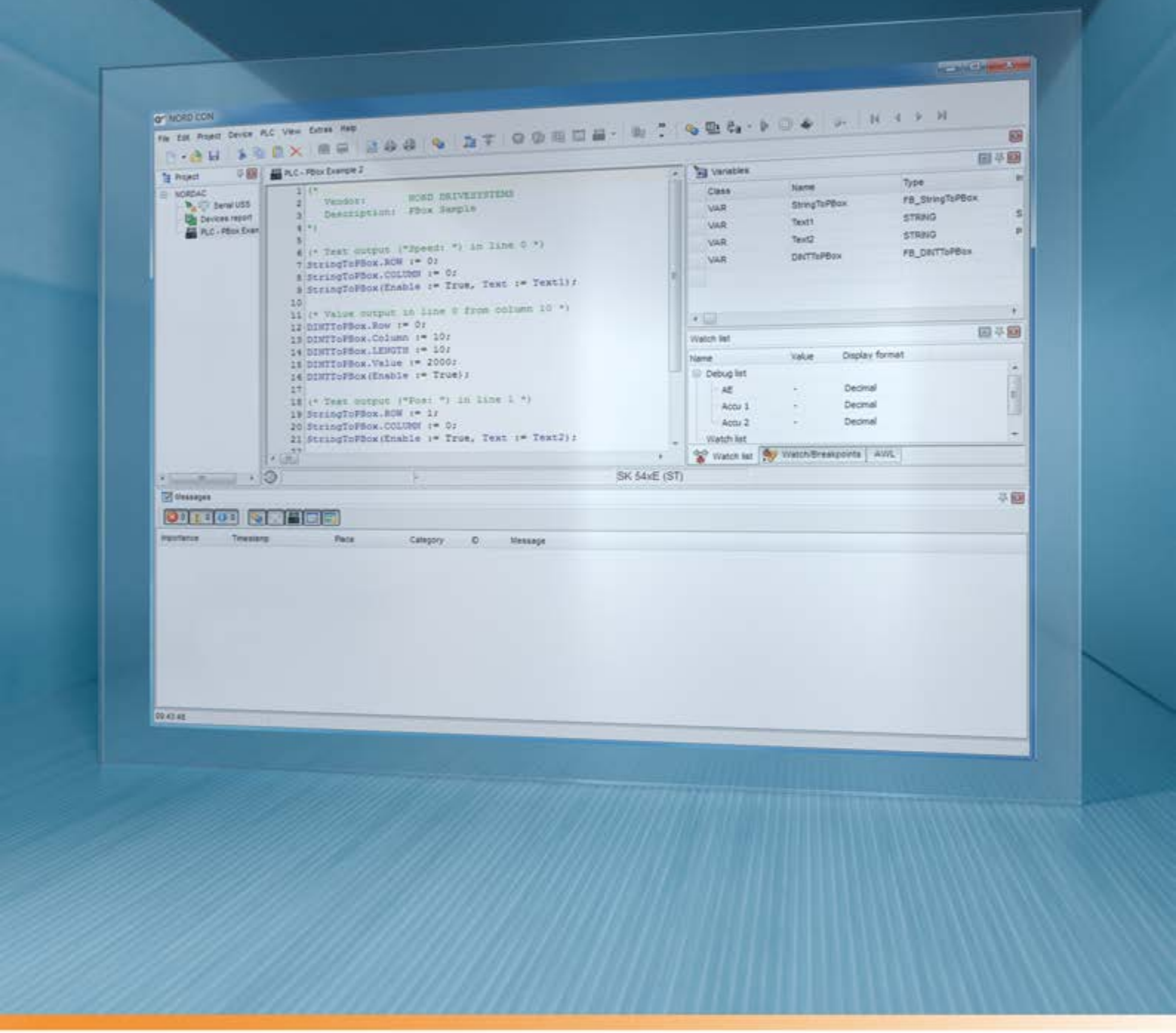
INTELLIGENT DRIVESYSTEMS, WORLDWIDE SERVICES



BU 0550 – fr

Fonctions PLC

Manuel supplémentaire pour les appareils NORDAC

NORD DRIVESYSTEMS

Sommaire

1 Introduction

1.1 Généralités

1.1.1 Documentation

Désignation : **BU 0550**

Numéro d'article : **6075504**

Série : **Fonctions PLC pour variateurs de fréquence et démarreurs des séries**

NORDAC *PRO* (SK 500P … SK 550P)
(SK 520E … SK 545E)

NORDAC *Flex* (SK 200E … SK 235E)

NORDAC *Base* (SK 180E / SK 190E)

NORDAC *Link* (SK 250E-FDS … SK 280E-FDS)

NORDAC *Link* (SK 155E-FDS / SK 175E-FDS)

1.1.2 Historique du document

Édition Numéro de commande	Série	Version Logiciel	Remarques
BU 0550, Septembre 2011 **6075504**/ 3911	SK 540E … SK 545E	V 2.0 R0	Première édition
Autres révisions : Octobre, 2011, février 2013, février 2017 Une vue d'ensemble des modifications des versions susmentionnées est disponible dans le document correspondant.			
BU 0550, Mai 2019 6075504/ 1919	SK 500P … SK 550P SK 540E … SK 545E SK 520E … SK 535E SK 200E … SK 235E SK 180E / SK 190E SK 250E-FDS … SK 280E-FDS SK 155E-FDS / SK 175E-FDS	V 1.0 R1 V 2.4 R0 V 3.2 R0 V 2.2 R0 V 1.3 R0 V 1.3 R0 V 1.2 R0	• Implémentation des types d'appareils NORDAC *PRO* SK 500P … SK 550P • Adaptations et corrections

1.1.3 Mention de droit d'auteur

1.1.4 Éditeur

Getriebebau NORD GmbH & Co. KG

Getriebebau-Nord-Straße 1

D-22941 Bargteheide, Allemagne

http://www.nord.com/

Tél. +49 (0) 45 32 / 289-0

Fax +49 (0) 45 32 / 289-2253

1.1.5 À propos de ce manuel

Ce manuel a pour but de vous aider à mettre en service les fonctions PLC d'un variateur de fréquence ou d'un démarreur du fabricant Getriebebau NORD GmbH & Co. KG (soit : NORD). Il s'adresse aux électriciens qui conçoivent, planifient, installent et configurent les programmes PLC (📖 chapitre 2.1 "Recrutement et qualification du personnel"). Les informations contenues dans ce manuel impliquent que les électriciens auxquels le travail est confié soient familiarisés avec les techniques de pilotage des entraînements, en particulier avec les appareils NORD.

Ce manuel contient exclusivement des informations et des descriptions des fonctions PLC, ainsi que des informations supplémentaires liées aux fonctions PLC pour l'appareil de Getriebebau NORD GmbH & Co. KG.

1.2 Documents complémentaires

Ce manuel est uniquement valable en combinaison avec le mode d'emploi de l'appareil utilisé. Toutes les informations requises pour une mise en service sûre de l'entraînement sont uniquement disponibles en combinaison avec ce document. Une liste des documents se trouve au 🕮 chapitre 5.2 "Documents et logiciels".

Les documents requis sont disponibles sous www.nord.com.

1.3 Conventions de représentation

1.3.1 Avertissements

Les mises en garde pour la sécurité des utilisateurs et des interfaces de bus sont mise en évidence comme suit :

RISQUE

Cette mise en garde signale des risques qui entraînent des blessures graves voire mortelles.

AVERTISSEMENT

Cette mise en garde signale des risques pouvant provoquer des blessures graves voire mortelles.

DANGER

Cette mise en garde signale des risques pouvant provoquer des blessures légères ou de moyenne gravité.

ATTENTION

Cette mise en garde signale un risque de dommage matériel.

1.3.2 Autres indications

Informations

Cette indication présente des conseils et informations importantes.

1.4 Utilisation conforme

Les fonctions PLC de la société Getriebebau NORD GmbH & Co. KG constituent une extension de fonctions par logiciel pour les variateurs de fréquence et les démarreurs NORD. Elles sont liées à l'appareil correspondant de façon indissociable et ne peuvent pas être utilisées sans lui. Les consignes de sécurité spécifiques de l'appareil concerné qui sont indiquées dans le manuel correspondant doivent ainsi être appliquées pleinement (🕮 chapitre 5.2 "Documents et logiciels").

Les fonctions PLC servent essentiellement de solution pour des tâches d'entraînement complexes avec un ou plusieurs appareils dans le domaine des techniques d'entraînement électroniques. Elles permettent également de simplifier les fonctions de commande et de surveillance proches de l'entraînement par un appareil équipé de façon correspondante.

2 Sécurité

2.1 Recrutement et qualification du personnel

Les fonctions PLC doivent uniquement être mises en service par des électriciens qualifiés. Ceux-ci doivent disposer des connaissances requises sur les fonctions PLC, sur la technique d'entraînement électronique utilisée et sur les outils de configuration (par ex. le logiciel NORD CON) et la périphérie liée à la tâche d'entraînement (entre autres, la commande).

Les électriciens doivent en outre être familiarisés avec l'installation, la mise en service et le fonctionnement des capteurs et des dispositifs de commande électronique d'entraînement. Ils doivent aussi connaître et suivre toutes les directives de prévention des accidents, prescriptions et lois en vigueur sur le lieu d'installation.

2.1.1 Personnel qualifié

Par personnel qualifié l'on entend des personnes qui en raison de leur formation et de leur expérience possèdent suffisamment de connaissances dans un domaine particulier et qui sont familiarisées avec les directives de sécurité du travail et de prévention des accidents ainsi que les règles de la technique reconnues.

Les personnes doivent être autorisées par le détenteur de l'installation à exécuter les opérations requises.

2.1.2 Électricien

Un électricien est une personne qui en raison de sa formation et de son expérience possède suffisamment de connaissances pour :

- la mise en service, l'arrêt, la mise hors tension, la mise à la terre et le marquage des circuits et des appareils,
- la maintenance conforme et l'utilisation de dispositifs de protection selon les normes de sécurité définies,
- les soins d'urgence aux blessés.

2.2 Consignes de sécurité

Utilisez la fonction technologique **Fonctions PLC** et l'appareil de la société Getriebebau NORD GmbH & Co. KG exclusivement conformément aux prescriptions, 🕮 chapitre 1.4 "Utilisation conforme".

Pour une utilisation sans danger de la fonction technologique, vous devez tenir compte des consignes du présent mode d'emploi.

Ne mettez l'appareil en service que s'il n'a pas été modifié sur le plan technique et à condition de disposer des protections requises. Veillez à ce que tous les connecteurs et câbles soient dans un état irréprochable.

Les travaux sur et avec l'appareil doivent uniquement être effectués par le personnel qualifié, 🕮 chapitre 2.1 "Recrutement et qualification du personnel".

3 PLC

3.1 Généralités

Les variateurs de fréquence NORD des séries SK 180E/SK 190E, SK 2xxE, SK 2xxE-FDS, SK 520E – SK 545E et SK 5xxP ainsi que les démarreurs de la série SK 155E-FDS/SK 175E-FDS comprennent un traitement logique conformément à la norme CEI 61131-3 en vigueur pour les automates programmables industriels (API / anglais : Programmable Logic Controller, PLC). La vitesse de réaction ou la puissance de calcul du PLC permet de réaliser de plus petites tâches dans le domaine du variateur. Des entrées de variateur ou des informations entrant via un bus de terrain peuvent ainsi être surveillées, évaluées et traitées ultérieurement en tant que valeurs de consigne correspondantes pour le variateur de fréquence. En combinaison avec d'autres appareils NORD, une visualisation des états de l'installation et une saisie de paramètres client spéciaux sont également possibles. De cette manière, dans certaines limites, un potentiel d'économies est obtenu en éliminant une solution PLC externe utilisée jusqu'à présent. Le langage de programmation Liste d'instructions (IL) est pris en charge. La liste d'instructions est un langage de programmation orienté sur la machine et basé sur du texte. Son étendue et son application sont définies dans la norme CEI 61131-3.

Informations

La programmation et le téléchargement dans l’appareil sont exclusivement effectués par le biais du programme NORD CON de NORD.

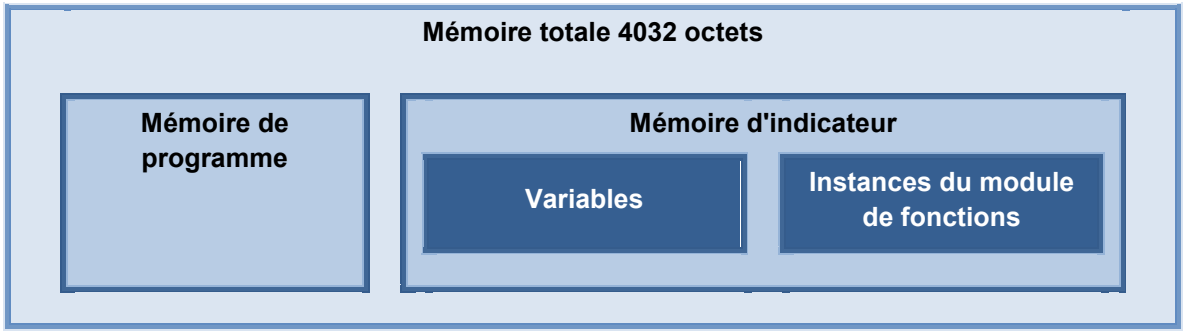
3.1.1 Spécifications de PLC

Fonction	Spécifications		
Norme	Selon CEI 61131-3		
Langage	Liste d'instructions (IL), texte structuré (ST)		
Tâche	Une tâche cyclique, appel de programme toutes les 5 ms		
Puissance de calcul	Environ 200 ordres AWL par ms		
Mémoire de programme	SK 5xxP, SK 520E … SK 545E, SK 2xxE, SK 2x0E-FDS	SK 190E / SK 180E	SK 155E-FDS / SK 175E-FDS
	8128 octets pour les indicateurs, les fonctions et le programme PLC	2032 octets pour les indicateurs, les fonctions et le programme PLC	2028 octets pour les indicateurs, les fonctions et le programme PLC
Nombre max. de commandes	env. 2580 commandes	env. 660 commandes	env. 660 commandes
	Remarque : Il s'agit ici d'une valeur moyenne. Une utilisation intensive des indicateurs, données de processus et fonctions minimise fortement le nombre possible de lignes ; voir le chapitre "Ressources".		
Messageries CAN librement accessibles	20		
Appareils pris en charge	SK 5xxP SK 54xE SK 53xE / SK 52xE à partir de V3.0 SK 2xxE à partir de V2.0 SK 2x0E-FDS SK 180E / SK 190E SK 155E-FDS / SK 175E- FDS		

3.1.2 Structure de PLC

3.1.2.1 Mémoire

La mémoire du PLC est divisée en mémoire de programme et mémoire d'indicateur. En plus des variables, les instances des blocs fonctionnels sont enregistrées dans la zone de la mémoire d'indicateur. Une instance est une zone de mémoire dans laquelle toutes les variables internes d'entrée et de sortie d'un module de fonctions sont enregistrées. Chaque déclaration du module de fonctions nécessite sa propre instance. La limite entre la mémoire de programme et la mémoire d'indicateur est définie de manière dynamique, selon la taille de la zone d'indicateur.



Dans la mémoire d'indicateur, deux classes différentes sont enregistrées dans la section des variables :

[VAR]

Variable de mémoire pour l'enregistrement d'informations d'aide et d'états. Les variables de ce type sont réinitialisées à chaque démarrage de PLC. Pendant l'exécution cyclique de PLC, les contenus de la mémoire sont conservés.

[VAR_ACCESS]

Sert à lire et décrire les données de processus (entrées, sorties, valeurs de consigne, etc.) du variateur de fréquence. Ces valeurs sont créées de nouveau lors de chaque cycle PLC.

3.1.2.2 Image de processus

L'appareil dispose de différentes dimensions physiques telles que le couple, la vitesse, les entrées, les sorties, etc. Ces dimensions sont réparties en valeurs réelles et de consigne. Elles peuvent être chargées dans l'image de processus du PLC et influencées par celle-ci. Les valeurs de processus requises doivent être définies dans la liste des variables sous la classe VAR_ACCESS. Pour chaque cycle PLC, toutes les données de processus du variateur définies sont lues de nouveau dans la liste des variables. À la fin de chaque cycle PLC, les données de processus pouvant être décrites sont retransmises au variateur. Voir la figure suivante.

Compte tenu de cette séquence, il est important de programmer une exécution de programme cyclique. La programmation de boucles afin d'attendre certains événements (par ex. un changement de niveau sur une entrée) ne permet pas d'obtenir le résultat souhaité. Le comportement est différent si les blocs fonctionnels accèdent aux valeurs de processus. Les valeurs de processus sont alors lues

lors de l'appel du bloc fonctionnel et lors de l'arrêt du bloc, les valeurs de processus sont immédiatement enregistrées.

Informations

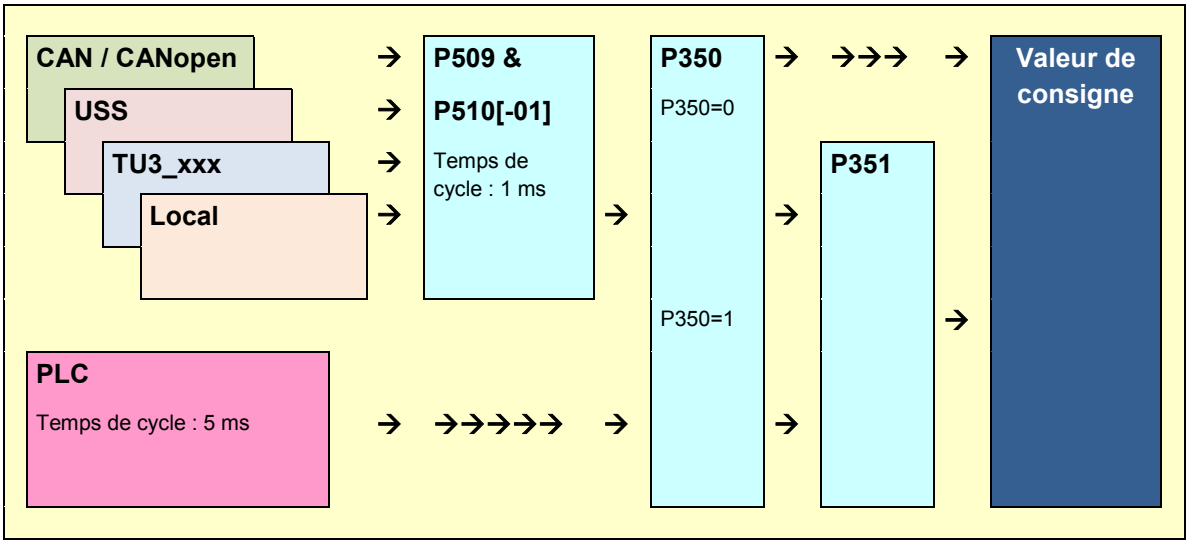
Si les blocs Motion MC_Power, MC_Reset, MC_MoveVelocity, MC_Move, MC_Home ou MC_Stop sont utilisés, les valeurs de processus "PLC_Control_Word" et "PLC_Set_Val1" à "PLC_Set_Val5" ne doivent pas être utilisées, car dans ce cas, les valeurs de la liste des variables remplaceraient systématiquement les modifications du bloc fonctionnel.

3.1.2.3 Tâche du programme

L'exécution du programme dans PLC est effectuée en une seule tâche. La tâche est appelée de façon cyclique toutes les 5 ms et sa durée de traitement max. est de 3 ms. Si un programme plus long ne peut pas être traité pendant cette durée, l'exécution du programme est interrompue et poursuivie lors de la prochaine tâche de 5 ms.

3.1.2.4 Traitement de la valeur de consigne

Le variateur dispose d'un certain nombre de sources de valeurs de consigne qui sont liées en définitive par le biais de plusieurs paramètres pour former une valeur de consigne du variateur de fréquence.



Si PLC est activé (P350=1), une présélection des valeurs de consigne de sources externes (valeurs de consigne principales) est réalisée via P509 & P510[-01]. Avec P351, il est enfin possible de décider d'utiliser des valeurs de consigne de PLC ou des valeurs entrant via P509/P510[-01]. Une combinaison des deux est également possible. Aucune modification relative aux valeurs de consigne secondaires (P510[-02]) n'est associée à la fonction PLC. Toutes les sources de valeurs de consigne secondaires et PLC transmettent leurs valeurs de consigne secondaires au variateur de fréquence avec la même priorité.

3.1.2.5 Traitement des données via l'accumulateur

L'accumulateur constitue l'unité informatique centrale du PLC. Pratiquement toutes les instructions fonctionnent uniquement en combinaison avec l'accumulateur. NORD PLC dispose de trois accumulateurs. Il s'agit d'un accumulateur 1 de 32 bits, d'un accumulateur 2 et de l'AE au format BOOL. L'AE est utilisé pour toutes les opérations booléennes de chargement, d'enregistrement et de

comparaison. Lorsqu'une valeur booléenne est chargée, elle est représentée dans l'AE. Des opérandes de relation transmettent le résultat à l'AE et des sauts conditionnels sont déclenchés par l'AE. L'accumulateur 1 et l'accumulateur 2 sont utilisés pour tous les opérandes au format BYTE, INT et DINT. L'accumulateur 1 est l'accumulateur principal et l'accumulateur 2 se charge des fonctions auxiliaires. Tous les opérandes de chargement et d'enregistrement fonctionnent par le biais de l'accumulateur 1. Tous les opérateurs arithmétiques enregistrent leur résultat dans l'accumulateur 1. Le contenu de l'accumulateur 1 est déplacé dans l'accumulateur 2 lors de chaque commande de chargement. Un opérateur suivant peut lier les deux accumulateurs ou les évaluer et enregistrer de nouveau le résultat dans l'accumulateur 1 qui est désigné ci-après de manière générale par le terme "accumulateur".

3.1.3 Étendue des fonctions

PLC prend en charge une quantité d'opérateurs, de fonctions et de modules de fonctions standard qui sont définis dans CEI 1131-3. Des informations détaillées se trouvent aux chapitres suivants. Une explication relative aux blocs fonctionnels qui sont également pris en charge est aussi disponible.

3.1.3.1 Motion Control Lib

Motion Control Lib est basée sur la spécification PLCopen "Function blocks for motion control" (blocs fonctionnels pour la commande de mouvement). Cette bibliothèque contient essentiellement des blocs fonctionnels pour le déplacement de l'entraînement. De plus, des blocs fonctionnels sont prévus pour la lecture et l'écriture de paramètres de l’appareil.

3.1.3.2 Réducteur électronique avec scie volante

Le variateur de fréquence dispose des fonctions de réducteur électronique (synchronisme en mode de positionnement) et de scie volante. Ces fonctions permettent de faire fonctionner le variateur avec un autre entraînement en synchronisme angulaire. De plus, avec la fonction supplémentaire Scie volante, il est possible de se synchroniser avec une position précise par rapport à un entraînement en marche. Le mode de fonctionnement Réducteur électronique peut être démarré et arrêté à tout moment. Une combinaison du contrôle de position classique avec ses commandes de déplacement et ses fonctions de réducteur est possible. Pour les fonctions de réducteur, un variateur de fréquence NORD avec un bus CAN interne est obligatoirement nécessaire sur l'axe maître.

3.1.3.3 Visualisation

À l'aide d'une ControlBox ou d'une ParameterBox, la visualisation de l'état de fonctionnement et le paramétrage du variateur de fréquence sont possibles. Ou bien, la fonctionnalité du maître CANopen des panneaux bus CAN du PLC peut être utilisée pour l'affichage d'informations.

ControlBox

La version la plus simple pour la visualisation est la ControlBox. Deux valeurs de processus permettent d'accéder à l'affichage à 4 chiffres et à l'état du clavier. Ainsi, des applications d'interface homme-machine (IHM) simples peuvent être créées très rapidement. Afin que le PLC puisse accéder à l'affichage, P001 doit être réglé sur "PLC-ControlBox Value". Une autre particularité est que le menu des paramètres n'est plus atteint par le biais des touches fléchées. À la place, les touches "On" et "Enter" doivent être actionnées en même temps.

ParameterBox

Dans le mode de visualisation, PLC permet de définir chacun des 80 caractères dans l'affichage de P-Box (4 lignes de 20 caractères). Des nombres ainsi que des textes peuvent être transmis. De plus, des saisies avec le clavier sur la P-Box de PLC sont possibles. De cette façon, des fonctions IHM (affichage de valeurs réelles, changement de fenêtre, transmission de valeurs de consigne, etc.) sont réalisées. L'accès à l'affichage de P-Box est effectué par le biais des blocs fonctionnels dans PLC.

La visualisation est effectuée via l'affichage de la valeur de fonctionnement de la ParameterBox. Le contenu de l'affichage de la valeur de fonctionnement est défini avec le paramètre de P-Box P1003. Ce paramètre se trouve sous l'option de menu principal "Affichage". P1003 doit être réglé sur la valeur "PLC display". À l'aide des touches fléchées de droite et de gauche, l'affichage de la valeur de fonctionnement peut être de nouveau sélectionné ultérieurement. L'affichage contrôlé par PLC apparaît alors. Ce paramètre reste effectif même après une nouvelle mise en service.

