2-15
- SMC_ControlAxisByPosVel 2-16
- SMC_ControlAxisByVel 2-16
- SMC_CoordinateTransformation3D 8-15
- SMC_DELETEOBJ 6-23
- SMC_DetermineCuboidBearing 8-16
- SMC_DisableAllDrives 2-10
- SMC_EnableAllDrives 2-10
- SMC_Encoder 2-7, 2-20
- SMC_Encoder_Ref 2-20
- SMC_Error 9-1
- SMC_ErrorString 9-1
- SMC_File_MaxCamEI 10-3
- SMC_File_MaxCamTap 10-3
- SMC_FollowPosition 2-12
- SMC_FollowPositionVelocity 2-13
- SMC_FollowVelocity 2-14
- SMC_GCODE_WORD 6-26
- SMC_GCodeViewer 6-3
- SMC_GearInPos 5-21
- SMC_GEOINFO 6-19, 6-20
- SMC_GEOINFO GETOBJ 6-23
- SMC_GEOINFO GETOBJFROMEND 6-23
- SMC_GetAxisGroupState 2-9
- SMC_GetCamSlaveSetPosition 5-23
- SMC_GETCOUNT 6-23
- SMC_GetMaxSetAccDec 2-19
- SMC_GetMaxSetVelocity 2-19
- SMC_GetMParameters 3-5, 6-16
- SMC_GetTappetValue 5-26
- SMC_GetTrackingError 2-19
- SMC_Homing 2-17
- SMC_Interpolator 6-12
- SMC_Interpolator2Dir 6-17
- SMC_Interpolator2Dir_SLOWTASK 6-17
- SMC_IsAxisGroupREady 2-8
- SMC_LimitCircularVelocities 6-11
- SMC_MoveContinuousAbsolute 5-8
- SMC_NCDECODER 6-1
- SMC_OUTQUEUE 6-23
- SMC_POSINFO 6-19
- SMC_ReadCAM 10-3
- SMC_ReadCANParameter 2-21
- SMC_ReadFBError 9-1
- SMC_ReadNCFile 10-2
- SMC_ReadNCQueue 10-1
- SMC_ReadSetPosition 5-29
- SMC_ResetAxisGroup 2-9
- SMC_RESTOREQUEUE 6-23
- SMC_ROTATEQUEUE2D_2D 6-18
- SMC_RoundPath 6-9
- SMC_SetControllerMode 2-11
- SMC_SetTorque 5-29
- SMC_ShowCNCREF 7-1
- SMC_ShowQueue 7-1
- SMC_SingleVar 10-2
- SMC_SmoothPath 6-7
- SMC_TimeAxisFB 2-16
- SMC_ToolCorr 6-4
- SMC_TRAFO_Gantry2 8-1
- SMC_TRAFO_Gantry2Tool1 8-5
- SMC_TRAFO_Gantry2Tool2 8-6
- SMC_TRAFO_Gantry3 8-2
- SMC_TRAFO_GantryH2 8-8
- SMC_TRAFO_Scara2 8-10
- SMC_TRAFO_Scara3 8-11
- SMC_TRAFO_Tripod 8-13
- SMC_TRAFO<n>_Gantry<n> 8-3
- SMC_TRAFOF_Gantry2 8-2
- SMC_TRAFOF_Gantry2Tool1 8-6
- SMC_TRAFOF_Gantry2Tool2 8-7
- SMC_TRAFOF_Gantry3 8-3
- SMC_TRAFOF_GantryH2 8-9
- SMC_TRAFOF_Scara2 8-10
- SMC_TRAFOF_Scara3 8-12
- SMC_TRAFOF_Tripod 8-14
- SMC_TRAFOV_Gantry 8-4
- SMC_TRANSLATEQUEUE3D_2D 6-18
- SMC_UnitVectorToRPY 8-16
- SMC_VARLIST 10-2
- SMC_VECTOR3D 6-19, 6-22
- SMC_VECTOR6D 6-23
- SMC_WriteCAM 10-4
- SMC_WriteCANParameter 2-21
- SMC_WriteDriveParamsToFile 2-28
- SMCCalcDirectionFromVector 8-5
- Smooth Path 6-7
- SoftMotion_CNC_Globals 6-18
- Spécification PLCopen 5-1
- Spline 3-6
- SPLINE 6-20
- StartMode 4-12
- Statut d'axes 5-2
- Statut de came 5-26
- Statut de décélération d'entraînement 5-4
- Statut d'une came 5-26
- Structure AXIS_REF 2-22
- Structure d'axe 2-22
- Structure de données CAM
- MC_CAM_REF 4-14
 - SMC_CAMXYVA 4-15
- Structure de données pour CAM
- MC_CAM_REF 4-14
 - SMC_CAMXYVA 4-15
- Structure OUTQUEUE 6-23
- Structure SMC_OUTQUEUE 6-23
- Structures CAM 4-13
- Supprimer le jeu 3-15
- Surveiller positions de consigne 2-15
- Système à portique 8-1, 8-2, 8-3, 8-4, 8-5, 8-6, 8-7, 8-8, 8-9
- Système de coordonnées 8-15, 8-16
- Système Scara 8-9, 8-10, 8-11, 8-12
- Systèmes Scara à double articulation 8-9

T

- Tableau de points d'appui 4-7
- Tableau de points d'appui équidistants 4-7
- Tableau de points d'appui optimisé en éléments 4-7
- Tâche Motion 2-2
- Taille d'une CAM 10-3
- Temps de cycle 2-2

Tolérance d'angle pour les arrêts 3-10
 Trajectoire décalée 6-4
 Transformation
 Système de coordonnées 8-15, 8-16
 Translation 6-18
 Translation de trajectoire 6-18
 Type de rampe de vitesse 2-4
 Type d'entraînement 2-4
 Type d'objet 7-1

V

Valeur de consigne 5-23
 Valeur epsilon pour contrôle de point zéro 6-18
 Valeurs de consigne esclave CAM 5-23
 Variables dans l'éditeur CNC 3-5

Variables globales pour réglages CNC 6-18
 Vecteur de déplacement 3-10
 Vecteurs d'unité 8-15
 Verrouillage 5-17
 VISU_SMC
 ShowCNCRef 7-1
 VISU_SMC_ShowQueue 7-1
 visualisation
 modèle la bibliothèque SM_DriveBasic.lib 2-20
 vitesse 2-13, 2-14, 2-16, 5-4, 5-15, 5-23, 6-11
 Vitesse d'axe 2-13, 2-14, 5-4, 5-13
 Vitesse de consigne 7-1
 Vitesse d'entraînement 5-4, 5-13
 Vitesse des axes 5-13
 Vitesse finale 7-1