3.1.3.4 Régulateur de processus

Le régulateur de processus est un régulateur PID-T1 avec une grandeur de sortie limitée. Avec ce module fonction, il est possible de réaliser simplement des commandes complexes dans PLC, par le biais desquelles divers processus (tels que par ex. des régulations de pression) peuvent être résolus avec plus d'élégance qu'avec les régulateurs tout ou rien fréquemment utilisés.

3.1.3.5 Communication CANopen

Hormis les canaux de communication disponibles en standard, le PLCoffre plusieurs autres possibilités de communication. Par le biais de l'interface de bus CAN du variateur de fréquence ou du bus système, des relations de communication supplémentaires avec d'autres appareils peuvent être établies. Le protocole utilisé pour ce faire est CANopen. Les relations de communication sont limitées au transfert de données PDO et aux commandes NMT. La communication CANopen disponible en standard dans le variateur de fréquence via SDO, PDO1, PDO2 et Broadcast n'est pas affectée par cette fonction PLC.

PDO (Process Data Objects)

Les PDO permettent de commander et de surveiller d'autres variateurs de fréquence. Il est également possible de connecter des appareils d'autres fournisseurs au PLC. Il peut s'agir de modules d'E/S, de codeurs CANopen, de panneaux, etc. Ainsi, le nombre d'entrées / de sorties du variateur de fréquence peut être étendu à volonté et des sorties analogiques sont aussi possibles.

NMT (Network Management Objects)

Tous les appareils CANopen doivent être mis à l'état de bus CANopen "Opérationnel" par le maître bus. Seul cet état de bus rend la communication PDO possible. Si aucun maître bus ne se trouve dans le bus CANopen, ceci doit être effectué par le PLC. Le module de fonctions FB_NMT est disponible à cette fin.

3.2 Création de programmes PLC

La création de programmes PLC est effectuée exclusivement via le programme PC NORD CON. L'éditeur PLC est ouvert à l'aide de l'option de menu "File/New/PLC program" ou du symbole  . Ce bouton est uniquement actif si un appareil avec les fonctions PLC est au centre de la vue d'ensemble des appareils.

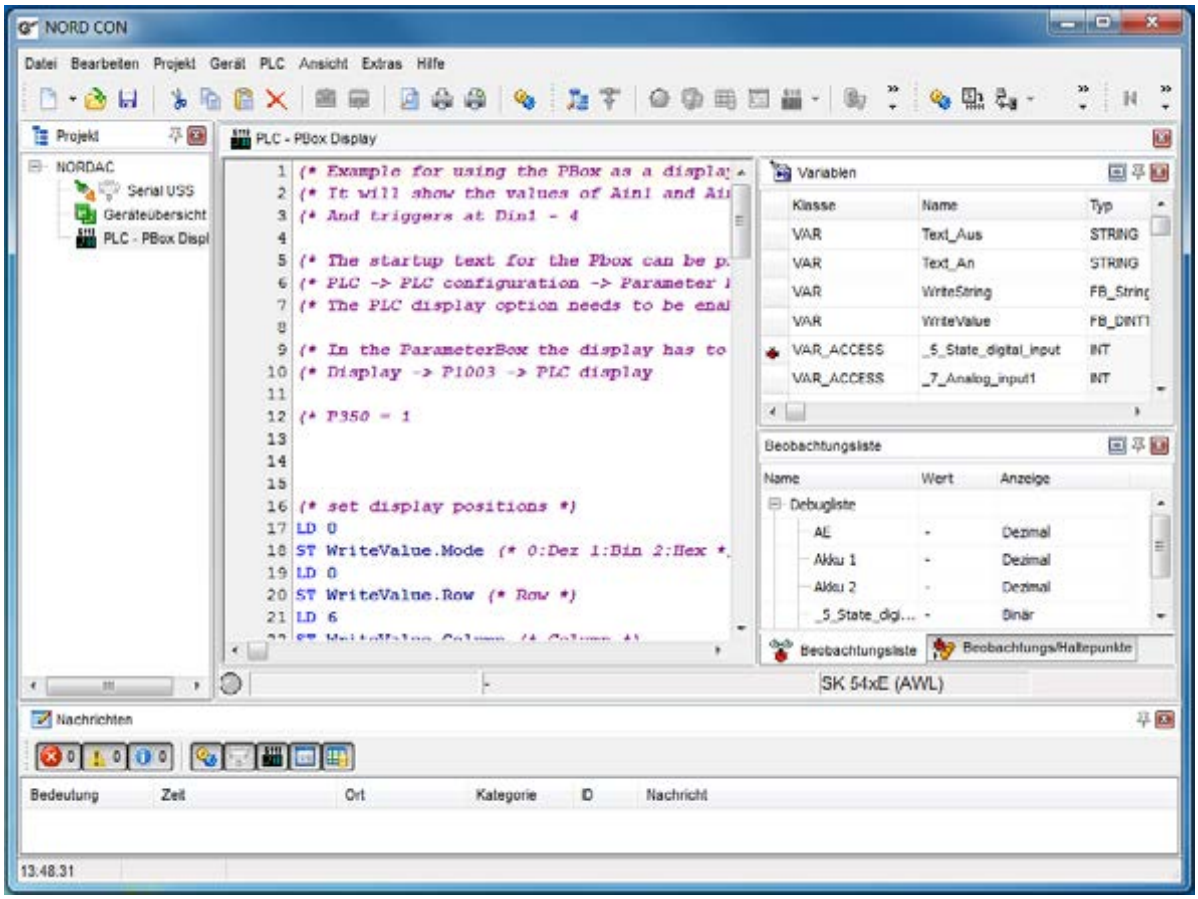
3.2.1 Chargement, enregistrement et impression

Les fonctions de chargement, enregistrement et impression sont exécutées à l'aide des entrées correspondantes du menu principal ou des barres d'outils. Lors de l'ouverture, il est recommandé de sélectionner le type de fichier "PLC Program" (*.awlx, *.nstx) dans la fenêtre "Open". Ainsi, seuls les fichiers pouvant être lus par l'éditeur PLC seront affichés. Si le programme PLC créé doit être enregistré, la fenêtre de l'éditeur PLC doit être active. Le programme PLC est sauvegardé avec "Save" ou "Save as". Dans le cas de l'opération "Save as", ceci peut également être vérifié avec le type de fichier (programme PLC (*.awlx*.nstx)). Pour l'impression du programme PLC, la fenêtre PLC correspondante doit être active. L'impression est lancée à l'aide de "File/Print" ou du symbole correspondant.

Les programmes PLC peuvent aussi être enregistrés en tant que programmes PLC sauvegardés. Pour cela, dans la fenêtre de fichier, l'utilisateur doit définir le type de fichier sur "Fichiers AWL sauvegardés" ou "Fichiers ST sauvegardés". Ensuite, le programme PLC est enregistré dans une version codée (*.awls ou *.nsts) et une version normale (*.awlx, *.nstx). Le programme PLC codé peut ensuite uniquement être transmis à l'appareil (voir).

3.2.2 Éditeur

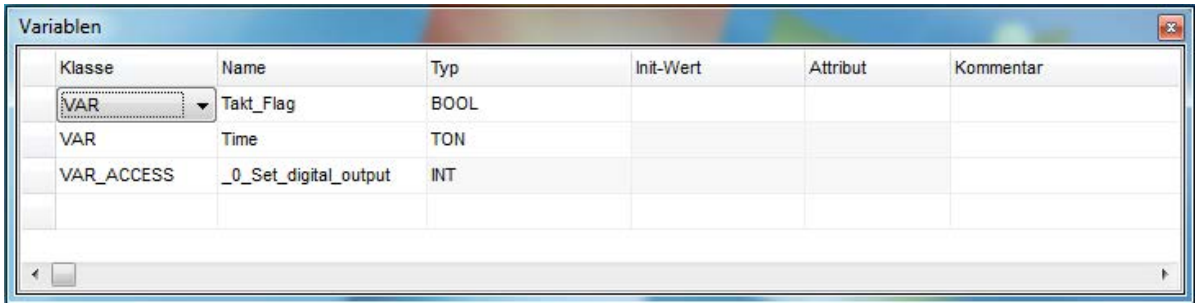
L'éditeur PLC est divisé en quatre fenêtres différentes.



Les différentes fenêtres sont décrites de façon plus détaillée dans les parties suivantes.

3.2.2.1 Variables et déclaration de modules de fonctions

Les variables, valeurs de processus et blocs fonctionnels nécessaires au programme sont déclarés dans cette fenêtre.



Variables

Les variables sont créées en définissant la Classe "VAR". Le nom de la variable peut être choisi librement. Dans le champ Type, il est possible de sélectionner BOOL, BYTE, INT ou DINT. Pour les variables, une initialisation de démarrage peut être saisie sous Valeur initiale.

Valeurs de processus

Celles-ci sont créées en sélectionnant l'entrée "VAR_ACCESS" sous Classe. Le Nom ne peut pas être choisi librement et le champ Valeur initiale est bloqué pour ce type.

Modules de fonctions

Sous Classe, l'entrée "VAR" est sélectionnée. Le Nom pour l'instance correspondante du module de fonctions (FB) peut être choisi librement. Le FB souhaité est sélectionné sous Type. Une Valeur initiale ne peut pas être définie pour le FB.

Toutes les options de menu qui concernent la fenêtre de variables sont appelées par le biais du menu contextuel. Des entrées peuvent ainsi être ajoutées et supprimées. Aussi bien les variables et variables de processus pour l'observation (fonction de point de surveillance) que pour le débogage (point d'arrêt) sont activées.

3.2.2.2 Fenêtre de saisie

Cette fenêtre sert à la saisie de programme et à la représentation du programme Liste d'instructions. Les fonctions suivantes sont disponibles :

- Mise en surbrillance de la syntaxe
- Signets
- Déclaration des variables
- Débogage

Mise en surbrillance de la syntaxe

Si la commande et la variable qui lui est affectée sont détectées par l'éditeur, la commande est représentée en bleu et la variable en noir. Si ce n'est pas le cas, la représentation apparaît en caractères italiques, fins et noirs.

Signets

Étant donné que les programmes dans l'éditeur peuvent atteindre une certaine longueur, il est possible de marquer et d'accéder de façon ciblée à des points importants du programme à l'aide de la fonction Signets. Pour marquer une ligne, le curseur doit se trouver dans cette ligne. Via l'option de menu "Switch bookmark" (bouton droit de la souris, menu), la ligne avec le signet souhaitée est marquée. Il est possible d'accéder aux signets par le biais de l'option de menu "Go to bookmark".

Déclaration des variables

Par le biais du menu de l'éditeur "Add Variable" (bouton droit de la souris), de nouvelles variables peuvent être déclarées par l'éditeur.

Débogage

Dans l'éditeur, la position des points de surveillance et d'arrêt est définie pour la fonction Débogage. Ceci peut être réalisé à l'aide des options de menu "Switch breakpoint" (Breakpoints) et "Switch monitoring point" (Watchpoints). La position des points d'arrêt peut aussi être définie en cliquant sur le bord gauche de la fenêtre de l'éditeur. Les variables et valeurs de processus à lire à partir du variateur de fréquence pendant le débogage, doivent être marquées. Ceci peut être effectué dans l'éditeur par le biais des options de menu "Debug variable" et "Watch variable". Pour cela, la variable correspondante doit être marquée avant de sélectionner l'option de menu souhaitée.

3.2.2.3 Fenêtre d'affichage des points de surveillance et d'arrêt

Cette fenêtre dispose de deux onglets qui sont décrits ci-après.

Points d'arrêt

Cette fenêtre présente tous les points d'arrêt et de surveillance définis. Ils peuvent être activés/désactivés par le biais des cases et supprimés avec le bouton Supprimer. Le bouton droit de la souris permet d'accéder au menu correspondant.

Liste d'observation

Toutes les variables sélectionnées pour l'observation sont représentées ici. Dans la colonne Valeur, le contenu actuel est représenté. Dans la colonne Affichage, le format de représentation est sélectionné.

3.2.2.4 Fenêtre de messages PLC

Dans cette fenêtre, tous les messages d'état et d'erreur de PLC sont saisis. Si le programme est traduit correctement, le message "Translated without error" apparaît. L'utilisation des ressources est indiquée une ligne après. En cas d'erreurs dans le programme PLC, le message "Error X" apparaît, X correspondant au nombre d'erreurs. Dans les lignes suivantes, le message d'erreur spécifique est affiché dans le format :

[Numéro de ligne] : description de l'erreur

3.2.3 Transmission du programme à l'appareil

Il existe plusieurs moyens de transmettre un programme PLC à l'appareil.

Transmettre directement le programme PLC :

1. Sélectionner l'appareil dans l'arborescence du projet.
2. Ouvrir le menu contextuel (appuyer sur le bouton droit de la souris).
3. Exécuter la fonction "Transmettre le programme PLC à l'appareil".
4. Sélectionner le fichier dans la fenêtre de fichier, puis appuyer sur "Ouvrir".

Transmettre le programme PLC avec l'éditeur PLC (hors ligne) :

1. Ouvrir le programme PLC avec la fonction "Ouvrir" (Fichier->Ouvrir).
2. Connecter l'éditeur PLC avec l'appareil (PLC->Connecter).
3. Compiler le programme PLC.
4. Transmettre le programme PLC à l'appareil.

Transmettre le programme PLC avec l'éditeur PLC (en ligne) :

1. Sélectionner l'appareil dans l'arborescence du projet.
2. Démarrer l'éditeur PLC.
3. Ouvrir le programme PLC.
4. Importer le programme PLC dans la vue en ligne.
5. Compiler le programme PLC.
6. Transmettre le programme PLC à l'appareil 

Informations **SK 1xxE-FDS - nombre limité de cycles d'écriture**

Dans les appareils SK 155E-FDS / SK 175E-FDS, une mémoire flash est utilisée comme support de stockage. Le nombre de cycles d'écriture d'une mémoire flash est fortement limité. Par conséquent, le programme est uniquement chargé en standard dans la mémoire RAM. Il peut alors être démarré et testé. Si le PLC doit ensuite être redémarré, le programme doit être de nouveau chargé vers l'appareil afin d'initialiser les variables PLC. Si le programme doit être enregistré dans l'appareil de façon durable, l'utilisateur doit exécuter l'action "Transmettre le programme PLC vers l'appareil et l'enregistrer".

3.2.4 Débogage

Comme il est rare que les programmes fonctionnent du premier coup, NORD PLC offre plusieurs possibilités pour trouver les erreurs. Ces possibilités peuvent être classées en deux grandes catégories, décrites ci-après.

3.2.4.1 Points de surveillance (Watchpoints)

La version de débogage la plus simple est la fonction avec points de surveillance. Elle offre une vue d'ensemble rapide du comportement de certaines variables. Pour cela, un point de surveillance est défini à un endroit quelconque du programme. Lorsque PLC exécute cette ligne de programme, jusqu'à 5 valeurs sont enregistrées et affichées dans la liste d'observation (fenêtre "Observation List"). Les 5 valeurs à observer peuvent être sélectionnées dans la fenêtre de saisie ou la fenêtre de variables via le menu contextuel. Si un point de surveillance a été défini sans code de programme, NORD CON recherche la ligne de code précédente. Si cette ligne de code est atteinte dans l'exécution du programme, l'actualisation des valeurs est réalisée. Si un point de surveillance est ignoré par un saut (instruction JMP, IF, Switch), les valeurs ne sont pas actualisées.

Informations

Les variables de blocs fonctionnels ne peuvent pas être ajoutées à la liste de surveillance dans la version actuelle !

3.2.4.2 Points d'arrêt (Breakpoints)

Via les points d'arrêt, il est possible d'arrêter le programme PLC de façon ciblée à la ligne de programme souhaitée. Si le PLC rencontre un point d'arrêt, l'AE, l'accumulateur 1 et l'accumulateur 2 sont lus, de même que toutes les variables qui ont été sélectionnées par le biais de l'option de menu "Debug variables" (menu contextuel). Il est possible de définir jusqu'à 5 points d'arrêt dans le programme PLC. Cette fonction est démarrée par le biais du symbole  . Le programme fonctionne alors jusqu'à ce qu'un point d'arrêt soit déclenché. Une nouvelle activation de la barre d'outils permet de relancer le programme jusqu'au prochain point d'arrêt. Si le programme doit continuer à fonctionner, le symbole  est activé.

3.2.4.3 Exécution pas à pas (Single Step)

Avec cette méthode de débogage, il est possible d'exécuter le programme PLC ligne par ligne, par étapes individuelles. À chaque étape, toutes les variables sélectionnées sont lues à partir du PLC du VF et affichées dans la fenêtre "Observation List". Les valeurs à observer peuvent être sélectionnées dans la fenêtre de saisie ou la fenêtre de variables à l'aide du menu du bouton droit de la souris. Pour le débogage avec l'exécution pas à pas, il est nécessaire d'avoir défini au moins un point d'arrêt avant le démarrage du débogage. L'actionnement du symbole  permet d'activer le mode débogage. Un débogage pas à pas par les lignes suivantes est uniquement possible par le biais du symbole  une fois que le programme a atteint le premier point d'arrêt. Certaines lignes de commande contiennent plusieurs commandes individuelles. À cet effet, il est possible de devoir exécuter deux ou plusieurs étapes avant que l'affichage des étapes ne continue dans la fenêtre de saisie. La position actuelle est indiquée par la petite flèche dans la fenêtre de gauche de l'éditeur PLC. L'actionnement du symbole  permet de continuer à exécuter le programme jusqu'au point d'arrêt suivant. Si le programme doit continuer à fonctionner, le symbole  est activé.

3.2.5 Configuration PLC

Le symbole  permet d'ouvrir la fenêtre de configuration PLC. Les paramètres de base pour PLC peuvent être définis ici. Ils sont décrits ci-après.

Surveillance du temps de cycle

Cette fonction contrôle le temps de traitement max. d'un cycle PLC. Les boucles continues programmées involontairement dans le programme PLC peuvent ainsi être interceptées. En cas de dépassement, l'erreur E22.4 est déclenchée dans le variateur de fréquence.

Autoriser le module de fonctions ParameterBox

Si une visualisation via la ParameterBox doit être effectuée dans le programme PLC, cette option doit être activée. Sinon, les blocs fonctionnels correspondants génèrent une erreur de compilation lors du démarrage du variateur de fréquence.

Données de commande non valides

Le PLC peut évaluer les mots de commande entrant par le biais des systèmes de bus possibles. Toutefois, les mots de commande peuvent uniquement passer lorsque le bit "PZD valid" (bit 10) est défini. Si des mots de commande non conformes au protocole USS doivent être évalués par le PLC, cette option doit être activée. Le bit 10 dans le premier mot n'est alors plus interrogé.

Démarrage à chaud après une erreur

Toutes les variables sont systématiquement chargées lors du démarrage du PLC avec "0" ou leur valeur d'initialisation, et ce, qu'il s'agisse du démarrage après un arrêt, d'un téléchargement de programme ou d'une erreur PLC. Via cette option, le contenu des variables n'est pas modifié en cas de démarrage à chaud. Un démarrage à chaud est effectué après une commande d'arrêt PLC ou une erreur PLC.

Ne pas mettre en pause l'heure du système au point d'arrêt

Pendant le débogage, si PLC se trouve à un point d'arrêt ou en mode d'exécution pas à pas, l'heure du système est mise en pause. L'heure du système constitue la base pour toutes les horloges de PLC. Si l'heure du système doit continuer à fonctionner pendant le débogage, cette fonction doit être activée.

3.3 Blocs fonctionnels

Les blocs fonctionnels sont de petits programmes qui peuvent enregistrer leurs valeurs d'état dans des variables internes. Par conséquent, pour chaque bloc fonctionnel, une propre instance est créée dans la liste des variables de NORD CON. Si un temporisateur doit contrôler 3 temps parallèlement, il doit être créé trois fois dans la liste des variables.

Informations **Détermination d'un front de signal**

Afin que les blocs fonctionnels suivants puissent déterminer un front à l'entrée, il est nécessaire que l'appel de la fonction soit exécuté deux fois avec des états différents à l'entrée.

3.3.1 CANopen

Via les blocs fonctionnels, le PLC peut configurer, surveiller des canaux PDO et transmettre sur ces derniers. Le PLC peut envoyer ou recevoir jusqu'à 8 octets de données de processus via un PDO. L'accès à chacun de ces PDO est effectué par le biais d'une adresse individuelle (COB-ID). Jusqu'à 20 PDO peuvent être configurés dans le PLC. Pour faciliter la commande, COB-ID n'est pas saisi directement. Au lieu de cela, l'adresse de l'appareil et le numéro PDO sont transmis au module de fonctions. COB-ID en résultant est déterminé sur la base de Pre-Defined Connection Set (CiA DS301). Les COB-ID suivants possibles en résultent pour le PLC.

PDO d'émission		**PDO de surveillance**	
PDO	**COB-ID**	**PDO**	**COB-ID**
PDO1	200h + adresse de l'appareil	PDO1	180h + adresse de l'appareil
PDO2	300h + adresse de l'appareil	PDO2	280h + adresse de l'appareil
PDO3	400h + adresse de l'appareil	PDO3	380h + adresse de l'appareil
PDO4	500h + adresse de l'appareil	PDO4	480h + adresse de l'appareil

Variateurs de fréquence NORD utilisent pour la transmission des données de processus PDO1 ; PDO2 est uniquement utilisé pour la valeur de consigne/réelle 4 et 5.

3.3.1.1 Vue d'ensemble

Module de fonctions	**Explication**
FB_PDOConfig	Configuration PDO
FB_PDOSend	Envoi PDO
FB_PDOReceive	Réception PDO
FB_NMT	Validation et blocage de PDO

3.3.1.2 FB_NMT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Après un *Power UP*, tous les participants CAN se trouvent dans l'état bus Pre-Operational. Dans cet état, vous ne pouvez ni recevoir ni envoyer un PDO. Afin que PLC puisse communiquer avec d'autres participants sur le bus CAN, ces participants doivent être définis dans l'état "Operational". En principe, ceci est effectué par le maître bus. À défaut de maître bus, cette tâche peut être réalisée par FB_NMT. Via les entrées **PRE**, **OPE** ou **STOP**, l'état de tous les participants connectés au bus peut être influencé. Les entrées sont adoptées avec un front positif sur **EXECUTE**. La fonction doit être appelée à plusieurs reprises jusqu'à ce que la sortie **DONE** ou **ERROR** soit définie sur 1.

Si la sortie **ERROR** a été définie sur 1, aucune alimentation de 24 V n'est présente sur la prise RJ45 CAN du variateur ou le pilote CAN du variateur est dans l'état *Bus off*. En cas de front négatif sur **EXECUTE**, toutes les sorties sont réinitialisées sur 0.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	**BOOL**	**DONE**	La commande NMT est envoyée	**BOOL**
PRE	Définition de tous les participants dans l'état Pre-Operational	**BOOL**	**ERROR**	Erreur dans le module de fonctions	**BOOL**
OPE	Définition de tous les participants dans l'état Operational	**BOOL**			
STOP	Définition de tous les participants dans l'état Stopped	**BOOL**			

3.3.1.3 FB_PDOConfig

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

Les PDO sont configurés via ce module de fonctions. Avec une instance de cette fonction, tous les PDO souhaités sont configurés. Pour chaque PDO, le module de fonctions doit seulement être appelé une fois. Jusqu'à 20 PDO peuvent être définis. Chaque PDO a son propre paramétrage. L'affectation des PDO dans d'autres modules de fonctions CANopen est effectuée par le biais de la boîte de message Number. **TARGETID** correspond à l'adresse de l'appareil. Dans le cas de Variateurs de fréquence NORD, cette adresse de l'appareil est réglée dans le P515 ou via le commutateur DIP. Sous PDO, le numéro de boîte de message souhaité est saisi (voir l'introduction). **LENGTH** définit la durée de transmission d'un PDO. Via **DIR**, le sens d'envoi/de réception est défini. Avec le front positif sur l'entrée **EXECUTE**, les données sont reprises. La sortie **DONE** peut être immédiatement interrogée après l'appel du module de fonctions. Si **DONE** est défini sur 1, le canal PDO a alors été configuré. Si **ERROR** = 1, un problème est survenu. La cause exacte est enregistrée dans **ERRORID**. En cas de front négatif sur **EXECUTE**, toutes les sorties sont réinitialisées sur 0.

PDO d'émission		**PDO de surveillance**	
PDO	**COB-ID**	**PDO**	**COB-ID**
PDO1	200h + adresse de l'appareil	**PDO1**	180h + adresse de l'appareil
PDO2	300h + adresse de l'appareil	**PDO2**	280h + adresse de l'appareil
PDO3	400h + adresse de l'appareil	**PDO3**	380h + adresse de l'appareil
PDO4	500h + adresse de l'appareil	**PDO4**	480h + adresse de l'appareil
PDO5	180h + adresse de l'appareil	**PDO5**	200h + adresse de l'appareil
PDO6	280h + adresse de l'appareil	**PDO6**	300h + adresse de l'appareil
PDO7	380h + adresse de l'appareil	**PDO7**	400h + adresse de l'appareil
PDO8	480h + adresse de l'appareil	**PDO8**	500h + adresse de l'appareil

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	BOOL	DONE	PDO configuré	BOOL
NUMBER	Numéro de boîte de messages Plage de valeurs = 0 à 19	BYTE	ERROR	Erreur dans le module de fonctions	BOOL
TARGETID	Adresse de l'appareil Plage de valeurs = 1 à 127	BYTE	ERRORID	Code d'erreur	INT
PDO	PDO Plage de valeurs = 1 à 4	BYTE			
LENGTH	Longueur PDO Plage de valeurs = 1 à 8	BYTE			
DIR	Envoi ou réception Envoi = 1 / Réception = 0	BOOL			

ERRORID	**Explication**
0	Pas d‘erreur
1800h	Plage de valeurs Number dépassée
1801h	Plage de valeurs TARGETID dépassée
1802h	Plage de valeurs PDO dépassée
1803h	Plage de valeurs LENGTH dépassée

Informations

Pas d’utilisation double de l’ID CAN

Aucun ID CAN déjà utilisé par l’appareil ne doit être paramétré !

Adresses de réception correspondantes :

- ID CAN = 0x180 + P515[-01] PDO1
- ID CAN = 0x180 + P515[-01]+1 ID CAN pour codeur absolu
- ID CAN = 0x280 + P515[-01] PDO2

Adresses d'envoi correspondantes :

- ID CAN = 0x200 + P515[-01] PDO1
- ID CAN = 0x300 + P515[-01] PDO2

Exemple dans ST :

```
(* Configurer PDO *)
PDOConfig(
    Execute := TRUE,
    (* Configurer la boîte de messages 1 *)
    Number := 1,
    (* Définir le numéro de nœud CAN *)
    TargetID := 50,
    (* Sélectionner PDO (standard pour le mot de commande PDO1, valeur de consigne 1,
                        valeur de consigne 2, valeur de consigne 3) *)
    PDO := 1,
    (* Définir la longueur des données (standard pour PDO1 égal à 8 *)
    LENGTH := 8,
    (* Envoyer *)
    Dir := 1);
```

ou

```
(* Configurer PDO *)
PDOConfig(
    Execute := TRUE,
    (* Configurer la boîte de messages 1 *)
    Number := 2,
    (* Définir le numéro de nœud CAN *)
    TargetID := 50,
    (* Sélectionner PDO (standard pour PDO2 valeur de consigne 4,
                        valeur de consigne 5 SK540E) *)
    PDO := 2,
    (* Définir la longueur des données (standard pour PDO2 égal à 4 *)
    LENGTH := 4,
    (* Envoyer *)
    Dir := 1);
```

ou

```
(* Configurer PDO *)
PDOConfig(
    Execute := TRUE,
    (* Configurer la boîte de messages 2 *)
    Number := 2,
    (* Définir le numéro de nœud CAN *)
    TargetID := 50,
    (* Sélectionner PDO (standard pour le mot de commande PDO1, valeur réelle 1,
                        valeur réelle 2, valeur réelle 3) *)
    PDO := 1,
    (* Définir la longueur des données (standard pour PDO1 égal à 8 *)
    LENGTH := 8,
    (* Recevoir *)
    Dir := 0);
```

3.3.1.4 FB_PDOReceive

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

Ce module de fonctions surveille l'entrée des messages d'un canal PDO préalablement configuré. Le contrôle commence lorsque l'entrée **ENABLE** est sur 1. Après l'appel de la fonction, la sortie **NEW** doit être vérifiée. Si elle passe sur 1, un nouveau message est alors arrivé. La sortie **NEW** est supprimée lors de l'appel suivant de la fonction. Les données reçues se trouvent dans **WORD1** jusqu'à **WORD4.** Via **TIME**, le canal PDO peut être surveillé quant à la réception cyclique. Si dans **TIME** une valeur comprise entre 1 et 32767 ms est saisie, un message doit alors être reçu au cours de cette période. Sinon, le module de fonctions passe dans l'état d'erreur (**ERROR** = 1). Via la valeur 0, cette fonction peut être désactivée. Le temporisateur de surveillance fonctionne par pas de 5 ms. En cas d'erreur, **ERROR** est défini sur 1. **DONE** est dans ce cas 0. Dans **ERRORID**, le code d'erreur correspondant est alors valable. En cas de front négatif sur **ENABLE**, la réinitialisation de **DONE**, **ERROR** et **ERRORID** est effectuée.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Exécuter	BOOL	NEW	Nouveau PDO reçu	BOOL
NUMBER	Numéro de boîte de messages Plage de valeurs = 0 à 19	BYTE	ERROR	Erreur dans le module de fonctions	BOOL
TIME	Fonction Watchdog Plage de valeurs = 0 à 32767 0 = désactivé 1 à 32767 = temps de surveillance	INT	ERRORID	Code d'erreur	INT
			WORD1	Données de réception mot 1	INT
			WORD2	Données de réception mot 2	INT
			WORD3	Données de réception mot 3	INT
			WORD4	Données de réception mot 4	INT

ERRORID	**Explication**
0	Pas d‘erreur
1800h	Plage de valeurs Number dépassée
1804h	La Box sélectionnée n'est pas correctement configurée
1805h	24 V manquant pour pilote de bus ou le pilote de bus est dans l'état "Bus off"
1807h	Temporisation de réception (fonction Watchdog)

Informations

Temps de cycle PLC

Le cycle PLC correspond à 5 ms, autrement dit, dans le cas d'un appel de la fonction dans le programme PLC, un message CAN peut uniquement être lu toutes les 5 ms. Si plusieurs messages doivent être envoyés rapidement les uns après les autres, des messages peuvent être écrasés.

Exemple dans ST :

```
IF bFirstTime THEN
  (* Définir les appareils dans l'état Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Configurer PDO *)
  PDOConfig(
    Execute := TRUE,
    (* Configurer la boîte de messages 2 *)
    Number := 2,
    (* Définir le numéro de nœud CAN *)
    TargetID := 50,
    (* Sélectionner PDO (standard pour le mot de commande PDO1, valeur réelle 1,
                         valeur réelle 2, valeur réelle 3) *)
    PDO := 1,
    (* Définir la longueur des données (standard pour PDO1 égal à 8 *)
    Length := 8,
    (* Recevoir *)
    Dir := 0);
END_IF;

(* Lire l'état et les valeurs réelles *)
PDOReceive(Enable := TRUE, Number := 2);
IF PDOReceive.New THEN
  State := PDOReceive.Word1;
  Sollwert1 := PDOReceive.Word2;
  Sollwert2 := PDOReceive.Word3;
  Sollwert3 := PDOReceive.Word4;
END_IF
```

3.3.1.5 FB_PDOSend

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

Avec ce module de fonctions, des PDO doivent être envoyés sur un canal préalablement configuré. Il est possible de les envoyer de façon ponctuelle ou cyclique. Les données à envoyer sont saisies dans **WORD1** jusqu'à **WORD4**. Un envoi de PDO est possible indépendamment de l'état CANopen du variateur de fréquence. Le canal PDO préalablement configuré est sélectionné via **NUMBER**. Les données à envoyer sont saisies dans **WORD1** jusqu'à **WORD4**. **CYCLE** permet de choisir un envoi ponctuel (paramètre=0) ou cyclique. Le PDO est envoyé par le biais d'un front positif sur **EXECUTE**. Si **DONE** = 1, toutes les entrées étaient correctes et le PDO est envoyé. Si **ERROR** = 1, un problème s'est produit. La cause exacte est enregistrée dans **ERRORID**. Toutes les sorties sont réinitialisées avec un front négatif sur **EXECUTE**. La base de temps de PLC correspond à 5 ms ; ceci est également valable pour l'entrée **CYCLE**. Seuls des cycles de transmission avec un multiple de 5 ms sont possibles.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	BOOL	DONE	PDO envoyé = 1	**BOOL**
NUMBER	Numéro de boîte de messages Plage de valeurs = 0 à 19	BYTE	ERROR	Erreur dans le module de fonctions	**BOOL**
CYCLE	Cycle de transmission Plage de valeurs = 0 à 255 0 = désactivé 1 à 255 = cycle de transmission en ms	BYTE	ERRORID	Code d'erreur	**INT**
WORD1	Données de transmission mot 1	INT			
WORD2	Données de transmission mot 2	INT			
WORD3	Données de transmission mot 3	INT			
WORD4	Données de transmission mot 4	INT			

ERRORID	**Explication**
0	Pas d‘erreur
1800h	Plage de valeurs Number dépassée
1804h	La Box sélectionnée n'est pas correctement configurée
1805h	24 V manquant pour pilote de bus ou le pilote de bus est dans l'état "Bus off"

Si **DONE** passe sur 1, le message à transmettre a été accepté par le module CAN mais n'est pas encore envoyé. La transmission se déroule en parallèle en arrière-plan. Maintenant, si plusieurs messages doivent être envoyés de façon successive directement via un module de fonctions, il se peut que le message précédent n'ait pas encore été envoyé lors du nouvel appel. Ceci peut être

détecté par le fait que ni le signal **DONE** ni le signal **ERROR** n'a été défini sur 1 après l'appel **CAL**. L'appel **CAL** peut maintenant être répété jusqu'à ce que l'un des deux signaux passe sur 1. Si plusieurs ID CAN différents doivent être décrits par le biais d'un seul module de fonctions, ceci est possible en effectuant une nouvelle configuration du module de fonctions. Ceci ne doit toutefois pas être réalisé dans le même cycle PLC que la transmission. Cela risquerait en effet de supprimer le message à envoyer lors de la configuration via FB_PDOConfig.

Exemple dans ST :

```
IF bFirstTime THEN
  (* Définir les appareils dans l'état Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Configurer PDO *)
  PDOConfig(
    Execute := TRUE,
    (* Configurer la boîte de messages 2 *)
    Number := 2,
    (* Définir le numéro de nœud CAN *)
    TargetID := 50,
    (* Sélectionner PDO (standard pour le mot de commande PDO1, valeur réelle 1, valeur
réelle 2, valeur réelle 3) *)
    PDO := 1,
    (* Définir la longueur des données (standard pour PDO1 égal à 8 *)
    Length := 8,
    (* Recevoir *)
    Dir := 0);
END_IF;

(* Lire l'état et les valeurs réelles *)
PDOReceive(Enable := TRUE, Number := 2);
IF PDOReceive.New THEN
  State := PDOReceive.Word1;
  Sollwert1 := PDOReceive.Word2;
  Sollwert2 := PDOReceive.Word3;
  Sollwert3 := PDOReceive.Word4;
END_IF
```

3.3.2 Réducteur électronique avec scie volante

Pour le *réducteur électronique* ("synchronisme angulaire") et la sous-fonction *Scie volante*, il existe deux blocs fonctionnels qui permettent une commande de ces fonctions. En outre, divers paramètres doivent être définis pour garantir un fonctionnement correct des deux blocs fonctionnels dans le variateur de fréquence maître et esclave. Un exemple est présenté dans le tableau suivant, avec SK 540E.

VF maître			**VF esclave**		
Paramètre	Réglage	Signification	Paramètre	Réglage	Signification
P502[-01]	20	Réglage fréquence après rampe	P509	10 *	CANopen émission *
P502[-02]	15	Incréments position actuelle HighWord	P510[-01]	10	CANopen émission
P502[-03]	10	Incréments position actuelle LowWord	P510[-02]	10	CANopen émission
P503	3	CANopen	P505	0	0,0 Hz
P505	0	0,0 Hz	P515[-02]	P515[-03]$_{\text{Maître}}$	Émission adresse esclave
P514	5	250 kbauds (au moins 100 kbauds)	P546[-01]	4	Addition fréquence
P515[-03]	P515[-02]Esclave	Émission adresse maître	P546[-02]	24	Consigne incréments position HighWord
			P546[-03]	23	Consigne incréments position LowWord
			P600	1,2	Contrôle position Marche
			Uniquement pour FB_Gearing		
			P553[-01]	21	Consigne position LowWord
			P553[-02]	22	Consigne position HighWord
* (P509) ne doit pas obligatoirement être sur {10} "CANopen Broadcast". Toutefois, sur le maître (P502 [-01]), le réglage {21} "Fréquence réelle sans glissement" doit être défini.					

Informations **Position réelle - format de transmission**

La position réelle du maître doit obligatoirement être transmise dans le format "Increments" (Inc).

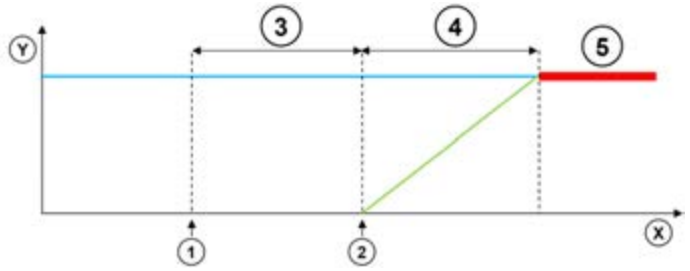
3.3.2.1 Vue d'ensemble

Module de fonctions	Explication
FB_Gearing	Module de fonctions pour la fonction de réducteur simple
FB_FlyingSaw	Module de fonctions pour la fonction de réducteur avec scie volante

3.3.2.2 FB_FlyingSaw

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

La fonction de scie volante est une extension de la fonction de réducteur. À l'aide de cette fonction, il est possible de se synchroniser avec une position précise par rapport à un entraînement en marche. La synchronisation est effectuée de façon relative contrairement à FB_Gearing, autrement dit, l'axe esclave se déplace de façon synchrone par rapport à la position du maître qui se trouvait au démarrage de la "Flying Saw". L'opération de synchronisation est représentée dans la figure suivante.



1	Position de l'initiateur
2	Point de démarrage de l'esclave
3	Distance de l'initiateur par rapport à la position de démarrage du variateur de fréquence esclave
4	Accélération
5	Synchronisme des deux entraînements
X	Position
Y	Vitesse

Lorsque la fonction est démarrée, le variateur de fréquence esclave accélère à la vitesse de l'axe maître. La rampe d'accélération est définie via le chemin **ACCELERATION**. En cas de faible vitesse, la rampe est ainsi plus plate et en cas de vitesses élevées du maître, une rampe plus raide est obtenue pour le variateur de fréquence esclave. Le chemin d'accélération apparaît en tours (1000 = 1,000 rév.) si P553 est indiqué en tant que position de consigne. Si la position de consigne INC est utilisée pour P553, le chemin d'accélération est indiqué en incréments.

Si l'initiateur est défini avec la distance devant la position de l'entraînement esclave enregistrée dans **ACCELERATION**, l'esclave est synchronisé précisément avec la position se déclenchant sur l'entraînement maître.

Le module de fonctions doit être activé via l'entrée **ENABLE**. Le démarrage de la fonction peut être effectué soit via une entrée digitale (P420[-xx]=64, *Démarrage scie volante*) ou **EXECUTE**. Le variateur de fréquence accélère ensuite à la vitesse de l'axe maître. Si la synchronisation par rapport à l'axe maître est atteinte, la sortie **DONE** passe sur 1.

L'entrée **STOP** ou la fonction d'entrée digitale P420[-xx] = 77, *Scie volante stoppée* permet de désactiver la fonction de réducteur ; le variateur de fréquence ralentit à 0Hz et s'arrête. Via l'entrée **HOME**, le variateur est déplacé à la position absolue 0. Une fois la commande **HOME** ou **STOP** terminée, la sortie correspondante affectée est activée. Une nouvelle activation de **EXECUTE** ou l'entrée digitale permet de redémarrer la fonction de réducteur. Avec la fonction d'entrée digitale (P420[-xx] = 63, *Mode de synchronisation arrêt*), la fonction de réducteur peut être arrêtée puis déplacée à la position absolue 0.

Si la fonction est interrompue par la fonction MC_Stop, **ABORT** est défini sur 1. En cas d'erreur, **ERROR** est défini sur 1 et le code d'erreur est défini dans **ERRORID**. Ces trois sorties sont réinitialisées si **ENABLE** passe à 0.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	VALID	Fréquence de consigne prédéfinie atteinte	**BOOL**
EXECUTE	Démarrage de la synchronisation	**BOOL**	DONEHOME	Déplacement Home terminé	
STOP	Arrêt de la synchronisation	**BOOL**	DONESTOP	Commande d'arrêt exécutée	
HOME	Déplacement en position 0	**BOOL**	ABORT	Commande interrompue	**BOOL**
ACCELERATION	Chemin d'accélération (1rév. = 1 000)	**DINT**	ERROR	Erreur dans le module de fonctions	**BOOL**
			ERRORID	Code d'erreur	**INT**

ERRORID	**Explication**
0	Pas d‘erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé

3.3.2.3 FB_Gearing

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Le module de fonctions FB_Gearing permet de synchroniser la position et la vitesse du variateur de fréquence sur celles d'un variateur maître. L'esclave qui utilise cette fonction suit toujours les mouvements du variateur maître.

La synchronisation est toujours absolue ce qui signifie que les positions esclave et maître sont toujours les mêmes.

Informations

Si l'esclave est commuté avec une autre position que le maître en mode de réducteur, l'esclave se déplace avec une fréquence max. en position maître.

Si un ratio de temps mort est indiqué, une nouvelle position en résulte après la remise en service.

La valeur de position sur laquelle la synchronisation est effectuée ainsi que la vitesse doivent être transmises via le canal d'émission. Par le biais de l'entrée **ENABLE**, la fonction est activée à condition que le contrôle de position soit activé et que l'étage final soit validé. L'étage final peut par ex. être validé avec la fonction MC_Power. Si **ENABLE** est défini sur 0, le variateur de fréquence freine à 0Hz et reste arrêté. Le variateur se trouve maintenant de nouveau en mode de contrôle de position. Si MC_Stop est activé, le variateur de fréquence quitte le mode de réducteur et la sortie **ABORT** passe sur 1. En cas d'erreurs dans le module de fonctions, **ERROR** passe sur 1 et la cause de l'erreur est dans **ERRORID**. En définissant **ENABLE** sur 0, **ERROR, ERRORID** et **ABORT** sont réinitialisés.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Synchronisme activé	**BOOL**	VALID	Fonction de réducteur activée	**BOOL**
RELATIVE	Mode relatif (à partir de V2.1)	**BOOL**	ABORT	Commande interrompue	**BOOL**
			ERROR	Erreur dans le module de fonctions	**BOOL**
			ERRORID	Code d'erreur	**INT**

ERRORID	**Explication**
0	Pas d‘erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé
1201h	La valeur de consigne PLC position High n'est pas paramétrée
1202h	La valeur de consigne PLC position Low n'est pas paramétrée

3.3.3 Motion Control

Motion Control Lib est basée sur la spécification PLCopen "Function blocks for motion control" (blocs fonctionnels pour la commande de mouvement). Elle contient des blocs fonctionnels pour la commande et le déplacement d'un variateur de fréquence et offre accès à ses paramètres. Afin que les blocs Motion fonctionnent, certains réglages sont effectués dans les paramètres de l'appareil.

Bloc fonctionnel	Réglages requis
MC_MoveVelocity	• P350 = PLC actif • P351 = La valeur de consigne principale vient de PLC • P553 [-xx] = Consigne de fréquence • P600 = Contrôle de position (mode de positionnement) désactivé
MC_MoveAbsolute	• P350 = PLC actif • P351 = La valeur de consigne principale vient de PLC • P600 = Contrôle de position (mode de positionnement) activé • Dans P553 [-xx] (Consigne_PLC), la position de réglage High Word doit être paramétrée • Dans P553 [-xx] (Consigne_PLC), la position de réglage Low Word doit être paramétrée • Dans P553 [-xx] (Consigne_PLC), la consigne de fréquence doit être paramétrée
MC_MoveRelative	
MC_MoveAdditive	
MC_Home	
MC_Power	• P350 = PLC actif • P351 = Le mot de commande vient de PLC
MC_Reset	
MC_Stop	

Informations

La valeur de consigne_PLC 1 à 5 et le mot de commande PLC peuvent être également décrits via les variables de processus. Si les modules de fonctions Motion Control doivent néanmoins être utilisés, aucune variable de processus correspondante ne doit être déclarée dans le tableau de variables car les résultats des modules de fonctions Motion Control seraient écrasés.

Informations **Détermination d'un front de signal**

Afin que les blocs fonctionnels suivants puissent déterminer un front à l'entrée, il est nécessaire que l'appel de la fonction soit exécuté deux fois avec des états différents à l'entrée.

Bloc fonctionnel	Explication
MC_ReadParameter	Accès en lecture aux paramètres de l’appareil
MC_WriteParameter	Accès en écriture aux paramètres de l’appareil
MC_MoveVelocity	Commande de déplacement en mode vitesse
MC_MoveAbsolute	Commande de déplacement avec indication de position absolue
MC_MoveRelative	Commande de déplacement avec indication de position relative
MC_MoveAdditive	Commande de déplacement avec indication de position supplémentaire
MC_Home	Démarre un déplacement Home
MC_Power	Connexion / déconnexion de la tension du moteur
MC_ReadStatus	État de l'appareil
MC_ReadActualPos	Lit la position actuelle
MC_Reset	Réinitialisation d’erreur dans l’appareil
MC_Stop	Arrête toutes les commandes de déplacement actives

3.3.3.1 MC_Control

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Ce module de fonctions sert à la commande du VF et fournit des possibilités de mot de commande VF un peu plus détaillées que MC_Power. Le VF est commandé via les entrées **ENABLE**, **DISABLEVOLTAGE** et **QUICKSTOP**, voir le tableau suivant.

Module entrées			**Comportement du variateur de fréquence**
ENABLE	QUICKSTOP	DISABLEVOLTAGE	
High	**Low**	**Low**	Le variateur de fréquence est connecté.
Low	**Low**	**Low**	Le variateur de fréquence ralentit à 0Hz (P103) et met ensuite le moteur hors tension.
X	**X**	**High**	Le variateur de fréquence est immédiatement mis hors tension, le moteur s'arrête sans freiner.
X	**High**	**Low**	Le variateur de fréquence réalise un arrêt rapide (P426) et met ensuite le moteur hors tension.

L'entrée **PARASET** permet de définir le jeu de paramètres activé

Si la sortie **STATUS** = 1, le VF est connecté et le moteur est sous tension.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	STATUS	Le moteur est sous tension	**BOOL**
DISABLEVOLTAGE	Mettre hors tension	**BOOL**	ERROR	Erreur dans le module de fonctions	**BOOL**
QUICKSTOP	Arrêt rapide	**BOOL**	ERRORID	Code d'erreur	**INT**
PARASET	Jeu de paramètres actif Plage de valeurs : 0 - 3	**BYTE**			

ERRORID	**Explication**
0	Pas d'erreur
1001h	Fonction d'arrêt activée
1300h	Le VF se trouve dans un état inattendu

Exemple dans ST :

```
(* Valider l'appareil avec Dig3*)
Control.Enable := _5_State_digital_input.2;
(* Les jeux de paramètres sont définis via Dig1 et Dig2. *)
Control.ParaSet := INT_TO_BYTE(_5_State_digital_input and 2#11);
Control;
(* L'appareil est-il validé ? *)
if Control.Status then
  (* Une autre position doit-elle être démarrée ? *)
  if SaveBit3 <> _5_State_digital_input.3 then
    SaveBit3 := _5_State_digital_input.3;
    if SaveBit3 then
      Move.Position := 500000;
    else
      Move.Position := 0;
    end_if;

    Move(Execute := False);
  end_if;
end_if;

(* Démarrer la position lorsque l'appareil est validé.  *)
Move(Execute := Control.Status);
```

3.3.3.2 MC_Control_MS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité							X

Ce module de fonctions sert à la commande du démarreur (MS).

Module entrées				**Comportement du variateur de fréquence**
ENABLE_RIGHT	ENABLE_LEFT	QUICKSTOP	DISABLEVOLTAGE	
High	**Low**	**Low**	**Low**	Le MS est activé, rotation vers la droite
Low	**High**	**Low**	**Low**	Le MS est activé, rotation vers la gauche
High	**High**	**Low**	**Low**	Le MS est désactivé
Low	**Low**	**Low**	**Low**	Le MS ralentit à 0 Hz (P103) et met ensuite le moteur hors tension
X	**X**	**X**	**High**	Le MS est immédiatement mis hors tension, le moteur s'arrête sans freiner
X	**X**	**High**	**Low**	Le MS réalise un arrêt rapide (P426) et met ensuite le moteur hors tension

(X = le niveau à l'entrée n'est pas important)

Si la sortie **STATUS** = 1, le MS est activé et le moteur est sous tension.

Si l'entrée **OPENBRAKE** est définie sur 1, le frein est desserré.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE_RIGHT	Valide à droite	**BOOL**	STATUS	Le moteur est sous tension	**BOOL**
ENABLE_LEFT	Valide à gauche	**BOOL**	ERROR	Erreur dans le module de fonctions	**BOOL**
DISABLEVOLTAGE	Mettre hors tension	**BOOL**	ERRORID	Code d'erreur	**INT**
QUICKSTOP	Arrêt rapide	**BOOL**			
OPENBRAKE	Desserrer le frein	**BOOL**			

ERRORID	**Explication**
0	Pas d‘erreur
1001h	Fonction d'arrêt activée
1300h	Le MS se trouve dans un état inattendu

3.3.3.3 MC_Home

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité		X	X	X	X	X	

Permet au variateur de fréquence de démarrer un accostage du point de référence si **EXECUTE** passe de 0 à 1 (front). Le variateur de fréquence se déplace avec la fréquence de consigne saisie dans **VELOCITY** . Si l'entrée est activée avec le signal de référence de position (P420[-xx] = point de référence), une inversion du sens de rotation se produit. En cas de front négatif du signal de référence de position, la valeur se trouvant dans **POSITION** est reprise. Ensuite, le variateur de fréquence freine à 0Hz, le signal **DONE** passe à 1. Pendant tout le déplacement **HOME**, la sortie **BUSY** est activée.

Si l'opération doit être interrompue (par ex. par un autre module de fonctions MC), **COMMANDABORTED** est défini.

En cas d'erreur, **ERROR** est défini sur 1. **DONE** est dans ce cas 0. Dans **ERRORID**, le code d'erreur correspondant est alors valable.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	**BOOL**	DONE	Position de consigne prédéfinie atteinte	**BOOL**
POSITION	Position de réglage	**DINT**	COMMAND-ABORTED	Commande interrompue	**BOOL**
VELOCITY	Consigne de fréquence	**INT**	ERROR	Erreur dans le module de fonctions	**BOOL**
			ERRORID	Code d'erreur	**INT**
			BUSY	Déplacement Home activé	**BOOL**

ERRORID	**Explication**
0	Pas d'erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé
1201h	Dans les valeurs de consigne PLC, la position High n'est pas saisie (P553)
1202h	Dans les valeurs de consigne PLC, la position Low n'est pas saisie (P553)
1D00h	Les codeurs absolus ne sont pas pris en charge

3.3.3.4 MC_Home (SK 5xxP)

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X						

Permet au variateur de fréquence de démarrer une recherche du point de référence si **EXECUTE** passe de 0 à 1 (front). Le variateur de fréquence se déplace avec la fréquence de consigne saisie dans **VELOCITY**. Si l'entrée est activée par le signal du capteur de point de référence (P420[-xx] = point de référence), une inversion du sens de rotation se produit. En cas de front négatif du signal du capteur de point de référence, la valeur se trouvant dans **POSITION** est reprise. Ensuite, le variateur de fréquence freine à 0Hz, le signal **DONE** passe à 1. Pendant tout le déplacement **HOME**, la sortie **BUSY** est activée.

Si l'opération doit être interrompue (par ex. par un autre module de fonctions MC), **COMMANDABORTED** est défini.

En cas d'erreur, **ERROR** est défini sur 1. **DONE** est dans ce cas 0. Dans **ERRORID**, le code d'erreur correspondant est alors valable.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	BOOL	**DONE**	Position de consigne prédéfinie atteinte	BOOL
POSITION	Consigne de position	DINT	**COMMAND-ABORTED**	Commande interrompue	BOOL
VELOCITY	Consigne de fréquence	INT	**ERROR**	Erreur dans le module de fonctions	BOOL
MODE	ci-après	BYTE	**ERRORID**	Code d'erreur	INT
			BUSY	Déplacement Home activé	BOOL

ERRORID	**Explication**
0	Pas d'erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé
1201h	Dans les valeurs de consigne PLC, la position High n'est pas saisie (P553)
1202h	Dans les valeurs de consigne PLC, la position Low n'est pas saisie (P553)
1D00h	Les codeurs absolus ne sont pas pris en charge
1D01h	Plage de valeurs de l'entrée "Mode" non atteinte ou dépassée (P623)

Mode

Valeur	Explication
1..14	Méthode de point de référence, voir P623
15	Lorsque le capteur du point de référence est atteint, l'entraînement s'inverse. En quittant le capteur du point de référence (front négatif), ceci est repris en tant que point de référence. Le point de référence est ainsi en principe du côté du capteur du point de référence sur lequel la recherche du point de référence a commencé. **Remarque :** Si le capteur du point de référence est "dépassé" (champs de détection du capteur trop fin, vitesse trop élevée), ceci est également repris en tant que point de référence en quittant le capteur du point de référence (front négatif). Le point de référence n'est ainsi en principe pas du côté du capteur du point de référence sur lequel la recherche du point de référence a commencé. (P623 = [15] Méthode Nord 1)
16	Comme 15, cependant un dépassement du capteur du point de référence n'entraîne pas la reprise en tant que point de référence. Ce n'est qu'après une inversion terminée qu'un front négatif entraîne la reprise en tant que point de référence. Le point de référence est ainsi assurément du côté du capteur du point de référence sur lequel l'approche du point de référence a commencé. (P623 = [16] Méthode Nord 2)
17	Lors du dépassement du capteur du point de référence pendant la recherche du point de référence (front positif → front négatif), l'entraînement reprend la moyenne des deux positions et la définit en tant que point de référence. L'entraînement s'inverse et reste sur le point de référence ainsi déterminé. (P623 = [17] Méthode Nord 3)
18..34	Méthode de point de référence, voir P623

3.3.3.5 MC_MoveAbsolute

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Écrit une valeur de consigne de position et de vitesse vers le variateur de fréquence si **EXECUTE** passe de 0 à 1 (front). La consigne de fréquence **VELOCITY** est transférée avec l'échelonnage indiqué dans MC_MoveVelocity.

POSITION :

MODE = False :

La position de consigne est obtenue à partir de la valeur transférée dans **POSITION**.

MODE = True :

La valeur transférée dans **POSITION** correspond à l'index augmenté de 1 du paramètre P613. La position enregistrée dans cet index des paramètres correspond à la position de consigne.

Exemple :

Mode = True ; Position = 12

Le module de fonctions se déplace dans la position qui est dans le jeu de paramètres actuel de P613[-13].

Lorsque le variateur de fréquence atteint la position de consigne, **DONE** est défini sur 1. **DONE** est supprimé avec la réinitialisation de **EXECUTE**. Si **EXECUTE** est supprimé avant d'atteindre la position cible, **DONE** est défini sur 1 pendant un cycle. Pendant le déplacement vers la position de consigne, **BUSY** est activée. Si l'opération doit être interrompue (par ex. par un autre module de fonctions MC), **COMMANDABORTED** est défini. En cas d'erreur, **ERROR** est défini sur 1 et le code d'erreur correspondant est défini dans **ERRORID**. **DONE** est dans ce cas 0. En cas de front négatif sur **EXECUTE**, toutes les sorties sont réinitialisées sur 0.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	**BOOL**	DONE	Position de consigne prédéfinie atteinte	**BOOL**
POSITION	Position de réglage	**DINT**	BUSY	La position de consigne n'est pas atteinte	**BOOL**
VELOCITY	Consigne de fréquence	**INT**	COMMAND-ABORTED	Commande interrompue	**BOOL**
MODE	Le mode source est la position de consigne	**BOOL**	ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			ERRORID	Code d'erreur	**INT**

ERRORID	Explication
0	Pas d'erreur
0x1000	Le VF n'est pas validé
0x1200	Le contrôle de position n'est pas activé
0x1201	Dans les valeurs de consigne PLC, la position High n'est pas saisie (P553)
0x1202	Dans les valeurs de consigne PLC, la position Low n'est pas saisie (P553)

Exemple dans ST :

```
(* L'appareil est validé si DIG1 = TRUE *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* L'appareil est validé et se met en position 20000 avec une fréquence de 50% max.
     Le moteur nécessite un codeur pour cette action et le contrôle de position doit être
     activé. *)
  MoveAbs(Execute := _5_State_digital_input.1, Velocity := 16#2000, Position := 20000);
END_IF
```

3.3.3.6 MC_MoveAdditive

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Correspond exactement à MC_MoveAbsolute à l'exception de l'entrée **DISTANCE**. La position de consigne est obtenue à partir de l'addition de la position de consigne actuelle et de la **DISTANCE** transmise.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	**BOOL**	DONE	Position de consigne prédéfinie atteinte	**BOOL**
DISTANCE	Position de réglage	**DINT**	COMMAND-ABORTED	Commande interrompue	**BOOL**
VELOCITY	Consigne de fréquence	**INT**	ERROR	Erreur dans le bloc fonctionnel	**BOOL**
MODE	Le mode source est la position de consigne	**BOOL**	ERRORID	Code d'erreur	**INT**
			BUSY	La position de consigne n'est pas atteinte	**BOOL**

ERRORID	**Explication**
0	Pas d'erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé
1201h	Dans les valeurs de consigne PLC, la position High n'est pas saisie (P553)
1202h	Dans les valeurs de consigne PLC, la position Low n'est pas saisie (P553)

3.3.3.7 MC_MoveRelative

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Correspond exactement à MC_MoveAbsolute à l'exception de l'entrée **DISTANCE**. La position de consigne est obtenue à partir de l'addition de la position réelle actuelle et de la **DISTANCE** transmise.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	**BOOL**	DONE	Position de consigne prédéfinie atteinte	**BOOL**
DISTANCE	Position de réglage	**DINT**	COMMAND-ABORTED	Commande interrompue	**BOOL**
VELOCITY	Consigne de fréquence	**INT**	ERROR	Erreur dans le bloc fonctionnel	**BOOL**
MODE	Le mode source est la position de consigne	**BOOL**	ERRORID	Code d'erreur	**INT**
			BUSY	La position de consigne n'est pas atteinte	**BOOL**

ERRORID	**Explication**
0	Pas d'erreur
1000h	Le VF n'est pas validé
1200h	Le contrôle de position n'est pas activé
1201h	Dans les valeurs de consigne PLC, la position High n'est pas saisie (P553)
1202h	Dans les valeurs de consigne PLC, la position Low n'est pas saisie (P553)

3.3.3.8 MC_MoveVelocity

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Définit la fréquence de consigne pour le variateur de fréquence si **EXECUTE** passe de 0 à 1 (front). Lorsque le variateur de fréquence atteint la fréquence de consigne, **INVELOCITY** est défini sur 1. Pendant que le VF accélère à la fréquence de consigne, la sortie **BUSY** est activée. Si **EXECUTE** a déjà été défini sur 0, **INVELOCITY** est défini sur 1 uniquement pour un cycle. Si l'opération doit être interrompue (par ex. par un autre module de fonctions MC), **COMMANDABORTED** est défini.

En cas de front négatif sur **EXECUTE**, toutes les sorties sont réinitialisées sur 0.

La saisie de **VELOCITY** est effectuée de façon échelonnée selon la formule suivante :

VELOCITY = (fréquence de consigne (Hz) × 0x4000) / P105

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	**BOOL**	INVELOCITY	Fréquence de consigne prédéfinie atteinte	**BOOL**
VELOCITY	Consigne de fréquence	**INT**	BUSY	La fréquence de consigne n'est pas encore atteinte	**BOOL**
			COMMAND-ABORTED	Commande interrompue	**BOOL**
			ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			ERRORID	Code d'erreur	**INT**

ERRORID	**Explication**
0	Pas d'erreur
1000h	Le VF n'est pas validé
1100h	Le VF n'est pas en mode vitesse (contrôle position activé)
1101h	Pas de fréquence de consigne paramétrée (P553)

Exemple dans AWL :

```
CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* Régler 20 Hz (max. 50 Hz) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute
```

Exemple dans ST :

```
(* Appareil prêt à fonctionner si DIG1 est défini *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* Appareil validé avec 50% de fréquence max. si DIG2 est défini *)
  MoveVelocity(Execute := _5_State_digital_input.1, Velocity := 16#2000);
END_IF
```

3.3.3.9 MC_Power

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Cette fonction permet d'activer ou de désactiver l'étage final de l'appareil. Si l'entrée **ENABLE** est définie sur 1, l'étage final est validé. Pour cela, l'appareil doit se trouver dans l'état "Switch-on Block" ou "Ready for Switch-on". Si l'appareil est dans l'état "Fault" ou "Fault reaction active", le défaut doit d'abord être éliminé et acquitté. Ce n'est qu'ensuite qu'une validation peut être effectuée via ce bloc. Si l'appareil se trouve dans l'état "Not Ready for Switch-on", une connexion n'est pas possible. Dans tous les cas, le bloc fonctionnel passe dans l'état d'erreur et **ENABLE** doit être défini sur 0 pour acquitter l'erreur.

Si l'entrée **ENABLE** est définie sur 0, l'appareil est désactivé. Si ceci se produit lorsque le moteur est en fonctionnement, celui-ci est préalablement abaissé à 0 Hz via la rampe réglée dans P103.

La sortie **STATUS** correspond à 1 lorsque l'étage final de l'appareil est activé ; sinon, il s'agit de 0.

ERROR et **ERRORID** sont réinitialisés si **ENABLE** passe à 0.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	STATUS	Le moteur est sous tension	**BOOL**
			ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			ERRORID	Code d'erreur	**INT**

ERRORID	**Explication**
0	Pas d‘erreur
1001h	Fonction d'arrêt activée
1300h	L’appareil se trouve dans l'état "Ready for Switch-on“ ou "Switch-on Block"

Exemple dans AWL :

```
CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* Régler 20 Hz (max. 50 Hz) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute
```

Exemple dans ST :

```
(* Activer Power Block *)
Power(Enable := TRUE);
IF Power.Status THEN
  (* L’appareil est prêt à la connexion *)
END_IF
```

3.3.3.10 MC_ReadActualPos

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Fournit en continu la position réelle actuelle du variateur de fréquence si **ENABLE** est sur 1. Dès qu'une position réelle valable se trouve à la sortie, **VALID** est défini comme valable. En cas d'erreur, **ERROR** est défini sur 1 et **VALID** est dans ce cas 0.

Position d'échelonnage : 1 tour de moteur = 1000

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	VALID	Sortie incorrecte	**BOOL**
			ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			POSITION	Position réelle actuelle du VF	**DINT**

Exemple dans ST :

```
ReadActualPos(Enable := TRUE);
IF ReadActualPos.Valid THEN
  Pos := ReadActualPos.Position;
END_IF
```

3.3.3.11 MC_ReadParameter

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Lit un paramètre de façon cyclique à partir de l'appareil dans la mesure où **ENABLE** est définie sur 1. Le paramètre lu est enregistré dans Value. Il est valable si **DONE** est défini sur 1. Pendant la lecture, la sortie **BUSY** est définie sur 1. Si **ENABLE** reste sur 1, le paramètre est lu en continu de façon cyclique. Le numéro de paramètre et l'index peuvent être modifiés à tout moment si **ENABLE** est activé. Il est toutefois difficile de déterminer quand la nouvelle valeur sera lue car le signal **DONE** est toujours défini sur 1. Dans ce cas, il est recommandé de définir le signal **ENABLE** pour un cycle sur 0, car le signal **DONE** est ensuite réinitialisé. L'index des paramètres est obtenu à partir de l'index de la documentation moins 1. Ainsi, par ex. P700 Index 3 ("Reason for switch-on block") est interrogé via l'index des paramètres 2. En cas d'erreur, **ERROR** est défini sur 1. **DONE** est dans ce cas 0 et **ERRORID** contient le code d'erreur. Si le signal **ENABLE** est défini sur 0, tous les signaux et **ERRORID** sont supprimés.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	DONE	La valeur est correcte	**BOOL**
PARAMETERNUMBER	Numéro de paramètre	**INT**	ERROR	La lecture a échoué	**BOOL**
PARAMETERINDEX	Index des paramètres	**INT**	BUSY	L'opération n'est pas terminée	**BOOL**
			ERRORID	Code d'erreur	**INT**
			VALUE	Paramètre lu	**DINT**

ERRORID	**Explication**
0	Numéro de paramètre non autorisé
3	Index des paramètres erroné
4	Pas de tableau
201	Élément de commande non valide dans la dernière commande reçue
202	Identifiant de réponse interne non représentable

Exemple dans ST :

```
(* Module Motion FB_ReadParameter *)
ReadParam(Enable := TRUE,Parameternumber := 102, ParameterIndex := 0);
IF ReadParam.Done THEN
  Value := ReadParam.Value;
  ReadParam(Enable := FALSE);
END_IF
```

3.3.3.12 MC_ReadStatus

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Lit l'état de l'appareil. La machine d'état est basée sur la spécification PLCopen "Function blocks for motion control" (blocs fonctionnels pour la commande de mouvement). Tant que **ENABLE** est sur 1, l'état est lu.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	BOOL	VALID	Sortie incorrecte	**BOOL**
			ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			ERRORSTOP	L'appareil a une erreur	**BOOL**
			DISABLED	L'étage final de l'appareil est désactivé	**BOOL**
			STOPPING	Une commande d'arrêt est activée	**BOOL**
			DISCRETEMOTION	L'un des trois modules de fonctions de positionnement est activé	**BOOL**
			CONTINUOUSMOTION	MC_Velocity activé	**BOOL**
			HOMING	MC_Home activé	**BOOL**
			STANDSTILL	L'appareil n'a pas de commande de déplacement activée. Sa vitesse est de 0 tr/min, avec un étage final activé.	**BOOL**

Exemple dans ST :

```
ReadStatus(Enable := TRUE);
IF ReadStatus.Valid THEN
  fError := ReadStatus.ErrorStop;
  fDisable := ReadStatus.Disabled;
  fStopping := ReadStatus.Stopping;
  fInMotion := ReadStatus.DiscreteMotion;
  fInVelocity := ReadStatus.ContinuousMotion;
  fInHome := ReadStatus.Homing;
  fStandStill := ReadStatus.StandStill;
end_if
```

3.3.3.13 MC_Reset

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Réinitialisation d'une erreur dans l'appareil (acquittement de défaut) dans le cas d'un front montant de **EXECUTE**. En cas d'erreur, **ERROR** est défini sur 1 et la cause de l'erreur est saisie dans **ERRORID**. En cas de front négatif sur **EXECUTE** , toutes les erreurs sont réinitialisées.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Démarrage	**BOOL**	DONE	Erreur de l'appareil réinitialisée	**BOOL**
			ERROR	Erreur dans le bloc fonctionnel	**BOOL**
			ERRORID	Code d'erreur	**INT**
			BUSY	Réinitialisation pas encore activée	**BOOL**

ERRORID	**Explication**
0	Pas d'erreur
1001h	Fonction d'arrêt activée
1700h	Une réinitialisation de l'erreur n'a pas pu être effectuée ; la cause de l'erreur est encore présente

Exemple dans ST :

```
Reset(Execute := TRUE);
IF Reset.Done THEN
 (* Der Fehler wurde zurückgesetzt *)
 Reset(Execute := FALSE);
ELSIF Reset.Error THEN
 (* Reset konnte nicht ausgeführt werden, die Ursache für den Fehler liegt noch an *)
 Reset(Execute := FALSE);
END_IF
```

3.3.3.14 MC_Stop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Dans le cas d'un front montant (0 à 1), l'appareil passe dans l'état **STANDINGSTILL**. Toutes les fonctions Motion actuellement activées sont arrêtées. L'appareil ralentit à 0 Hz et désactive l'étage final. Tant que la commande Stop est activée (**EXECUTE** = 1), tous les autres modules de fonctions Motion sont bloqués. La sortie **BUSY** est activée avec le front montant sur **EXECUTE** et reste ainsi jusqu'à ce qu'un front descendant soit effectué sur **EXECUTE**.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Démarrage	**BOOL**	DONE	La commande est exécutée	**BOOL**
			BUSY	La commande est activée	**BOOL**

3.3.3.15 MC_WriteParameter_16 / MC_WriteParameter_32

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Écrit un paramètre 16/32 Bit dans l'appareil si **EXECUTE** passe de 0 à 1 (front). Le paramètre a été écrit si **DONE** est défini sur 1. Pendant la lecture, la sortie **BUSY** est définie sur 1. En cas d'erreur, **ERROR** est défini sur 1 et **ERRORID** contient le code d'erreur. Les signaux **DONE**, **ERROR**, **ERRORID** restent définis jusqu'à ce que **EXECUTE** passe de nouveau sur 0. Si le signal **EXECUTE** passe sur 0, le processus d'écriture n'est pas interrompu. Seul le signal **DONE** reste uniquement défini pour 1 cycle PLC.

VAR_INPUT			**VAR_OUTPUT**		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Validation	BOOL	**DONE**	La valeur est correcte	BOOL
PARAMETERNUMBER	Numéro de paramètre	INT	**BUSY**	L'écriture est activée	BOOL
PARAMETERINDEX	Index des paramètres	INT	**ERROR**	La lecture a échoué	BOOL
VALUE	Valeur à écrire	INT	**ERRORID**	Code d'erreur	INT
RAMONLY	Conservez la valeur dans la RAM (version V2.1)	BOOL			

ERRORID	**Explication**
0	Numéro de paramètre non autorisé
1	Valeur de paramètre non modifiable
2	Limite inférieure ou supérieure de la valeur dépassée
3	Index des paramètres erroné
4	Pas de tableau
5	Type de données non autorisé
6	Seulement réinitialisable (seule la valeur 0 peut être écrite)
7	Élément descriptif non modifiable
201	Élément de commande non valide dans la dernière commande reçue
202	Identifiant de réponse interne non représentable

Exemple dans ST :

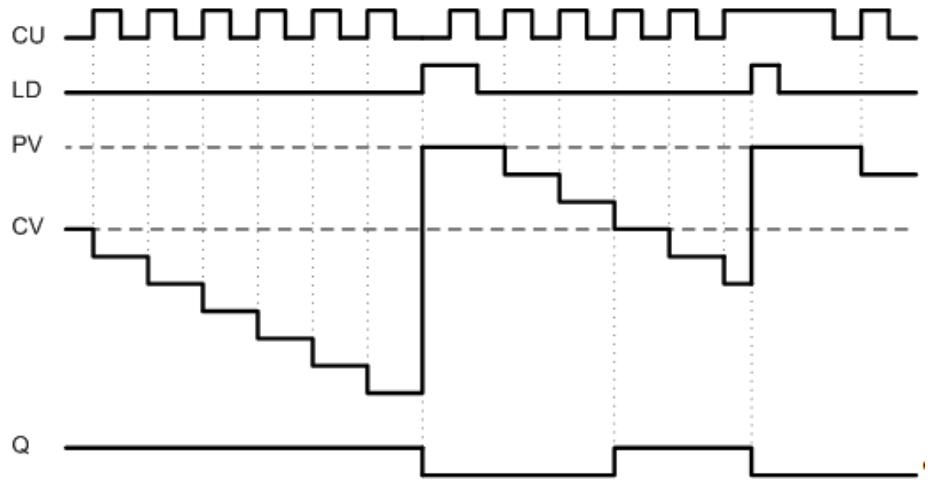
```
WriteParam16(Execute := TRUE, ParameterNumber := 102, ParameterIndex := 0, Value := 300);
  IF WriteParam16.Done THEN
    WriteParam16(Execute := FALSE);
  END_IF;
```

3.3.4 Standard

3.3.4.1 Décompteur CTD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Pour chaque front montant en **CD**, le compteur du bloc fonctionnel **CV** diminue la valeur de un, si CV est supérieur à -32768. Si **CV** est inférieur ou égal à 0, la sortie **Q** reste sur TRUE. Via **LD**, le compteur **CV** peut être défini sur la valeur enregistrée dans **PV**.



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
CD	Entrée de compteur	BOOL	Q	TRUE, si CV <= 0	BOOL
LD	Chargement de la valeur de démarrage	BOOL	CV	État du compteur actuel	INT
PV	Valeur de démarrage	INT			

Exemple dans AWL :

```
LD VarBOOL1
ST CTDInst.CD
LD VarBOOL2
ST CTDInst.LD
LD VarINT1
ST CTDInst.PV
CAL CTDInst
LD CTDInst.Q
ST VarBOOL3
LD CTDInst.CV
ST VarINT2
```

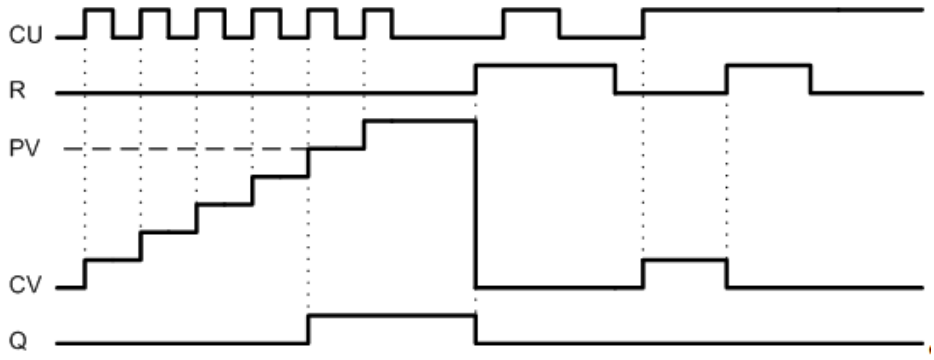
Exemple dans ST :

```
CTDInst(CD := VarBOOL1, LD := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTDInst.Q;
VarINT2 := CTDInst.CV;
```

3.3.4.2 Compteur CTU

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Pour chaque front montant en **CU**, le compteur du bloc fonctionnel **CV** augmente la valeur de un. **CV** peut être compté jusqu'à la valeur 32767. Tant que **CV** est supérieur ou égal à **PV**, la sortie **Q** reste sur TRUE. Via **R**, le compteur **CV** peut être redéfini sur zéro.



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
CU	Entrée de compteur	BOOL	Q	TRUE, si CV >= 0	BOOL
R	Remise à zéro de l'état du compteur	BOOL	CV	État du compteur actuel	INT
PV	Valeur de démarrage	INT			

Exemple dans AWL :

```
LD VarBOOL1
ST CTUInst.CU
LD VarBOOL2
ST CTUInst.R
LD VarINT1
ST CTUInst.PV
CAL CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1)
LD CTUInst.Q
ST VarBOOL3
LD CTUInst.CV
ST VarINT2
```

Exemple dans ST :

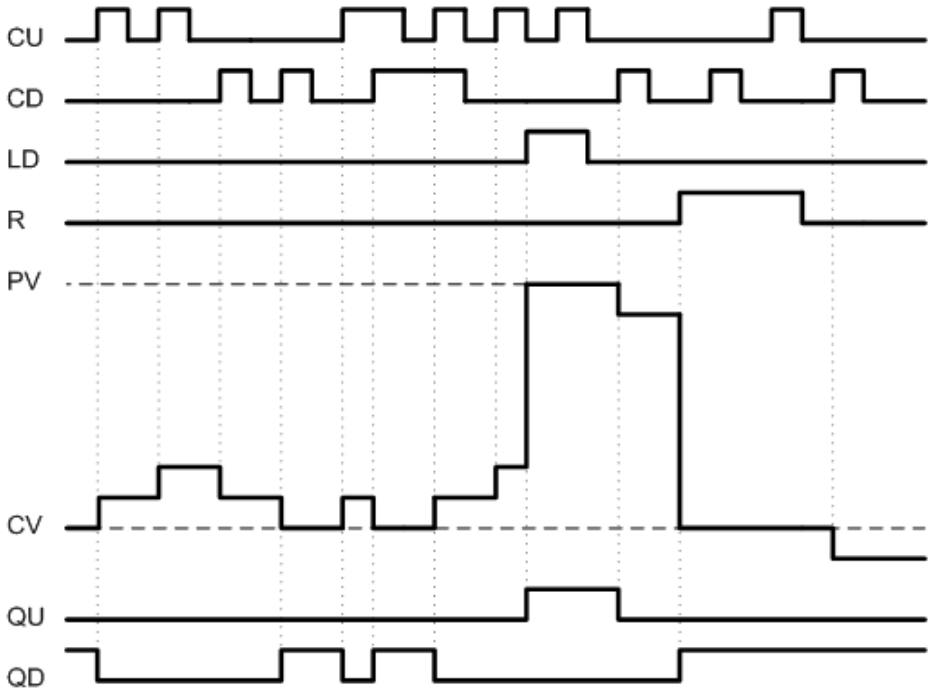
```
CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTUInst.Q;
VarINT2 := CTUInst.CV;
```

3.3.4.3 Compteur et décompteur CTUD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Pour chaque front montant en **CU**, le compteur CV augmente la valeur de un, tant que CV est inférieur à 32767. Pour chaque front montant en **CD**, le compteur **CV** diminue la valeur de un, tant que **CV** est supérieur à -32768. Via **R**, le compteur **CV** peut être défini sur zéro. Via **LD**, la valeur enregistrée dans **PV** est copiée dans **CV**.

R est prioritaire par rapport à **LD**, **CU** et **CV**. **PV** peut être modifié à tout moment, **QU** se base toujours sur la valeur actuellement définie.



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
CU	Comptage	**BOOL**	QU	TRUE, si CV >= PV	**BOOL**
CD	Décomptage	**BOOL**	QD	TRUE, si CV <= 0	**BOOL**
R	Remise à zéro de l'état du compteur	**BOOL**	CV	État du compteur actuel	**INT**
LD	Chargement de la valeur de démarrage	**BOOL**			
PV	Valeur de démarrage	**INT**			

Exemple dans AWL :

```
LD VarBOOL1
ST CTUDInst.CU
LD VarBOOL3
ST CTUDInst.R
LD VarBool4
ST CTUDInst.LD
LD VarINT1
ST CTUInst.PV
CAL CTUDInst
LD CTUDInst.QU
ST VarBOOL5
LD CTUDInst.QD
ST VarBOOL5
LD CTUInst.CV
ST VarINT2
```

Exemple dans ST :

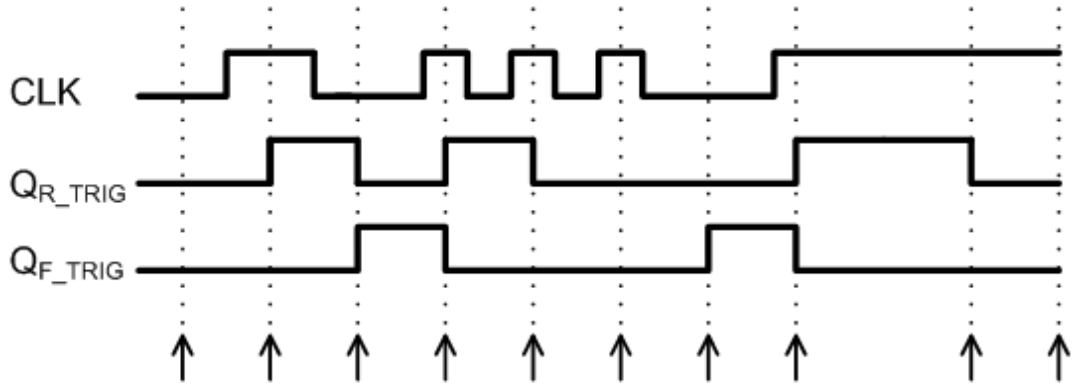
```
CTUDInst(CU:=VarBOOL1, R:=VarBOOL3, LD:=VarBOOL4, PV:=VarINT1);
VarBOOL5 := CTUDInst.QU;
VarBOOL5 := CTUDInst.QD;
VarINT2 := CTUDInst.CV;
```

3.3.4.4 R_TRIG et F_TRIG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Les deux fonctions servent à la détection des fronts. Si un front est détecté sur **CLK**, **Q** est défini sur TRUE jusqu'au prochain appel de fonction, puis de nouveau sur FALSE. Ce n'est qu'après un nouveau front que **Q** peut de nouveau être TRUE pour un cycle.

- R_TRIG = front montant
- F_TRIG = front descendant



Appels de fonctions

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
CLK	Définir	BOOL	Q	Sortie	BOOL

Exemple dans AWL :

```
LD VarBOOL1
ST RTRIGInst.CLK
CAL RTRIGInst
LD RTRIGInst.Q
ST VarBOOL2
```

Exemple dans ST :

```
RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

Informations

La sortie de la fonction est uniquement modifiée lorsque la fonction est exécutée. Par conséquent, il est recommandé d'appeler la détection des fronts en continu avec le cycle PLC.

3.3.4.5 RS Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Fonction bistable ; la sortie **Q1** est définie via **S** et supprimée via **R1**. Si TRUE est présent pour **R1** et **S** en même temps, **R1** est prioritaire.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
S	Définir	**BOOL**	**Q1**	Sortie	**BOOL**
R1	Remise à zéro	**BOOL**			

Exemple dans AWL :

```
LD VarBOOL1
ST RSInst.S
LD VarBOOL2
ST RSInst.R1
CAL RSInst
LD RSInst.Q1
ST VarBOOL3
```

Exemple dans ST :

```
RSInst(S:= VarBOOL1 , R1:=VarBOOL2);
VarBOOL3 := RSInst.Q1;
```

3.3.4.6 SR Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Fonction bistable ; la sortie **Q1** est définie via **S1** et supprimée via **R**. Si TRUE est présent pour **R1** et **S** en même temps, **S1** est prioritaire.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
S1	Définir	**BOOL**	Q1	Sortie	**BOOL**
R	Remise à zéro	**BOOL**			

Exemple dans AWL :

```
LD VarBOOL1
ST SRInst.S1
LD VarBOOL2
ST SRInst.R
CAL RSInst
LD SRInst.Q1
ST VarBOOL3
```

Exemple dans ST :

```
SRInst(S1:= VarBOOL1 , R:=VarBOOL2);
VarBOOL3 := SRInst.Q1;
```

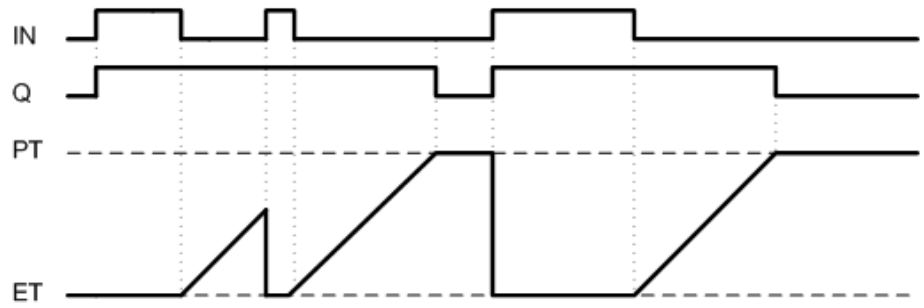
3.3.4.7 Temporisateur de déclenchement TOF

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Si **IN** = TRUE, **Q** est défini sur TRUE. Le temporisateur démarre lorsque **IN** passe sur FALSE. Tant que le temporisateur fonctionne (**ET** < **PT**), **Q** reste sur TRUE. Si (**ET** = **PT**), le temporisateur est arrêté, **Q** devient alors FALSE. Dans le cas d'un nouveau front montant sur **IN**, le temporisateur **ET** est de nouveau défini sur zéro.

Pour une saisie simplifiée, des littéraux peuvent également être utilisés ici, comme par ex.

- LD TIME#50s20ms = 50,020 secondes
- LD TIME#1d30m = 1 jour et 30 minutes



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
IN	Temporisateur activé	BOOL	**Q**	TRUE ß (ET<PT)	BOOL
PT	Durée	DINT	**ET**	État actuel du temporisateur	DINT

Exemple dans AWL :

```
LD VarBOOL1
ST TOFInst.IN
LD DINT#5000
ST TOFInst.PT
CAL TOFInst
LD TOFInst.Q
ST VarBOOL2
```

Exemple dans ST :

```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TOFInst.Q;
```

Informations — **Temporisateur ET**

Le temps **ET** est exécuté indépendamment d'un cycle PLC. Le démarrage du temporisateur avec IN et la définition de la sortie Q sont exécutés lors de l'appel de la fonction "CAL". L'appel de la fonction se déroule dans un cycle PLC. Dans le cas de programmes PLC plus longs, il peut être supérieur à 5 ms de sorte qu'une gigue puisse apparaître.

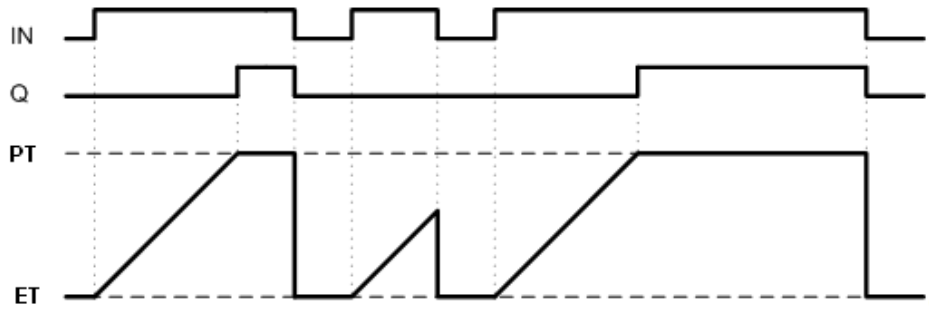
3.3.4.8 Temporisateur d'enclenchement TON

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Si **IN** = TRUE, le temporisateur avance. Si **ET** = PT, **Q** est défini sur TRUE et le temporisateur est arrêté. **Q** reste sur TRUE tant que **IN** est également sur TRUE. Avec un nouveau front montant sur **IN**, le temporisateur redémarre à zéro. **PT** peut être modifié pendant le fonctionnement du temporisateur. La durée est saisie dans **PT** en millisecondes. Ceci permet un retard compris entre 5 ms et 24,8 jours. Comme la base de temps de PLC correspond à 5 ms, le retard minimal est également de 5 ms.

Pour une saisie simplifiée, des littéraux peuvent également être utilisés ici, comme par ex.

- LD TIME#50s20ms = 50,020 secondes
- LD TIME#1d30m = 1 jour et 30 minutes



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
IN	Temporisateur activé	BOOL	**Q**	TRUE ß (IN=TRUE & ET=PT)	BOOL
PT	Durée	DINT	**ET**	État actuel du temporisateur	DINT

Exemple dans AWL :

```
LD VarBOOL1
ST TONInst.IN
LD DINT#5000
ST TONInst.PT
CAL TONInst
LD TONInst.Q
ST VarBOOL2
```

Exemple dans ST :

```
TONInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TONInst.Q;
```

Informations — **Temporisateur ET**

Le temps **ET** est exécuté indépendamment d'un cycle PLC. Le démarrage du temporisateur avec IN et la définition de la sortie Q sont exécutés lors de l'appel de la fonction "CAL". L'appel de la fonction se déroule dans un cycle PLC. Dans le cas de programmes PLC plus longs, il peut être supérieur à 5 ms de sorte qu'une gigue puisse apparaître.

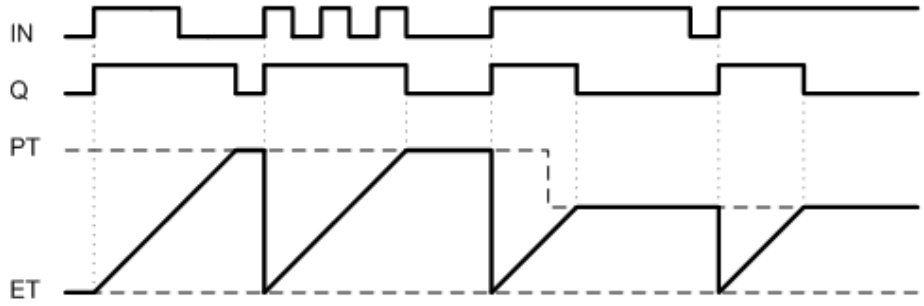
3.3.4.9 Impulsion TP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Dans le cas d'un front positif sur **IN**, le temporisateur est démarré avec la valeur 0. Le temporisateur compte jusqu'à la valeur saisie dans **PT** et s'arrête. Cette opération ne peut pas être interrompue ! PT peut être modifié pendant le comptage. La sortie **Q** est TRUE tant que le temporisateur **ET** est inférieur à **PT**. Si **ET = PT** et qu'un front montant est détecté sur **IN**, le temporisateur est de nouveau démarré avec la valeur 0.

Pour une saisie simplifiée, des littéraux peuvent également être utilisés ici, comme par ex.

- LD TIME#50s20ms = 50,020 secondes
- LD TIME#1d30m = 1 jour et 30 minutes



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
IN	Temporisateur activé	BOOL	**Q**	TRUE ß (ET<PT)	BOOL
PT	Durée	DINT	**ET**	État actuel du temporisateur	DINT

Exemple dans AWL :

```
LD VarBOOL1
ST TPInst.IN
LD DINT#5000
ST TPInst.PT
CAL TPInst
LD TPInst.Q
ST VarBOOL2
```

Exemple dans ST :

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TPInst.Q;
```

Informations — **Temporisateur ET**

Le temps **ET** est exécuté indépendamment d'un cycle PLC. Le démarrage du temporisateur avec IN et la définition de la sortie Q sont exécutés lors de l'appel de la fonction "CAL". L'appel de la fonction se déroule dans un cycle PLC. Dans le cas de programmes PLC plus longs, il peut être supérieur à 5 ms de sorte qu'une gigue puisse apparaître.

3.3.5 Accès aux zones mémoire du variateur de fréquence

S'il est nécessaire d'enregistrer temporairement de grandes quantités de données, de les transmettre à d'autres appareils ou de les faire recevoir par d'autres appareils, l'utilisation des modules FB_WriteTrace et FB_ReadTrace est affichée.

3.3.5.1 FB_ReadTrace

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

À l'aide de ce module de fonctions, différentes zones de mémoire du VF peuvent être lues directement.

Si un front positif est détecté par le module de fonctions au niveau de l'entrée **ENABLE**, tous les paramètres présents pour cette entrée sont repris. L'emplacement de mémoire à lire est marqué par **STARTINDEX** et **MEMORY.** Dans le cas d'une lecture réussie, la sortie **VALID** passe sur 1 et dans **VALUE** se trouve la valeur lue.

Si le module de fonctions est à présent appelé plusieurs fois et que l'entrée **ENABLE** reste sur 1, l'adresse mémoire à lire est augmentée de 1 lors de chaque appel ; le contenu de la nouvelle cellule de mémoire est immédiatement copié dans la sortie **VALUE**.

L'index de mémoire actuel pour l'accès suivant peut être lu sous la sortie **ACTINDEX.** Une fois la fin de la mémoire atteinte, la sortie **READY** passe sur 1 et la lecture est arrêtée.

Des valeurs au format INT ou DINT peuvent être lues. Dans le cas des valeurs INT, seul le composant Low doit être évalué par la sortie **VALUE**. L'affectation est effectuée via l'entrée **SIZE**, un 0 correspond aux valeurs INT et un 1 aux valeurs DINT.

L'affectation des zones mémoire est effectuée via l'entrée MEMORY :

MEMORY = 1 à P613[0-251] correspond aux 504 valeurs INT ou 252 valeurs DINT

MEMORY = 0 à P900[0-247] jusqu'à P906[0-111] correspond aux 3200 valeurs INT ou 1600 valeurs DINT

Le module de fonctions ne peut pas être interrompu par d'autres blocs.

Avec un front négatif sur ENABLE, toutes les sorties sont définies sur 0 et le fonctionnement du module de fonctions est terminé.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Exécuter	**BOOL**	VALID	Lecture réussie	**BOOL**
SIZE	Format de sauvegarde	**BOOL**	READY	La mémoire complète est lue	**BOOL**
MEMORY	Sélection de la zone de mémoire	**BYTE**	ERROR	Une erreur du module de fonctions	**BOOL**
STARTINDEX	Indique la cellule de mémoire à décrire	**INT**	ERRORID	Code d'erreur	**INT**
			ACTINDEX	Index de mémoire actuel à partir duquel la lecture est effectuée au prochain cycle	**INT**
			VALUE	Valeur lue	**DINT**

ERRORID	**Explication**
0	Pas d‘erreur
1A00h	Plage de valeurs STARTINDEX dépassée
1A01h	Plage de valeurs MEMORY dépassée

3.3.5.2 FB_WriteTrace

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

Via ce module de fonctions, des quantités de valeurs individuelles ou de plus grande taille peuvent être provisoirement enregistrées dans le VF. L'enregistrement des valeurs n'est pas effectué de façon durable. Autrement dit, après un redémarrage du VF, les valeurs seront perdues.

Si un front positif est détecté par le module de fonctions au niveau de l'entrée **ENABLE**, tous les paramètres présents pour cette entrée sont repris. La valeur se trouvant dans VALUE est écrite à l'emplacement de mémoire marqué **STARTINDEX** et **MEMORY**. Dans le cas d'une écriture réussie, la sortie VALID passe sur 1.

Si le module de fonctions est appelé maintenant plusieurs fois et que l'entrée **ENABLE** reste sur 1, pour chaque appel du bloc fonctionnel, l'entrée **VALUE** est lue et enregistrée et l'adresse mémoire est augmentée de 1. L'index de mémoire actuel pour l'accès suivant peut être lu sous la sortie **ACTINDEX.** Une fois la fin de la mémoire atteinte, la sortie FULL passe sur 1 et l'enregistrement est arrêté. Si l'entrée **OVERWRITE** est toutefois définie sur 1, l'index de mémoire est de nouveau sur **STARTINDEX** et les valeurs enregistrées préalablement sont écrasées.

Des valeurs au format INT ou DINT peuvent être enregistrées. Dans le cas des valeurs INT, seul le composant Low est évalué par l'entrée **VALUE**. L'affectation est effectuée via l'entrée **SIZE**, un 0 correspond aux valeurs INT et un 1 aux valeurs DINT.

L'affectation des zones mémoire est effectuée via l'entrée MEMORY :

MEMORY = 1 à P613[0-251] correspond aux 504 valeurs INT ou 252 valeurs DINT

MEMORY = 0 à P900[0-247] jusqu'à P906[0-111]correspond aux 3200 valeurs INT ou 1600 valeurs DINT

Le module de fonctions ne peut pas être interrompu par d'autres blocs.

Avec un front négatif sur **ENABLE**, toutes les sorties sont définies sur 0 et le fonctionnement du module de fonctions est terminé.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Exécuter	**BOOL**	VALID	Écriture réussie	**BOOL**
SIZE	Format de sauvegarde	**BOOL**	FULL	La mémoire entière est pleine	**BOOL**
OVERWRITE	Remplacement possible des données en mémoire	**BOOL**	ERROR	Une erreur du module de fonctions	**BOOL**
MEMORY	Sélection de la zone de mémoire	**BYTE**	ERRORID	Code d'erreur	**INT**
STARTINDEX	Indique la cellule de mémoire à décrire	**INT**	ACTINDEX	Index de mémoire actuel sur lequel l'enregistrement est effectué au prochain cycle	**DINT**
VALUE	Valeur à enregistrer	**DINT**			

ERRORID	**Explication**
0	Pas d‘erreur
1A00h	Plage de valeurs STARTINDEX dépassée
1A01h	Plage de valeurs MEMORY dépassée

Informations

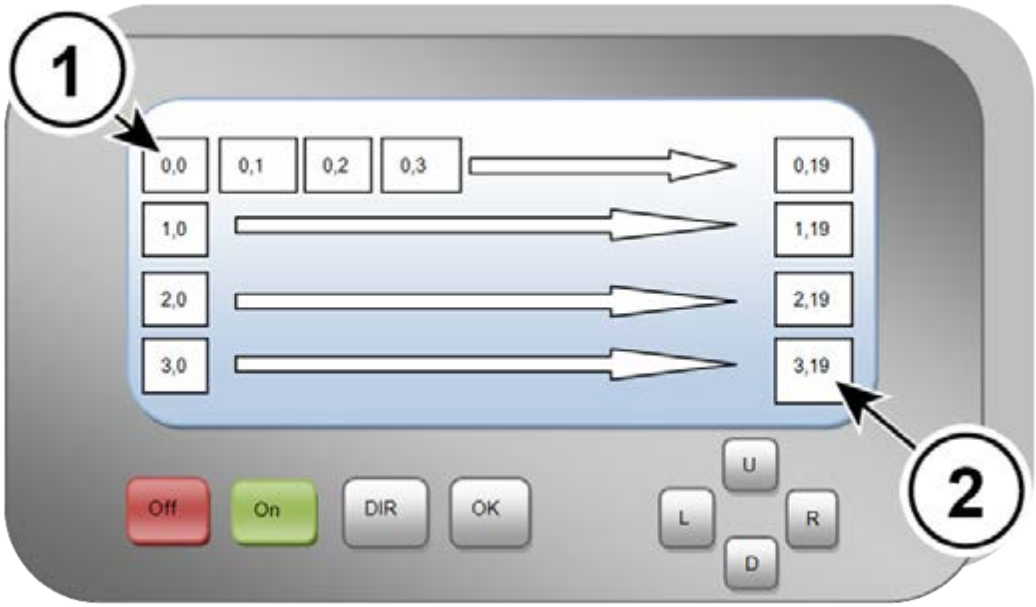
Attention ! La zone de mémoire au paramètre MEMORY = 0 est également utilisée par la fonction Scope. Une utilisation de la fonction Scope écrase les valeurs enregistrées !

3.3.6 Visualisation de la ParameterBox

Dans la ParameterBox, l'écran complet peut être utilisé pour la représentation des informations. Pour cela, la ParameterBox doit être dans le mode de visualisation. Ceci est possible à partir de la version de microprogramme V4.3 de la ParameterBox (paramètre P1308) et se déroule comme suit :

- Sous l'option de menu "Display", régler le paramètre P1003 sur "PLC Display"
- Les touches fléchées de droite et de gauche permettent de passer à l'affichage des paramètres de fonction
- PLC Affichage est à présent activé dans la ParameterBox et reste ainsi de manière durable

Dans le mode de visualisation de la ParameterBox, le contenu de l'écran peut être décrit par le biais des deux modules de fonctions expliqués ci-après. Préalablement, le point "Allow ParameterBox function modules" (Autoriser les modules de fonctions Parameterbox) doit toutefois être activé dans la fenêtre de configuration PLC (bouton ). Par le biais de la valeur de processus "Parameterbox_key_state", l'état du clavier de la Box peut être interrogé en supplément. Ainsi, des entrées dans le programme PLC peuvent être réalisées. La figure suivante présente la structure de l'écran et la position des touches à lire pour la ParameterBox.



1	Premier caractère	(0,0 → Ligne=0, Colonne=0)
2	Dernier caractère	(3,19 → Ligne=3, Colonne=19)

3.3.6.1 Vue d'ensemble de la visualisation

Module de fonctions	Explication
FB_STRINGToPBox	Copie une chaîne dans la ParameterBox
FB_DINTToPBox	Copie une valeur DINT vers la ParameterBox

3.3.6.2 FB_DINTToPBOX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Ce module de fonctions convertit une valeur DINT en chaîne ASCII et copie cette dernière dans la ParameterBox. Le résultat peut être au format décimal, binaire ou hexadécimal. La sélection est effectuée via **MODE**. **ROW** et **COLUMN** permettent de définir la position de départ de la chaîne dans l'écran de la ParameterBox. Le paramètre **LENGTH** transmet la longueur de la chaîne en caractères. Dans le **MODE** décimal, le paramètre **POINT** positionne une virgule dans le nombre à représenter. Dans **POINT**, le nombre de caractères apparaissant à droite de la virgule est indiqué. La fonction **POINT** est désactivée avec le réglage 0. Si le nombre contient plus de caractères que la longueur autorisée et qu'aucune virgule n'est placée, le dépassement est indiqué par le caractère "**#**". Si une virgule se trouve dans le nombre, tous les chiffres après la virgule peuvent être omis en cas de besoin. En MODE hexadécimal et binaire, les bits aux valeurs faibles sont toujours représentés si la longueur paramétrée est trop courte. Tant que **ENABLE** est sur 1, toutes les modifications au niveau des entrées sont immédiatement reprises. Si **VALID** passe sur 1, la chaîne a alors été correctement transmise. En cas d'erreur, **ERROR** est défini sur 1. **VALID** est dans ce cas 0. Dans **ERRORID**, le code d'erreur correspondant est alors valable. En cas de front négatif sur **ENABLE**, la réinitialisation de **VALID**, **ERROR** et **ERRORID** est effectuée.

Exemples :

Réglage	Nombre à afficher	Affichage P-Box
Length = 5 **Point = 0**	12345	12345
Length = 5 **Point = 0**	-12345	#####
Length = 10 **Point = 3**	123456789	123456,789
Length = 8 **Point = 3**	123456789	123456,7

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Transfert de la chaîne	**BOOL**	VALID	Chaîne transmise	**BOOL**
MODE	Format de représentation 0 = Décimal 1 = Binaire 2 = Hexadécimal Plage de valeurs= 0 à 2	**BYTE**	ERROR	Erreur dans le module de fonctions	**BOOL**
ROW	Ligne de l'écran Plage de valeurs = 0 à 3	**BYTE**	ERRORID	Code d'erreur	**INT**
COLUMN	Colonne de l'écran Plage de valeurs = 0 à 19	**BYTE**			
POINT	Position de la virgule Plage de valeurs = 0 à 10 0 = la fonction est désactivée	**BYTE**			
LENGTH	Longueur en sortie Plage de valeurs = 1 à 11	**BYTE**			
VALUE	Nombre à sortir	**DINT**			

ERRORID	**Explication**
0	Pas d‘erreur
1500h	La chaîne écrase la zone mémoire du tableau de la ParameterBox
1501h	À l'entrée LINE, la plage de valeurs a été dépassée
1502h	À l'entrée ROW, la plage de valeurs a été dépassée
1504h	À l'entrée POINT, la plage de valeurs a été dépassée
1505h	À l'entrée LENGTH, la plage de valeurs a été dépassée
1506h	À l'entrée MODE, la plage de valeurs a été dépassée

Exemple dans ST :

```
(* Initialisation *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Vérifier la position réelle *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Position in der PBox anzeigen (PBox P1003 = PLC Anzeige ) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Activer ou désactiver l'appareil via DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* L'appareil est-il activé ? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;
```

3.3.6.3 FB_STRINGToPBOX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Ce module de fonctions copie une chaîne (de caractères) dans le bloc mémoire de la ParameterBox. **ROW** et **COLUMN** permettent de définir la position de départ de la chaîne dans l'écran de la ParameterBox. Le paramètre **TEXT** transmet la chaîne souhaitée au module de fonctions, le nom de la chaîne se trouve dans le tableau de variables. Tant que **ENABLE** est sur 1, toutes les modifications au niveau des entrées sont immédiatement reprises. Dans le cas d'une entrée **CLEAR** définie, le contenu complet de l'écran est remplacé par des espaces avant d'écrire la chaîne sélectionnée. Si **VALID** passe sur 1, la chaîne a alors été correctement transmise. En cas d'erreur, **ERROR** est défini sur 1. **VALID** est dans ce cas 0. Dans **ERRORID**, le code d'erreur correspondant est alors valable. En cas de front négatif sur **ENABLE**, la réinitialisation de **VALID**, **ERROR** et **ERRORID** est effectuée.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Transfert de la chaîne	**BOOL**	VALID	Chaîne transmise	**BOOL**
CLEAR	Effacer l'écran	**BOOL**	ERROR	Erreur dans le module de fonctions	**BOOL**
ROW	Ligne de l'écran Plage de valeurs = 0 à 3	**BYTE**	ERRORID	Code d'erreur	**INT**
COLUMN	Colonne de l'écran Plage de valeurs = 0 à 19	**BYTE**			
TEXT	Texte à afficher	**STRING**			

ERRORID	**Explication**
0	Pas d'erreur
1500h	La chaîne écrase la zone mémoire du tableau de la ParameterBox
1501h	À l'entrée ROW, la plage de valeurs a été dépassée
1502h	À l'entrée COLUMN, la plage de valeurs a été dépassée
1503h	Le numéro de chaîne sélectionné n'existe pas
1506h	Dans la configuration PLC, l'option "Allow ParameterBox function modules" (Autoriser les modules de fonctions Parameterbox) n'est pas activée.

Exemple dans ST :

```
(* Initialisation *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Vérifier la position réelle *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Afficher la position dans la PBox (PBox P1003 = PLC Affichage ) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Activer ou désactiver l'appareil via DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* L'appareil est-il activé ? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;
```

3.3.7 FB_Capture (Enregistrement d'événements rapides)

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

Le temps de cycle de PLC correspond à 5 ms, ce cycle est parfois trop long pour enregistrer des événements externes très rapides. Via FB Capture, il est possible d'enregistrer certaines valeurs physiques sur des fronts au niveau des entrées du VF. La surveillance des entrées est effectuée dans un cycle de 1 ms. Les valeurs ainsi enregistrées peuvent être lues ultérieurement par le PLC.

Dans le cas d'un front positif sur **EXECUTE**, toutes les entrées sont lues et la fonction Capture est activée. L'entrée **INPUT** permet de sélectionner le VF à surveiller. Via **EDGE**, le type de front et le comportement du module sont sélectionnés.

EDGE = 0 Avec le premier front positif, la valeur sélectionnée est enregistrée sous **OUTPUT1** et **DONE1** est défini sur 1. Le front positif suivant enregistre sous **OUTPUT2** et **DONE2** est défini sur 1. Le module de fonctions est ensuite désactivé.

EDGE = 1 Comportement identique à **EDGE** = 0, sauf que le front négatif est le déclencheur.

EDGE = 2 Avec le premier front positif, la valeur sélectionnée est enregistrée sous **OUTPUT1** et **DONE1** est défini sur 1. Le front négatif suivant enregistre sous **OUTPUT2** et **DONE2** est défini sur 1. Le module de fonctions est ensuite désactivé.

EDGE = 3 Comportement identique à **EDGE** = 2, sauf que le front négatif est d'abord le déclencheur, puis le front positif.

Si l'entrée **CONTINUOUS** est définie sur 1, seul le réglage 0 et 1 est encore important pour **EDGE**. Le module de fonctions continue à fonctionner et enregistre toujours le dernier événement à déclencher sous **OUTPUT1**. **DONE1** reste activé à partir du premier événement. **DONE2** et **OUTPUT2** ne sont pas utilisés.

La sortie **BUSY** reste active jusqu'à ce que les deux événements Capture (**DONE1** et **DONE2**) se soient produits.

La fonction du module peut être terminée à tout moment par un front négatif sur **EXECUTE**. Toutes les sorties conservent leurs valeurs. Avec un front positif sur **EXECUTE**, toutes les sorties sont tout d'abord supprimées, puis la fonction du module est démarrée.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	BOOL	**DONE1**	Valeur valable dans OUTPUT1	BOOL
CONTINUOUS	Exécution unique ou fonctionnement continu	BOOL	**DONE2**	Valeur valable dans OUTPUT2	BOOL
INPUT	**SK54xE** Entrée à surveiller 0 = entrée 1 - - - - 7 = entrée 8 **SK52xE, SK53xE, SK2xxE, SK2xx-EFDS** Entrée à surveiller 0 = entrée 1 - - - - 3 = entrée 4	BYTE	**BUSY**	Le module de fonctions attend encore les événements Capture	BOOL
EDGE	Front à déclencher	BYTE	**ERROR**	Une erreur du module de fonctions	BOOL
SOURCE	Taille à enregistrer 0 = position en tours 1 = fréquence réelle 2 = couple	BYTE	**ERRORID**	Code d'erreur	INT
			OUTPUT1	Valeur pour le premier événement Capture	DINT
			OUTPUT2	Valeur pour le deuxième événement Capture	DINT

ERRORID	Explication
0	Pas d‘erreur
1900h	Plage de valeurs INPUT dépassée
1901h	Plage de valeurs EDGE dépassée
1902h	Plage de valeurs SOURCE dépassée
1903h	Plus de deux instances sont actives

Exemple dans ST :

```
Power(ENABLE := TRUE);
IF Power.STATUS THEN
    Move(EXECUTE := TRUE, POSITION := Pos, VELOCITY := 16#2000);
    (* Le module de fonctions Capture FB attend au niveau de DIG1 un signal élevé (High Signal).
    S’il est détecté, le module de fonctions enregistre la position réelle. À l’aide de la
    caractéristique "OUTPUT1", la valeur peut être interrogée. *)
    Capture(EXECUTE := TRUE, INPUT := 0);

    IF Capture.DONE1 THEN
      Pos := Capture.OUTPUT1;
      Move(EXECUTE := FALSE);
    END_IF;
END_IF;
```

Informations

Plusieurs instances de ce module de fonctions peuvent exister dans le programme PLC. Mais seulement deux instances peuvent être activées au même moment !

3.3.8 FB_DinCounter

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	ab V2.3	ab V3.1	ab V2.1	X	ab V1.1	

Ce module de fonctions sert à compter les impulsions via les entrées digitales. Tous les fronts (Low – High et High – Low) sont comptés. La largeur d'impulsions minimale est de 250µs.

Le module de fonctions est activé via ENABLE. Avec le front positif, les entrées PV, UD, DIN et MODE sont reprises et toutes les sorties sont supprimées.

UD définit le sens de comptage

- 0 = grand comptage
- 1 = petit comptage

Dans PV, une valeur du compteur peut être saisie. Selon la définition de l'entrée MODE, le résultat est différent.

MODE

- 0 = dépassement, le compteur fonctionne en tant que compteur permanent. Le dépassement est possible dans le sens positif et négatif. Lors du démarrage de la fonction, CV = PV est défini. Dans ce mode, BUSY reste toujours sur 1 et Q toujours sur 0.
- 1 = sans dépassement
  - Le comptage croissant à CV démarre à 0, BUSY = 1, et se poursuit jusqu'à CV=>PV. BUSY passe ensuite sur 0 et Q sur 1. Le comptage s'arrête.
  - Le comptage décroissant à CV démarre avec PV et se poursuit jusqu'à CV<=0. Pendant cette durée, BUSY = 1 et passe sur 0 lorsque la fin du comptage est atteinte. En contrepartie, Q passe sur 1.
  - Le redémarrage du compteur est atteint via un nouveau front sur l'entrée ENABLE

DIN définit l'entrée de mesure. Le nombre d'entrée dépend du VF correspondant (jusqu'à 4 pièces).

- Entrée 1 = 0
- Entrée 2 = 1
- Entrée 3 = 2
- Entrée 4 = 3

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Validation	**BOOL**	Q	Comptage terminé	**BOOL**
UD	Sens de comptage 0 = grand comptage 1 = petit comptage	**BOOL**	BUSY	Le compteur tourne	**BOOL**
PV	Valeur du compteur	**INT**	ERROR	Une erreur du module de fonctions	**BOOL**
MODE	Mode	**BYTE**	ERRORID	Code d'erreur	**INT**
DIN	Entrée de mesure	**BYTE**	CV	Valeur du compteur	**INT**
			CF	Fréquence de comptage (résolution 0,1)	**INT**

ERRORID	**Explication**
0	Pas d‘erreur
0x1E00	L’entrée digitale est déjà utilisée par un autre compteur
0x1E01	L’entrée digitale n'existe pas
0x1E02	Plage de valeurs MODE dépassée

3.3.9 FB_FunctionCurve

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	

Ce module de fonctions exécute une commande cartographique. Des points définis peuvent être transmis au bloc fonctionnel par le biais desquels il émule une fonction. La sortie se comporte ensuite conformément au diagramme caractéristique enregistré. Une interpolation linéaire est effectuée entre les différents points de base. Les points de base sont définis avec les valeurs X et Y. Les valeurs X sont pour cela toujours de type **INT**, les valeurs Y peuvent être soit de type **INT** ou **DINT** selon la taille du point de base supérieur. Si **DINT** est utilisé, l'espace disque requis est plus important. Les points de base sont saisis dans la fenêtre des variables, dans la colonne "Init Value". Si à l'entrée **ENABLE** un TRUE a été détecté, la valeur de sortie correspondante **OUTVALUE** est calculée à l'aide de la valeur d'entrée **INVALUE**. **VALID** signale avec un TRUE que la valeur de sortie **OUTVALUE** est valable. Tant que **VALID** est FALSE, la sortie **OUTVALUE** a la valeur 0. Si la valeur d'entrée **INVALUE** dépasse l'extrémité supérieure ou inférieure du diagramme caractéristique, la première ou la dernière valeur de sortie du diagramme caractéristique reste à la sortie jusqu'à ce que **INVALUE** se trouve de nouveau dans la zone du diagramme caractéristique. En cas de dépassement ou de sous-dépassement du diagramme caractéristique, la sortie correspondante **MINLIMIT** ou **MAXLIMIT** est définie sur TRUE. **ERROR** devient TRUE lorsque les abscisses (valeurs X) du diagramme caractéristique n'augmentent pas en continu ou si aucun tableau n'est initialisé. Une erreur correspondante est alors émise via **ERRORID** et la valeur de sortie devient 0. L'erreur est réinitialisée lorsque **ENABLE** = FALSE.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Exécuter	**BOOL**	VALID	Valeur de sortie incorrecte	**BOOL**
INVALUE	Valeur d'entrée (x)	**INT**	ERROR	Erreur dans le module de fonctions	**BOOL**
			ERRORID	Code d'erreur	**INT**
			MAXLIMIT	Limite maximale atteinte	**BOOL**
			MINLIMIT	Limite minimale atteinte	**BOOL**
			OUTVALUE	Valeur de sortie (y)	**DINT**

ERRORID	**Explication**
0	Pas d'erreur
1400h	Abscisses (valeurs X) du diagramme caractéristique pas toujours croissantes
1401h	Aucun diagramme caractéristique initialisé

3.3.10 FB_PIDT1

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

P-I-DT1 est un régulateur discret qui peut être librement paramétré. Si certains composants individuels ne sont pas nécessaires, comme le P, le I ou le composant DT1, leurs paramètres sont décrits avec 0. Le composant T1 fonctionne uniquement en combinaison avec le composant D. Un régulateur PT1 ne peut donc pas être paramétré. En raison de la limite de mémoire interne, les paramètres de régulation sont limités aux domaines suivants :

Plage de valeurs autorisée pour les paramètres de régulation			
Paramètre	**Plage de valeurs**	**Échelonnage**	**Plage de valeurs obtenue**
P (Kp)	0 – 32767	1/100	0,00 – 32,767
I (Ki)	0 – 10240	1/100	0,00 – 10,240
D (Kd)	0 – 32767	1/1000	0,000 – 3,2767
T1 (ms)	0 – 32767	1/1000	0,000 – 3,2767
Max	-32768 – 32767		
Min	-32768 – 32767		

Si l'entrée **ENABLE** est définie sur TRUE, le régulateur commence à calculer. Les paramètres de régulation sont repris uniquement en cas de front montant de **ENABLE.** Pendant que **ENABLE** est sur TRUE, une modification des paramètres de régulation reste sans effet. Si **ENABLE** est sur FALSE, la sortie reste sur la dernière valeur.

Le bit de sortie **VALID** est défini tant que la valeur de sortie Q se déplace dans les limites min. et max. et que l'entrée **ENABLE** est sur TRUE.

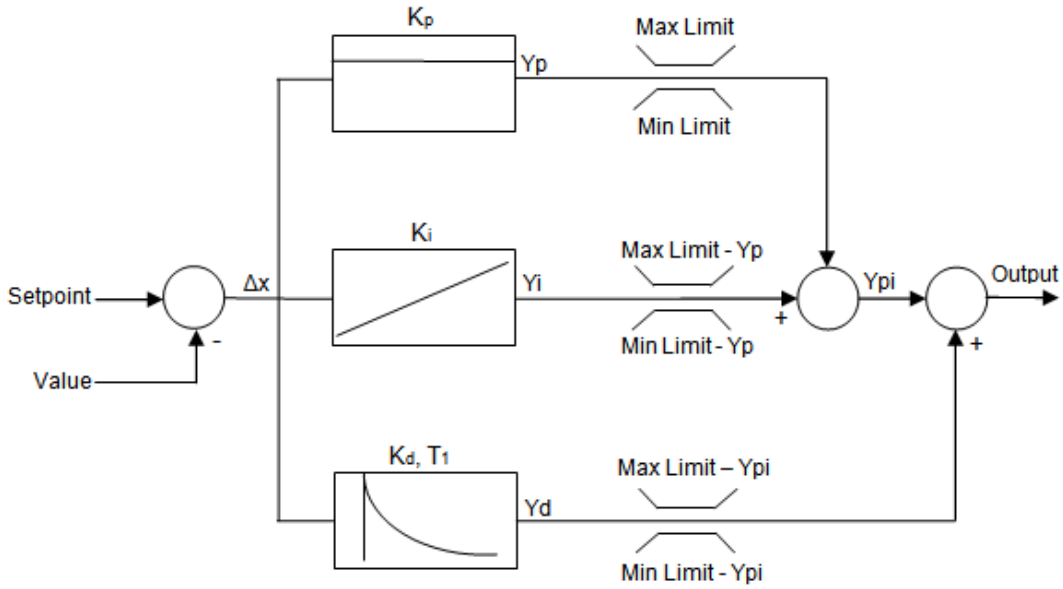
ERROR est défini dès qu'une erreur apparaît. Le bit **VALID** est alors FALSE et la cause de l'erreur doit être détectée via **ERRORID** (voir le tableau ci-après).

Si le bit **RESET** est défini sur TRUE, le contenu de l'intégrateur et du différenciateur est défini sur 0. Si l'entrée **ENABLE** est sur FALSE, la sortie **OUTPUT** est également définie sur 0. Si l'entrée **ENABLE** est définie sur TRUE, seul le composant P agit sur la sortie **OUTPUT**.

Si la valeur de sortie **OUTPUT** dépasse les valeurs de sortie maximales et minimales, le bit correspondant **MAXLIMIT** ou **MINLIMIT** est défini et le bit **VALID** est défini sur FALSE.

Informations

Si le programme complet ne peut pas être traité au cours d'un cycle PLC, le régulateur calcule la valeur de sortie une seconde fois avec les anciens échantillons. Une fréquence d'échantillonnage constante est ainsi obtenue. Par conséquent, il est nécessaire d'exécuter la commande CAL pour le régulateur PIDT1 dans chaque cycle PLC et uniquement à la fin du programme PLC !



VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
ENABLE	Exécuter	**BOOL**	VALID	Valeur de sortie incorrecte	**BOOL**
RESET	Réinitialiser les valeurs de sortie	**BOOL**	ERROR	Erreur dans le module de fonctions	**BOOL**
P	Composant P (Kp)	**INT**	ERRORID	Code d'erreur	**INT**
I	Composant I (Ki)	**INT**	MAXLIMIT	Limite maximale atteinte	**BOOL**
D	Composant D (Kd)	**INT**	MINLIMIT	Limite minimale atteinte	**BOOL**
T1	Composant T1 en ms	**INT**	OUTPUT	Valeur de sortie	**INT**
MAX	Valeur de sortie maximale	**INT**			
MIN	Valeur de sortie minimale	**INT**			
SETPOINT	Valeur de consigne	**INT**			
VALUE	Valeur réelle	**INT**			

ERRORID	**Explication**
0	Pas d‘erreur
1600h	Composant P pas compris dans la plage de valeurs
1601h	Composant I pas compris dans la plage de valeurs
1602h	Composant D pas compris dans la plage de valeurs
1603h	Composant T1 pas compris dans la plage de valeurs

3.3.11 FB_ResetPostion

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	ab V2.3	ab V3.1	ab V2.1	X	ab V1.2	

Dans le cas d'un front sur l'entrée **EXECUTE**, la position réelle est définie sur la valeur saisie dans la position. Dans le cas de codeurs absolus, la position réelle est uniquement réinitialisée sur 0. La valeur dans la position n'est pas utilisée.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	**BOOL**			
Position	Position	**DINT**			

3.3.12 FB_Weigh

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	à partir de V2.3	à partir de V3.1	à partir de V2.1	X	à partir de V1.2	

Ce module sert à déterminer le couple moyen pendant un déplacement avec une vitesse constante. À partir de cette valeur, des valeurs physiques comme le poids déplacé peuvent par ex. être déterminées.

Le module de fonctions est démarré par le biais d'un front positif sur l'entrée **EXECUTE.** Avec le front, toutes les entrées sont reprises par le module de fonctions. Le VF se déplace avec la vitesse définie sous **SPEED**. Une fois le temps défini sous **STARTTIME** écoulé, la mesure commence. La durée de mesure est définie sous **MEASURETIME**. Une fois le temps de mesure écoulé, le VF s'arrête. Si l'entrée **REVERSE** = 1, la mesure démarre de nouveau avec toutefois la vitesse inversée. Sinon, la mesure est terminée, la sortie **DONE** passe sur 1 et dans **VALUE** se trouve le résultat de mesure.

Tant que la mesure est en cours, **BUSY** est activé.

L'échelonnage du résultat de mesure **VALUE** est 1 = 0,01% du couple nominal du moteur.

L'appel d'un autre module de fonctions Motion arrête la fonction de mesure et la sortie **ABORT** passe sur 1.

Toutes les sorties du module de fonctions sont réinitialisées avec un nouveau front positif sur **EXECUTE**.

VAR_INPUT			VAR_OUTPUT		
Entrée	**Explication**	**Type**	**Sortie**	**Explication**	**Type**
EXECUTE	Exécuter	**BOOL**	DONE	Mesure terminée	**BOOL**
REVERSE	Changement du sens de rotation	**BOOL**	BUSY	Mesure en cours	**BOOL**
STARTTIME	Temps jusqu'au début de la mesure en ms	**INT**	ABORT	Mesure interrompue	**BOOL**
MEASURETIME	Temps de mesure en ms	**INT**	ERROR	Une erreur du module de fonctions	**BOOL**
SPEED	Vitesse de mesure en % (échelonnée sur la fréquence maximale, 16#4000 correspond à 100%)	**INT**	ERRORID	Code d'erreur	**INT**
			VALUE	Résultat de mesure	**INT**

ERRORID	**Explication**
0	Pas d'erreur
0x1000	Le VF n'est pas en service
0x1101	La fréquence de consigne ne doit pas être paramétrée en tant que valeur de consigne (P553)
0x1C00	Plage de valeurs STARTTIME dépassée
0x1C01	Plage de valeurs MEASURETIME dépassée
0x1C02	La tolérance des valeurs de mesure les unes par rapport aux autres est supérieure à 1/8

Exemple dans ST :

```
(* Valider l'objet *)
Power(Enable := TRUE);
(* L'appareil est-il validé ? *)
if Power.Status then
  (* Définir le temps de démarrage à 2000 ms *)
  Weigh.STARTTIME := 2000;
  (* Définir le temps de mesure à 1000 ms *)
  Weigh.MEASURETIME := 1000;
  (* Définir la vitesse à 25% de la vitesse maximale *)
  Weigh.SPEED := 16#1000;
end_if;

Weigh(EXECUTE := Power.Status);
(* La pesée est-elle terminée ? *)
if Weigh.done then
  Value := Weigh.Value;
end_if;
```

Informations

Seule une instance de ce module de fonctions est autorisée dans le programme PLC !

3.4 Opérateurs

3.4.1 Opérateurs arithmétiques

Informations

Certains des opérateurs suivants peuvent également comprendre d'autres commandes. Celles-ci doivent être indiquées entre parenthèses après l'opérateur. Veiller à ce qu'un espace se trouve après la parenthèse ouvrante. La parenthèse fermante doit être placée sur une ligne de programme distincte.

```
LD Var1
ADD( Var2
SUB Var3
)
```

3.4.1.1 ABS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données			X	X

Forme la valeur absolue à partir de l'accumulateur

Exemple dans AWL :

```
LD -10 (* Charge la valeur -10 *)
ABS (* Accumulateur = 10 *)
ST Value1 (* Enregistre la valeur 10 dans Value1 *)
```

Exemple dans ST :

```
Value1 := ABS(-10); (* Le résultat est 10 *)
```

3.4.1.2 ADD et ADD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Ajoute des variables et des constantes ensemble, avec les signes corrects. La première valeur pour l'addition se trouve dans l'accumulateur et la seconde est chargée avec la commande ADD ou se trouve entre les parenthèses. Plusieurs variables ou constantes peuvent également être ajoutées à la commande ADD. Dans le cas de la parenthèse d'addition, l'accumulateur est ajouté avec le résultat de la partie entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs à ajouter doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 10
ADD 204 (* Addition de deux constantes *)
ST Value
LD 170 (* Addition d'une constante et de 2 variables. *)
ADD Var1, Var2 (* 170dez + Var1 + Var2 *)
ST Value
LD Var1
ADD( Var2
SUB Var3 (* Var1 + ( Var2 - Var3 ) *)
)
ST Value
```

Exemple dans ST :

```
Résultat := 10 + 30; (* Le résultat est 40 *)
Résultat := 10 + Var1 + Var2;
```

3.4.1.3 DIV et DIV(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Divise l'accumulateur par l'opérande. Dans le cas de divisions par zéro, le résultat maximal possible est indiqué dans l'accumulateur, par ex. dans le cas d'une division avec des valeurs INT, c'est la valeur 0x7FFF ou si le diviseur est négatif, c'est la valeur 0x8000. Dans le cas de la parenthèse de division, l'accumulateur est divisé par le résultat de la partie entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs à diviser doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 10
DIV 3 (* Division de deux constantes *)
ST iValue (* Le résultat est 9 *)
LD 170 (* Division d'une constante et de 2 variables. *)
DIV Var1, Var2 (* (170dez : Var1) : Var2 *)
ST Value
LD Var1 (* Divise Var1 par le contenu des parenthèses *)
DIV( Var2
SUB Var3
) (* Var1 : ( Var2 - Var3 ) *)
ST Value
```

Exemple dans ST :

```
Résultat := 30 / 10; (* Le résultat est 3 *)
Résultat := 30 / Var1 / Var2;
```

3.4.1.4 LIMIT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

La commande limite la valeur de l'accumulateur aux valeurs min. et max. transférées. La valeur max. est saisie dans l'accumulateur si elle est dépassée, tout comme la valeur min. si celle-ci n'est pas atteinte. Il n'y a aucun effet si la valeur est comprise dans les limites.

Exemple dans AWL :

```
LD 10 (* Charge la valeur 10 dans l'accumulateur *)
LIMIT 20, 30 (* La valeur est comparée avec les limites 20 et 30. *)
(* Si la valeur dans l'accumulateur est inférieure, l'accumulateur est remplacé par 20*)
ST iValue (* Enregistre la valeur 20 dans Value1 *)
```

Exemple dans ST :

```
Résultat := Limit(10, 20, 30); (* Le résultat est 20 *)
```

3.4.1.5 MAX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Cette commande détermine la valeur maximale de deux variables ou constantes. Pour cela, le contenu de l'accumulateur actuel est comparé avec la valeur transmise dans la commande MAX. Après la commande, la valeur la plus grande des deux est dans l'accumulateur. Les deux valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 100 (* Chargement de 100 dans l'accumulateur *)
MAX 200 (* Comparaison avec la valeur 200 *)
ST iValue (* Enregistrement de 200 dans Value2 (car c'est la valeur la plus grande) *)
```

Exemple dans ST :

```
Résultat := Max(100, 200); (* Le résultat est 200 *)
```

3.4.1.6 MIN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Cette commande détermine la valeur minimale de deux variables ou constantes. Pour cela, le contenu de l'accumulateur actuel est comparé avec la valeur transmise dans la commande MIN. Après la commande, la valeur la plus petite des deux est dans l'accumulateur. Les deux valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 100 (* Chargement de 100 dans l'accumulateur *)
MIN 200 (* Comparaison avec la valeur 200 *)
ST Value2 (* Enregistrement de 100 dans Value2 (car c'est la valeur la plus petite) *)
```

Exemple dans ST :

```
Résultat := Min(100, 200); (* Enregistrement de 100 dans Value2 (car c'est la valeur la plus petite) *)
```

3.4.1.7 MOD et MOD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

L'accumulateur est divisé par une ou plusieurs variables ou constantes ; le reste de la division se trouve dans l'accumulateur en tant que résultat. Dans le cas de la parenthèse modulo, l'accumulateur est divisé par le résultat de la partie entre parenthèses, à partir duquel le modulo est formé. Jusqu'à 6 niveaux de parenthèses sont possibles.

Exemple dans AWL :

```
LD 25 (* Chargement du dividende *)
MOD 20 (* Division 25/20 à modulo = 5 *)
ST Var1 (* Enregistrement du résultat 5 dans Var1 *)
LD 25 (* Chargement du dividende *)
MOD( Var1 (* Résultat = 25/(Var1 + 10) à modulo dans l'accumulateur *)
ADD 10
)
ST Var3 (* Enregistrement du résultat 10 dans Var3 *)
```

Exemple dans ST :

```
Résultat := 25 MOD 20; (* Enregistrement du résultat 5 dans Var1 *)
Résultat := 25 MOD (Var1 + 10); (* Résultat = 25/(Var1 + 10) à modulo dans l'accumulateur *)
```

3.4.1.8 MUL et MUL(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Multiplication de l'accumulateur avec une ou plusieurs variables ou constantes. Dans le cas de la parenthèse de multiplication, l'accumulateur est multiplié avec le résultat de la partie entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les deux valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 25 (* Chargement du multiplicateur *)
MUL Var1, Var2 (* 25 * Var1 * Var2 *)
ST Var2 (* Enregistrement du résultat *)
LD 25 (* Chargement du multiplicateur *)
MUL( Var1 (* Résultat = 25*(Var1 + Var2) *)
ADD Var2
ST Var3 (* Enregistrement du résultat en tant que variable Var3 *)
)
```

Exemple dans ST :

```
Résultat := 25 * Var1 * Var2;
Résultat := 25 * (Var1 + Var2);
```

3.4.1.9 MUX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Par le biais d'un index qui se trouve devant la commande dans l'accumulateur, différentes constantes ou variables peuvent être sélectionnées. La première valeur est atteinte via l'index 0. La valeur sélectionnée est chargée dans l'accumulateur. Le nombre de valeurs est seulement limité par la capacité de la mémoire de programme.

Exemple dans AWL :

```
LD 1 (* Sélection de l'élément souhaité *)
MUX 10,20,30,40,Value1 (* Commande MUX avec 4 constantes et une variable *)
ST Value (* Enregistrement du résultat = 20 *)
```

Exemple dans ST :

```
Résultat := Mux(1, 10, 20, 30, 40, Value1) (* Enregistrement du résultat = 20 *)
```

3.4.1.10 SUB et SUB(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Soustraction de l'accumulateur avec une ou plusieurs variables ou constantes. Dans le cas de la parenthèse de soustraction, l'accumulateur est soustrait avec le résultat de la partie entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs à soustraire doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 10
SUB Var1 (* Résultat = 10 - Var1 *)
Résultat ST
LD 20
SUB Var1, Var2, 30 (* Résultat = 20 - Var1 - Var12 - 30 *)
Résultat ST
LD 20
SUB( 6 (* Soustraction de 20 avec le contenu des parenthèses *)
AND 2
) (* Résultat = 20 - (6 AND 2) *)
Résultat ST (* Résultat = 18 *)
```

Exemple dans ST :

```
Résultat := 10 - Value1 ;
```

3.4.2 Opérateurs mathématiques étendus

Informations

Les opérateurs cités ici sont caractérisés par un calcul intensif. Des durées d'exécution du programme PLC nettement plus longues peuvent en résulter.

3.4.2.1 COS, ACOS, SIN, ASIN, TAN, ATAN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Type de données				X

Calcul de la fonction mathématique correspondante. La valeur à calculer doit être dans l'accumulateur en minutes d'arc. La mise à l'échelle correspond à 1 = 1000.

Conversion : Angle en radian = (angle en degré * PI / 180)*1000.
Par ex. un angle de 90° est converti ainsi à 90° * 3.14 / 180) *1000 = 1571

$$AE = \sin\left(\frac{AE}{1000}\right) \cdot 1000 \qquad AE = \cos\left(\frac{AE}{1000}\right) \cdot 1000 \qquad AE = \tan\left(\frac{AE}{1000}\right) \cdot 1000$$

Exemple dans AWL :

```
LD 1234
SIN
Résultat ST (* Résultat = 943 *)
```

Exemple dans ST :

```
Résultat := COS(1234); (* Résultat= 330 *)
Résultat := ACOS(330); (* Résultat = 1234 *)
Résultat := SIN(1234); (* Résultat = 943 *)
Résultat := ASIN(943); (* Résultat = 1231 *)
Résultat := TAN(999); (* Résultat = 1553 *)
Résultat := ATAN(1553); (* Résultat = 998 *)
```

3.4.2.2 EXP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Type de données				X

Forme à partir de l'accumulateur la fonction exponentielle pour la base du nombre d'Euler (2,718). 3 décimales peuvent être indiquées, autrement dit, 1,002 doit être saisi comme 1002.

$$AE = e^{\left(\frac{AE}{1000}\right)} \cdot 1000$$

Exemple dans AWL :

```
LD 1000
EXP
Résultat ST (* Résultat = 2718 *)
```

Exemple dans ST :

```
Résultat := EXP(1000); (* Résultat = 2718 *)
```

3.4.2.3 LN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Type de données				X

Logarithme de base e (2,718). 3 décimales peuvent être indiquées, autrement dit, 1,000 doit être saisi comme 1000.

$$AE = \ln\left(\frac{AE}{1000}\right) \cdot 1000$$

Exemple dans AWL :

```
LD 1234
LN
Résultat ST
```

Exemple dans ST :

```
Résultat := LN(1234); (* Résultat = 210 *)
```

3.4.2.4 LOG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Type de données				X

Forme à partir de l'accumulateur le logarithme de base 10. 3 décimales peuvent être indiquées, autrement dit, 1,000 doit être saisi comme 1000.

$$AE = log_{10}\left(\frac{AE}{1000}\right) \cdot 1000$$

Exemple dans AWL :

```
LD 1234
LOG
Résultat ST (* Résultat = 91 *)
```

Exemple dans ST :

```
Résultat := LOG(1234); (* Résultat = 91 *)
```

3.4.2.5 SQRT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Type de données				X

Forme à partir de l'accumulateur la racine carrée. 3 décimales peuvent être indiquées, autrement dit, 1,000 doit être saisi comme 1000.

$$AE = \sqrt{\left(\frac{AE}{1000}\right)} \cdot 1000$$

Exemple dans AWL :

```
LD 1234
SQRT
Résultat ST (* Résultat = 1110 *)
```

Exemple dans ST :

```
Résultat := SQRT(1234); (* Résultat = 1110 *)
```

3.4.3 Opérateurs binaires

3.4.3.1 AND et AND(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Opération ET bit à bit de l'AE/l'accumulateur avec une ou deux variables ou constantes. Opération ET(…) bit à bit avec l'AE/l'accumulateur et l'AE/l'accumulateur formé préalablement entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Toutes les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 170
AND 204 (* Opération AND entre 2 constantes *)
(* Accumulateur = 136 (voir l'exemple après le tableau) *)

LD 170 (* Opération entre une constante et 2 variables.*)
AND Var1, Var2 (* Accumulateur = 170déc AND Var1 AND Var2 *)

LD Var1
AND ( Var2 (* AE/Accumulateur = Var1 AND ( Var2 OR Var3 ) *)
OR Var3
)
```

Exemple dans ST :

```
Résultat := 170 AND 204; (* Résultat = 136déc *)
```

Var2	Var1	Résultat
0	0	0
0	1	0
1	0	0
1	1	1

Exemple : 170déc (1010 1010bin) AND 204déc (1100 1100bin) = (1000 1000bin) 136déc

3.4.3.2 ANDN et ANDN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Opération ET bit à bit de l'AE/l'accumulateur avec un opérande inversé. Opération ET(…) bit à bit avec l'AE/l'accumulateur et le résultat inversé des parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 2#0000_1111
ANDN 2#0011_1010 (* Opération ANDN entre 2 constantes *)
(* Accumulateur = 2#1111_0101 *)

LD 170 (* Opération entre une constante et 2 variables. *)
ANDN Var1, Var2 (* Accumulateur = 170d ANDN Var1 ANDN Var2 *)

LD Var1
ANDN ( Var2 (* AE/Accumulateur = Var1 ANDN ( Var2 OR Var3 ) *)
OR Var3
)
```

Var2	Var1	Résultat
0	0	1
0	1	1
1	0	1
1	1	0

Exemple : 170déc (1010 1010bin) AND 204déc (1100 1100bin) = (1000 1000bin) 136déc

3.4.3.3 NOT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Négation bit à bit de l'accumulateur.

Exemple dans AWL :

```
LD BYTE#10 (* Chargement de la valeur 10déc dans l'accumulateur au format Byte *)
NOT (* La valeur est résolue au niveau du bit (0000 1010), *)
(* inversée bit à bit (1111 0101) et reconvertie en valeur décimale *)
(* résultat = 245déc *)
ST Var3 (* Enregistrement du résultat en tant que variable Var3 *)
```

Exemple dans ST :

```
Résultat := not BYTE#10; (* Résultat = 245déc *)
```

3.4.3.4 OR et OR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Opération OU bit à bit de l'AE/l'accumulateur avec une ou deux variables ou constantes. Opération OU(…) bit à bit avec l'AE/l'accumulateur et l'AE/l'accumulateur formé préalablement entre parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Toutes les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 170
OR 204 (* Opération OR entre 2 constantes *)

LD 170 (* Opération entre une constante et 2 variables. *)
OR Var1, Var2 (* Accumulateur = 170d OR Var1OR Var2 *)

LD Var1
OR ( Var2 (* AE/Accumulateur = Var1 OR ( Var2 AND Var3 ) *)
AND Var3
)
```

Exemple dans ST :

```
Résultat := 170 or 204; (* Résultat = 238 *)
```

Var2	Var1	Résultat
0	0	0
0	1	1
1	0	1
1	1	1

3.4.3.5 ORN et ORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Opération OU bit à bit de l'AE/l'accumulateur avec un opérande inversé. Opération OU(…) bit à bit avec l'AE/l'accumulateur et le résultat inversé des parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 2#0000_1111
ORN 2#0011_1010 (* Opération ORN entre 2 constantes *)
(* Accumulateur = 2#1100_0000 *)

LD 170 (* Opération entre une constante et 2 variables. *)
ORN Var1, Var2 (* Accumulateur = 170d ORN Var1 ORN Var2 *)

LD Var1
ORN ( Var2 (* AE/Accumulateur = Var1 ORN ( Var2 OR Var3 ) *)
OR Var3
)
```

Exemple dans ST :

```
Résultat := 2#0000_1111 ORN 2#0011_1010; (* Résultat = 2#1100_0000 *)
```

Var2	Var1	Résultat
0	0	1
0	1	0
1	0	0
1	1	0

3.4.3.6 ROL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Rotation à gauche de bit à bit de l'accumulateur. Le contenu de l'accumulateur est décalé de n vers la gauche, le bit de gauche étant réintroduit à droite.

Exemple dans AWL :

```
LD 175    (* Charge la valeur 1010_1111*)
ROL 2     (* Le contenu de l'accumulateur effectue 2 rotations vers la gauche *)
ST Value1 (* Enregistre la valeur 1011_1110 *)
```

Exemple dans ST :

```
Résultat := ROL(BYTE#175, 2); (* Résultat = 2#1011_1110 *)
Résultat := ROL(INT#175, 2); (* Résultat = 16#C02B *)
```

3.4.3.7 ROR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Rotation à droite de bit à bit de l'accumulateur. Le contenu de l'accumulateur est décalé de n vers la droite, le bit de droite étant réintroduit à gauche.

Exemple dans AWL :

```
LD 175    (* Charge la valeur 1010_1111*)
ROR 2     (* Le contenu de l'accumulateur effectue 2 rotations vers la droite *)
ST Value1 (* Enregistre la valeur 1110_1011 *)
```

Exemple dans ST :

```
Résultat := ROR(BYTE#175, 2); (* Résultat = 2#1110_1011 *)
```

3.4.3.8 S et R

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Mise à un et remise à zéro d'une variable booléenne lorsque le résultat de l'opération précédent (l'AE) était TRUE.

Exemple dans AWL :

```
LD TRUE    (* Charge l'AE avec TRUE *)
S Var1     (* VAR1 définie sur TRUE *)
R Var1     (* VAR1 définie sur FALSE *)
```

Exemple dans ST :

```
Résultat := TRUE;
Résultat := FALSE;
```

3.4.3.9 SHL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Décalage à gauche de bit à bit de l'accumulateur. Le contenu de l'accumulateur est décalé de n vers la gauche, les bits sortis sont perdus.

Exemple dans AWL :

```
LD 175     (* Charge la valeur 1010_1111 *)
SHL 2      (* Le contenu de l'accumulateur est décalé 2 fois vers la gauche *)
ST Value1  (* Enregistre la valeur 1011_1100 *)
```

Exemple dans ST :

```
Résultat := SHL(BYTE#175, 2); (* Résultat = 2#1011_1100 *)
Résultat := SHL(INT#175, 2); (* Résultat = 16#2BC *)
```

3.4.3.10 SHR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Décalage à droite de bit à bit de l'accumulateur. Le contenu de l'accumulateur est décalé de n vers la droite, les bits sortis sont perdus.

Exemple dans AWL :

```
LD 175    (* Charge la valeur 1010_1111 *)
SHR 2     (* Le contenu de l'accumulateur est décalé 2 fois vers la droite *)
ST Value1 (* Enregistre la valeur 0010_1011 *)
```

Exemple dans ST :

```
Résultat := SHR(BYTE#175, 2); (* Résultat = 2#0010_1011 *)
```

3.4.3.11 XOR et XOR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Opération "exclusive OR" bit à bit entre l'AE/l'accumulateur et une à deux variables ou constantes. La première valeur se trouve dans l'AE/l'accumulateur et la seconde est chargée avec la commande ou se trouve entre parenthèses. Les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 2#0000_1111
XOR 2#0011_1010 (* Opération XOR entre 2 constantes *)
                  (* Accumulateur = 2#0011_0101 *)

LD 170          (* Opération entre une constante et 2 variables. *)
XOR Var1, Var2  (* Accumulateur = 170d XOR Var1 XOR Var2 *)

LD Var1
XOR ( Var2      (* AE/Accumulateur = Var1 XOR ( Var2 OR Var3 ) *)
OR Var3
)
```

Exemple dans ST :

```
Résultat := 2#0000_1111 XOR 2#0011_1010; (* Résultat = 2#0011_0101 *)
```

Var2	Var1	Résultat
0	0	0
0	1	1
1	0	1
1	1	0

3.4.3.12 XORN et XORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Opération OU exclusif bit à bit de l'AE/l'accumulateur avec un opérande inversé. Opération OU exclusif (…) bit à bit avec l'AE/l'accumulateur et le résultat inversé des parenthèses. Jusqu'à 6 niveaux de parenthèses sont possibles. Les valeurs doivent appartenir au même type de variable.

Exemple dans AWL :

```
LD 2#0000_1111
XORN 2#0011_1010 (* Opération XORN entre 2 constantes *)
                 (* Accumulateur = 2#1100_1010 *)

LD 170           (* Opération entre une constante et 2 variables. *)
XORN Var1, Var2  (* Accumulateur = 170d XORN Var1 XORN Var2 *)

LD Var1
XORN ( Var2      (* AE/Accumulateur = Var1 XORN ( Var2 OR Var3 ) *)
OR Var3
)
```

Exemple dans ST :

```
Résultat := 2#0000_1111 XORN 2#0011_1010; (* Résultat = 2#1100_1010 *)
```

Var2	Var1	Résultat
0	0	1
0	1	0
1	0	0
1	1	1

3.4.4 Opérateurs de chargement et d'enregistrement

3.4.4.1 LD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Charge une constante ou une variable dans l'AE ou l'accumulateur.

Exemple dans AWL :

```
LD 10 (* Charge 10 pour BYTE *)
LD -1000 (* Charge -1000 pour INT *)
LD Value1 (* Charge la variable Value1 *)
```

3.4.4.2 LDN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Charge un booléen inversé dans l'AE.

Exemple dans AWL :

```
LDN Value1 (* Value1 = TRUE à AE = FALSE *)
ST Value2 (* Enregistrement sur Value2 = FALSE *)
```

3.4.4.3 ST

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X	X	X	X

Enregistre le contenu de l'AE / de l'accumulateur sur une variable. La variable à enregistrer doit correspondre au type de données préalablement chargé et traité.

Exemple dans AWL :

```
LD 100 (* Charge la valeur 1010_1111 *)
ST Value1 (* Le contenu de l'accumulateur 100 est enregistré dans Value1 *)
```

3.4.4.4 STN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Enregistre le contenu de l'AE sur une variable et l'inverse. La variable à enregistrer doit correspondre au type de données préalablement chargé et traité.

Exemple dans AWL :

```
LD Value1 (* Value1 = TRUE à AE = TRUE *)
STN Value2 (* Enregistrement sur Value2 = FALSE *)
```

3.4.5 Opérateurs de comparaison

3.4.5.1 EQ

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si les valeurs sont égales, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1 (* Value1 = 5 *)
EQ 10 (* AE = 5 égal à 10 ? *)
JMPC NextStep (* AE = FALSE à Pas de saut du programme *)
ADD 1
NextStep:
ST Value1
```

Exemple dans ST :

```
(* La valeur est-elle = 10 ? *)
if Value = 10 then
  Value2 := 5;
  end_if;
```

3.4.5.2 GE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si la valeur dans l'accumulateur est supérieure ou égale à la variable ou à la constante, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1 (* Value1 = 5 *)
GE 10 (* 5 est supérieur ou égal à 10 ? *)
JMPC NextStep (* AE = FALSE à Pas de saut du programme *)
ADD 1

NextStep:
ST Value1
```

Exemple dans ST :

```
(* 5 est supérieur ou égal à 10 ? *)
if Value >= 10 then
  Value := Value - 1
  end_if;
```

3.4.5.3 GT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si la valeur dans l'accumulateur est supérieure à la variable ou à la constante, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1(* Value1 = 12 *)
GT 8 (* 12 est supérieur à 8 ? *)
JMPC NextStep (* AE = TRUE - Saut du programme *)
ADD 1
NextStep:
ST Value1
```

Exemple dans ST :

```
(* 12 est supérieur à 8 ? *)
if Value > 8 then
  Value := 0;
end_if;
```

3.4.5.4 LE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si la valeur dans l'accumulateur est inférieure ou égale à la variable ou à la constante, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1 (* Value1 = 5 *)
LE 10 (* 5 est inférieur ou égal à 10 ? *)
JMPC NextStep:
ST Value1
```

Exemple dans ST :

```
(* Value est-elle inférieure ou égale à 10 ?*)
if Value <= 10 then
  Value := 11;
  end_if;
```

3.4.5.5 LT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si la valeur dans l'accumulateur est inférieure à la variable ou à la constante, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1 (* Value1 = 12 *)
LT 8 (* 12 est inférieur à 8 ? *)
JMPC NextStep (* AE = FALSE à Pas de saut du programme *)
ADD 1
NextStep:
ST Value1
```

Exemple dans ST :

```
(* La valeur est-elle inférieure à 0 ?*)
if Value < 0 then
  Value := 0;
  end_if;
```

3.4.5.6 NE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X	X	X

Compare le contenu de l'accumulateur avec une variable ou une constante. Si la valeur dans l'accumulateur est différente de la variable ou de la constante, l'AE est défini sur TRUE.

Exemple dans AWL :

```
LD Value1 (* Value1 = 5 *)
NE 10 (*5 est différent de 10 ?*)
JMPC NextStep (* AE = FALSE à Saut du programme *)
ADD 1
NextStep:
ST Value1
```

Exemple dans ST :

```
if Value <> 5 then
Value := 5;
end_if;
```

3.5 Valeurs de processus

Toutes les entrées et sorties analogiques et digitales ou la valeur de consigne de bus et réelle peuvent être lues et traitées ultérieurement par le PLC ou définies par le PLC (s'il s'agit d'une valeur de sortie). L'accès aux valeurs individuelles est effectué via les valeurs de processus énoncées ci-après. Pour toutes les valeurs de sortie, la sortie (par ex. des sorties digitales ou la valeur de consigne PLC) doit être programmée de sorte que le PLC soit prévu en tant que source d'événement. Toutes les données de processus sont lues par le PLC à chaque déroulement cyclique au début de l'appareil et sont seulement écrites dans l'appareil à la fin du programme PLC ! Les tableaux suivants présentent toutes les valeurs auxquelles la fonction PLC peut directement accéder. L'accès à toutes les autres valeurs de processus doit être réalisé via les blocs fonctionnels MC_ReadParameter ou MC_WriteParameter.

3.5.1 Entrées et sorties

Les valeurs de processus décrivant l'interface E/S de l'appareil sont résumées ici.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_0_Set_digital_output	Définition des sorties digitales	Bit 0 : MFR1 Bit 1 : MFR2 Bit 2 : DOUT 1 Bit 3 : DOUT 2 Bit 4 : DOUT 1 CU5-MLT Bit 5 : DOUT 2 CU5-MLT Bit 6 : DOUT 3 CU5-MLT Bit 7 : DOUT 4 CU5-MLT Bit 8 : fonct. dig. AOUT Bit 9 : libre Bit 10 : BusES Bit0 Bit 11 : BusES Bit1 Bit 12 : BusES Bit2 Bit 13 : BusES Bit3 Bit 14 : BusES Bit4 Bit 15 : BusES Bit5	UINT	R/W	SK 5xxP
_0_Set_digital_output	Définition des sorties digitales	Bit 0 : MFR1 Bit 1 : MFR2 Bit 2 : DOUT1 Bit 3 : DOUT2 Bit 4 : fonct. dig. AOUT Bit 5 : DOUT3 (Din7) Bit 6 : Mot état bit 10 Bit 7 : Mot état bit 13 Bit 8 : BusES Bit0 Bit 9 : BusES Bit1 Bit 10 : BusES Bit2 Bit 11 : BusES Bit3 Bit 12 : BusES Bit4 Bit 13 : BusES Bit5 Bit 14 : BusES Bit6 Bit 15 : BusES Bit7	UINT	R/W	SK 54xE
_0_Set_digital_output	Définition des sorties	Bit 0 : MFR1	UINT	R/W	SK 52xE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
	digitales	Bit 1 : MFR2 Bit 2 : DOUT1 Bit 3 : DOUT2 Bit 4 : fonct. dig. AOUT Bit 5 : libre Bit 6 : Mot état bit 10 Bit 7 : Mot état bit 13 Bit 8 : BusES Bit0 Bit 9 : BusES Bit1 Bit 10 : BusES Bit2 Bit 11 : BusES Bit3 Bit 12 : BusES Bit4 Bit 13 : BusES Bit5 Bit 14 : BusES Bit6 Bit 15 : BusES Bit7			SK 53xE
_0_Set_digital_output	Définition des sorties digitales	Bit 0 : DOUT1 Bit 1 : BusES Bit0 Bit 2 : BusES Bit1 Bit 3 : BusES Bit2 Bit 4 : BusES Bit3 Bit 5 : BusES Bit4 Bit 6 : BusES Bit5 Bit 7 : BusES Bit6 Bit 8 : BusES Bit7 Bit 9 : Bus PZD Bit 10 Bit 10 : Bus PZD Bit 13 Bit 11 : DOUT2	UINT	R/W	SK 2xxE SK 2xxE-FDS
_0_Set_digital_output	Définition des sorties digitales	Bit 0 : DOUT1 Bit 1 : DOUT2 Bit 2 : BusES Bit0 Bit 3 : BusES Bit1 Bit 4 : BusES Bit2 Bit 5 : BusES Bit3 Bit 6 : BusES Bit4 Bit 7 : BusES Bit5 Bit 8 : BusES Bit6 Bit 9 : BusES Bit7 Bit 10 : Bus PZD Bit 10 Bit 11 : Bus PZD Bit 13	UINT	R/W	SK 180E SK 190E
_0_Set_digital_output	Définition des sorties digitales	Bit 0 : DOUT1 Bit 1 : DOUT2 Bit 2 : DOUT_BRAKE Bit 3 : DOUT_BUS1 Bit 4 : DOUT_BUS2	UINT	R/W	SK 155E-FDS SK 175E-FDS
_1_Set_analog_output	Définition de la sortie analogique du VF	10,0V = 100	BYTE	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE
_2_Set_external_	Définition de la sortie	10,0V = 100	BYTE	R/W	SK 5xxP

Nom	Fonction	Échelonnage	Type	Accès	Appareil
analog_out1	analogique 1 IOE				SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_3_Set_external_ analog_out2	Définition de la sortie analogique 2 IOE	10,0V = 100	BYTE	R/W	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_4_State_digital_output	État des sorties digitales	Bit 0 : MFR1 Bit 1 : MFR2 Bit 2 : DOUT 1 Bit 3 : DOUT 2 Bit 4 : DOUT 1 CU5-MLT Bit 5 : DOUT 2 CU5-MLT Bit 6 : DOUT 3 CU5-MLT Bit 7 : DOUT 4 CU5-MLT Bit 8 : fonct. dig. AOUT Bit 9 : libre Bit 10 : DOUT1 IOE1 Bit 11 : DOUT2 IOE1 Bit 12 : DOUT1 IOE2 Bit 13 : DOUT2 IOE2 Bit 14 : libre Bit 15 : libre	INT	R	SK 5xxP
_4_State_digital_output	État des sorties digitales	Bit 0 : MFR1 Bit 1 : MFR2 Bit 2 : DOUT1 Bit 3 : DOUT2 Bit 4 : fonct. dig. AOUT Bit 5 : DOUT3 (Din7) Bit 6 : Mot état bit 8 Bit 7 : Mot état bit 9 Bit 8 : BusES Bit0 Bit 9 : BusES Bit1 Bit 10 : BusES Bit2 Bit 11 : BusES Bit3 Bit 12 : BusES Bit4 Bit 13 : BusES Bit5 Bit 14 : BusES Bit6 Bit 15 : BusES Bit7	INT	R	SK 54xE
_4_State_digital_output	État des sorties digitales	P711	BYTE	R	SK 52xE SK 53xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_4_State_digital_output	État des sorties digitales	Bit 0 : DOUT1 Bit 1 : DOUT2 Bit 2 : DOUT_BRAKE Bit 3 : DOUT_BUS1 Bit 4 : DOUT_BUS2	BYTE	R	SK 155E-FDS SK 175E-FDS
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6 Bit 6 : DIN1 CU5-MLT Bit 7 : DIN2 CU5-MLT Bit 8 : DIN3 CU5-MLT Bit 9 DIN4 CU5-MLT Bit 10 libre Bit 11 libre Bit 12 : Fonction digitale AIN1 Bit 8 : Fonction digitale AIN2	INT	R	SK 5xxP
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6 Bit 6 : DIN7 Bit 7 : Fonction digitale AIN1 Bit 8 : Fonction digitale AIN2	INT	R	SK 54xE
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6 Bit 6 : DIN7	INT	R	SK 52xE SK 53xE
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : libre Bit 5 : Sonde CTP Bit 6 : libre Bit 7 : libre Bit 8 : DIN1 IOE 1 Bit 9 : DIN2 IOE 1 Bit 10 : DIN3 IOE 1	INT	R	SK 2xxE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
		Bit 11 : DIN4 IOE 1 Bit 12 : DIN1 IOE 2 Bit 13 : DIN2 IOE 2 Bit 14 : DIN3 IOE 2 Bit 15 : DIN4 IOE 2			
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : AIN1 Bit 4 : AIN2 Bit 5 : Sonde CTP Bit 6 : libre Bit 7 : libre Bit 8 : DIN1 IOE 1 Bit 9 : DIN2 IOE 1 Bit 10 : DIN3 IOE 1 Bit 11 : DIN4 IOE 1 Bit 12 : DIN1 IOE 2 Bit 13 : DIN2 IOE 2 Bit 14 : DIN3 IOE 2 Bit 15 : DIN4 IOE 2	INT	R	SK 180E SK 190E
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : CTP (sonde) Bit 4 : DIN-BUS1 (ASil1) Bit 5 : DIN-BUS2 (ASil2) Bit 6 : DIN-BUS3 (ASil3) Bit 7 : DIN-BUS4 (ASil4) Bit 8 : BDDI1 (ASIO3) Bit 9 : BDDI2 (ASIO4) Bit 10 : STO	INT	R	SK 155E-FDS SK 175E-FDS
_5_State_Digital_input	État des entrées digitales	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6/AIN1 Bit 6 : DIN7/AIN2 Bit 7 : Sonde CTP Bit 8 : DIN1 IOE 1 Bit 9 : DIN2 IOE 1 Bit 10 : DIN3 IOE 1 Bit 11 : DIN4 IOE 1 Bit 12 : DIN1 IOE 2 Bit 13 : DIN2 IOE 2 Bit 14 : DIN3 IOE 2 Bit 15 : DIN4 IOE 2	INT	R	SK 2xxE-FDS
_6_Delay_digital_inputs	État des entrées digitales selon P475	Bit 0 : DIN1	INT	R	SK 5xxP

Nom	Fonction	Échelonnage	Type	Accès	Appareil
		Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6 Bit 6 : DIN7 Bit 7 : Fonction digitale AIN1 Bit 8 : Fonction digitale AIN2			SK 54xE
_6_Delay_digital_inputs	État des entrées digitales selon P475	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6 Bit 6 : DIN7	INT	R	SK 52xE SK 53xE
_6_Delay_digital_inputs	État des entrées digitales selon P475	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : AIN1 Bit 4 : AIN2 Bit 5 : Sonde CTP Bit 6 : libre Bit 7 : libre Bit 8 : DIN1 IOE 1 Bit 9 : DIN2 IOE 1 Bit 10 : DIN3 IOE 1 Bit 11 : DIN4 IOE 1 Bit 12 : DIN1 IOE 2 Bit 13 : DIN2 IOE 2 Bit 14 : DIN3 IOE 2 Bit 15 : DIN4 IOE 2	INT	R	SK 2xxE SK 180E SK 190E
_6_Delay_digital_inputs	État des entrées digitales selon P475	Bit 0 : DIN1 Bit 1 : DIN2 Bit 2 : DIN3 Bit 3 : DIN4 Bit 4 : DIN5 Bit 5 : DIN6/AIN1 Bit 6 : DIN7/AIN2 Bit 7 : Sonde CTP Bit 8 : DIN1 IOE 1 Bit 9 : DIN2 IOE 1 Bit 10 : DIN3 IOE 1 Bit 11 : DIN4 IOE 1 Bit 12 : DIN1 IOE 2 Bit 13 : DIN2 IOE 2 Bit 14 : DIN3 IOE 2 Bit 15 : DIN4 IOE 2	INT	R	SK 2xxE-FDS
_7_Analog_input1	Valeur de l'entrée	10,00V = 1000	INT	R	tous

Nom	Fonction	Échelonnage	Type	Accès	Appareil
	analogique 1 (AIN1)				
_8_Analog_input2	Valeur de l'entrée analogique 2 (AIN2)	10,00V = 1000	INT	R	tous
_9_Analog_input3	Valeur de la fonction analogique DIN2	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_10_Analog_input4	Valeur de la fonction analogique DIN3	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_11_External_analog_input1	Valeur de l'entrée analogique 1 (IOE 1)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_12_External_analog_input2	Valeur de l'entrée analogique 2 (1.IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_13_External_analog_input3	Valeur de l'entrée analogique 1 (2.IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_14_External_analog_input4	Valeur de l'entrée analogique 2 (IOE 2)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_15_State_analog_output	État de l'entrée analogique	10,0V = 100	BYTE	R	SK 5xxP SK 54xE
_16_State_ext_analog_out1	État de la sortie analogique (IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_17_State_ext_analog_out2	État de la sortie analogique (IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
					SK 180E SK 190E
_18_Dip_switch_state	État des commutateurs DIP	Bit 0 : DIP1 Bit 1 : DIP2 Bit 2 : DIP3 Bit 3 : DIP4 Bit 4 : DIP_I1 Bit 5 : DIP_I2 Bit 6 : DIP_I3 Bit 7 : DIP_I4	INT	R	SK 155E-FDS SK 175E-FDS

3.5.2 Valeurs de consigne et réelles PLC

Les valeurs de processus indiquées ici forment l'interface du PLC vers l'appareil. La fonction des valeurs de consigne PLC est définie dans (P553).

Informations

La valeur de processus PLC_control_word écrase le bloc fonctionnel MC_Power. Les valeurs de consigne PLC écrasent les blocs fonctionnels MC_Move…. et MC_Home.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_20_PLC_control_ word	Mot de commande PLC	Correspond au profil USS	INT	R/W	tous
_21_PLC_set_val1	Consigne PLC 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_22_PLC_set_val2	Consigne PLC 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_23_PLC_set_val3	Consigne PLC 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_24_PLC_set_val4	Consigne PLC 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_25_PLC_set_val5	Consigne PLC 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_26_PLC_additional_	Mot de commande	Correspond au profil	INT	R/W	SK 5xxP

Nom	Fonction	Échelonnage	Type	Accès	Appareil
control_word1	supplémentaire PLC 1	USS			SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_27_PLC_additional_ control_word2	Mot de commande supplémentaire PLC 2	Correspond au profil USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_28_PLC_status_word	Mot d'état PLC	Correspond au profil USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_29_PLC_act_val1	Valeur de consigne PLC 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_30_PLC_act_val2	Valeur de consigne PLC 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_31_PLC_act_val3	Valeur de consigne PLC 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_32_PLC_act_val4	Valeur de consigne PLC 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
					SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_33_PLC_act_val5	Valeur de consigne PLC 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_34_PLC_Busmaster_ Control_word	Mot de commande de la fonction maître (fonction du maître bus) via PLC	Correspond au profil USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_35_PLC_32Bit_set_ val1	Consigne PLC 32Bits - P553[1] = Low Part de la valeur 32Bits - P553[2] = High Part de la valeur 32Bits	_	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_36_PLC_32Bit_act_ val1	Valeur réelle PLC 32Bits - Valeur réelle PLC 1 = Low Part de la valeur 32Bits - Valeur réelle PLC 2 = High Part de la valeur 32Bits	_	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_37_PLC_status_bits	Sorties d'état virtuelles de PLC	Bit 0 : PLC-DOUT1 Bit 1 : PLC-DOUT2 Bit 2 : PLC-DOUT3 (non défini pour le moment) Bit 3 : PLC-DOUT4 (non défini pour le moment)	INT	R/W	SK 155E-FDS SK 175E-FDS
_38_PLC_control_bits	Sorties de commande virtuelles de PLC	Bit 0 : PLC-DIN1 Bit 1 : PLC-DIN2 Bit 2 : PLC-DIN3 Bit 3 : PLC-DIN4 Bit 4 : PLC-DIN5 Bit 5 : PLC-DIN6 Bit 6 : PLC-DIN7 Bit 7 : PLC-DIN8	INT	R/W	SK 155E-FDS SK 175E-FDS

3.5.3 Valeurs de consigne et réelles de bus

Ces valeurs de processus sont le reflet de toutes les valeurs de consigne et réelles qui sont échangées par l'appareil par le bus.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_40_Inverter_status	Mot d'état du VF	Correspond au profil USS	INT	R	tous
_41_Inverter_act_val1	Valeur réelle 1 du VF	100% = 4000h	INT	R	tous
_42_Inverter_act_val2	Valeur réelle 2 du VF	100% = 4000h	INT	R	tous
_43_Inverter_act_val3	Valeur réelle 3 du VF	100% = 4000h	INT	R	tous
_44_Inverter_act_val4	Valeur réelle 4 du VF	100% = 4000h	INT	R	SK 5xxP SK 54xE
_45_Inverter_act_val5	Valeur réelle 5 du VF	100% = 4000h	INT	R	SK 5xxP SK 54xE
_46_Inverter_lead_val1	Émission fonction maître : Valeur maître 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_47_Inverter_lead_val2	Émission fonction maître : Valeur maître 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_48_Inverter_lead_val3	Émission fonction maître : Valeur maître 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_49_Inverter_lead_val4	Émission fonction maître : Valeur maître 4	100% = 4000h	INT	R	SK 5xxP SK 54xE
_50_Inverter_lead_val5	Émission fonction maître : Valeur maître 5	100% = 4000h	INT	R	SK 5xxP SK 54xE
_51_Inverter_control_word	Résultat du mot de commande bus	Correspond au profil USS	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
					SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_52_Inverter_set_val1	Résultat de la valeur de consigne principale 1 bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_53_Inverter_set_val2	Résultat de la valeur de consigne principale 2 bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_54_Inverter_set_val3	Résultat de la valeur de consigne principale 3 bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_55_Inverter_set_val4	Résultat de la valeur de consigne principale 4 bus	100% = 4000h	INT	R	SK 5xxP SK 54xE
_56_Inverter_set_val5	Résultat de la valeur de consigne principale 5 bus	100% = 4000h	INT	R	SK 5xxP SK 54xE
_57_Broadcast_set_val 1	Émission esclave : valeur de consigne secondaire 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_58_Broadcast_set_val 2	Émission esclave : valeur de consigne secondaire 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_59_Broadcast_set_val 3	Émission esclave : valeur de consigne secondaire 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_60_Broadcast_set_val 4	Émission esclave : valeur de consigne secondaire 4	100% = 4000h	INT	R	SK 5xxP SK 54xE
_61_Broadcast_set_val 5	Émission esclave : valeur de consigne secondaire 5	100% = 4000h	INT	R	SK 5xxP SK 54xE
_62_Inverter_32Bit_set _val1	Résultat de la valeur de consigne principale 32Bits 1 bus	- Low Part dans P546[1] - High Part dans P546[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_63_Inverter_32Bit_act _val1	Valeur réelle 1 du VF 32Bits	- Low Part dans P543[1] - High Part dans P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_64_Inverter_32Bit_lea d_val1	32Bits valeur maître 1	- Low Part dans P502[1] - High Part dans P502[2]	LONG	R	SK 5xxP SK 54xE SK 2xxE SK 180E SK 190E
_65_Broadcast_32Bit_s et_val1	32Bits émission esclave valeur de consigne secondaire 1	- Low Part dans P543[1] - High Part dans P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_66_BusIO_input_bits	Données de bus E/S entrée	- Bit0 – 7 = BusE/S entrée Bit 0 – 7 - Bit 8 = Drapeau 1 - Bit 9 = Drapeau 2 - Bit 10 = Mot commande bus bit8 - Bit 11 = Mot commande bus bit9	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_67_BusIO_output_bits	Données de bus E/S sortie	Bit0 = Bus / AS-i Sortie Digitale 1	INT	R	SK 5xxP SK 54xE

Nom	Fonction	Échelonnage	Type	Accès	Appareil
		Bit1 = Bus / AS-i Sortie Digitale 2 Bit2 = Bus / AS-i Sortie Digitale 3 Bit3 = Bus / AS-i Sortie Digitale 4 Bit4 = Bus /1.IOE Sortie Digitale 1 Bit5 = Bus /1.IOE Sortie Digitale 2 Bit6 = Bus / 2.IOE Sortie Digitale 1 Bit7 = Bus / 2.IOE Sortie Digitale 2 Bit8 = Mot état bus bit 10 Bit9 = Mot état bus bit 11			
_67_BusIO_output_bits	Données de bus E/S sortie	Bit0 = Bus / AS-i Sortie Digitale 1 Bit1 = Bus / AS-i Sortie Digitale 2 Bit2 = Bus / AS-i Sortie Digitale 3 Bit3 = Bus / AS-i Sortie Digitale 4 Bit4 = AS-i Actionneur 1 Bit5 = AS-i Actionneur 2 Bit6 = Drapeau 1 Bit7 = Drapeau 2 Bit8 = Mot état bus bit 10 Bit9 = Mot état bus bit 11	INT	R	SK 53xE SK 52xE
_67_BusIO_output_bits	Données de bus E/S sortie	Bit0 = Bus / AS-i Sortie Digitale 1 Bit1 = Bus / AS-i Sortie Digitale 2 Bit2 = Bus / AS-i Sortie Digitale 3 Bit3 = Bus / AS-i Sortie Digitale 4 Bit4 = Bus / IOE Sortie Digitale 1 Bit5 = Bus / IOE Sortie Digitale 2 Bit6 = Bus / 2.IOE Sortie Digitale 1 Bit7 = Bus / 2.IOE Sortie Digitale 2 Bit8 = Mot état bus bit 10 Bit9 = Mot état bus bit 11	INT	R	SK 2xxE
_67_BusIO_output_bits	Données de bus E/S sortie	Bit0 = Bus / AS-i Sortie Digitale 1 Bit1 = Bus / AS-i Sortie Digitale 2	INT	R	SK 2xxE-FDS

Nom	Fonction	Échelonnage	Type	Accès	Appareil
		Bit2 = Bus / AS-i Sortie Digitale 3 Bit3 = Bus / AS-i Sortie Digitale 4 Bit4 = Bus / AS-i Sortie Digitale 5 Bit5 = Bus / AS-i Sortie Digitale 6 Bit6 = Bus / 2.IOE Sortie Digitale 1 Bit7 = Bus / 2.IOE Sortie Digitale 2 Bit8 = Mot état bus bit 10 Bit9 = Mot état bus bit 11			

3.5.4 ControlBox et ParameterBox

Les valeurs de processus indiquées ici permettent d'accéder aux consoles de commande. Des applications d'interface homme-machine (IHM) simples sont ainsi possibles.

Informations

Afin d'afficher les "key_states" dans PLC, les ControlBox et ParameterBox doivent se trouver dans le mode d'affichage PLC. Sinon, seule une valeur "0" est représentée.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_70_Set_controlbox_ show_val	Valeur d'affichage pour la ControlBox	Valeur d'affichage = Bit 29 – Bit 0 Décimale = Bit 31 – Bit30	DINT	R/W	tous
_71_Controlbox_key_ state	État du clavier de la ControlBox	Bit 0 : ON Bit 1 : OFF Bit 2 : DIR Bit 3 : UP Bit 4 : DOWN Bit 5 : Enter	BYTE	R	tous
_72_Parameterbox_ key_state	État du clavier de la ParameterBox	Bit 0 : ON Bit 1 : OFF Bit 2 : DIR Bit 3 : UP Bit 4 : DOWN Bit 5 : Enter Bit 6 : Right Bit 7 : Left	BYTE	R	tous

3.5.5 Paramètres d'informations

Les valeurs réelles de l’appareil les plus importantes sont indiquées ici.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_80_Current_fault	Numéro de défaut actuel	Erreur 10.0 = 100	BYTE	R	tous
_81_Current_warning	Avertissement en cours	Avertissement 10.0 = 100	BYTE	R	tous
_82_Current_reason_FI_blocked	Cause actuelle pour l'état de blocage	Problème 10.0 = 100	BYTE	R	tous
_83_Input_voltage	Tension de secteur actuelle	100 V = 100	INT	R	tous
_84_Current_frequenz	Fréquence actuelle	10Hz = 100	INT	R	tous
_85_Current_set_point_frequency1	Fréquence de consigne actuelle provenant de la source de valeur de consigne	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_86_Current_set_point_frequency2	Fréquence de consigne actuelle du variateur	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_87_Current_set_point_frequency3	Fréquence de consigne actuelle en aval de la rampe	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_88_Current_Speed	Vitesse actuelle calculée	100rpm = 100	INT	R	tous
_89_Actual_current	Courant de sortie actuel	10.0A = 100	INT	R	tous
_90_Actual_torque_current	Intensité de couple actuelle	10.0A = 100	INT	R	tous
_91_Current_voltage	Tension actuelle	100V = 100	INT	R	tous

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_92_Dc_link_voltage	Tension actuelle du circuit intermédiaire	100V = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_93_Actual_field_ current	Courant magnétique réel	10.0A = 100	INT	R	tous
_94_Voltage_d	Composants de tension actuels Axe d	100V = 100	INT	R	tous
_95_Voltage_q	Composants de tension actuels Axe q	100V = 100	INT	R	tous
_96_Current_cos_phi	Cos Phi réel	0.80 = 80	BYTE	R	tous
_97_Torque	Couple actuel	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_98_Field	Champ actuel	100% = 100	BYTE	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_99_Apparent_power	Puissance apparente actuelle	1,00KW = 100	INT	R	tous
_100_Mechanical_ power	Puissance mécanique actuelle	1,00KW = 100	INT	R	tous
_101_Speed_encoder	Vitesse actuelle mesurée	100rpm = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE
_102_Usage_rate_ motor	Taux d'utilisation actuel du moteur (valeur momentanée)	100% = 100	INT	R	tous

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_103_Usage_rate_ motor_I2t	Taux d'utilisation actuel du moteur I2t	100% = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_104_Usage_rate_ brake_resistor	Taux d'utilisation actuel de la résistance de freinage	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_105_Head_sink_temp	Température actuelle du radiateur	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_106_Inside_temp	Température actuelle de la pièce	100°C = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_107_Motor_temp	Température moteur actuelle	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_108_Actual_net_ frequency	Fréquence réseau actuelle	10Hz = 100	INT	R	SK 155E-FDS SK 175E-FDS
_109_Mains_phase_ sequence	Séquence ordre de phase actuelle	0=CW, 1=CCW	BYTE	R	SK 155E-FDS SK 175E-FDS
_141_Pos_Sensor_Inc	Position du codeur incrémental	0.001 tour	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_142_Pos_Sensor_	Position du codeur	0.001 tour	DINT	R	SK 5xxP

Nom	Fonction	Échelonnage	Type	Accès	Appareil
Abs	absolu				SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_143_Pos_Sensor_Uni	Position du codeur universel	0.001 tour	DINT	R	SK 5xxP SK 54xE
_144_Pos_Sensor_ HTL	Position du codeur HTL	0.001 tour	DINT	R	SK 5xxP SK 54xE
_145_Actual_pos	Position réelle	0.001 tour	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_146_Actual_ref_pos	Position de réglage actuelle	0.001 tour	DINT	R	S SK 5xxP K 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_147_Actual_pos_diff	Différence de position entre la valeur de consigne et la valeur réelle	0.001 tour	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E

3.5.6 Erreur PLC

Par le biais des User Error Flags, les erreurs de l'appareil E23.0 à E24.7 peuvent être définies à partir du programme PLC.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_110_ErrorFlags	Génère des erreurs utilisateur dans l'appareil	**Bit 0: E 23.0** **Bit 1: E 23.1** **Bit 2: E 23.2** **Bit 3: E 23.3** **Bit 4: E 23.4** **Bit 5: E 23.5** **Bit 6: E 23.6** **Bit 7: E 23.7**	BYTE	R/W	tous
_111_ErrorFlags_ext	Génère des erreurs utilisateur dans l'appareil	**Bit 0: E 24.0** **Bit 1: E 24.1** **Bit 2: E 24.2** **Bit 3: E 24.3** **Bit 4: E 24.4** **Bit 5: E 24.5** **Bit 6: E 24.6** **Bit 7: E 24.7**	BYTE	R/W	tous

3.5.7 Paramètres PLC

Ces groupes de données de processus permettent d'accéder directement aux paramètres PLC P355, P356 et P360.

Nom	Fonction	Échelonnage	Type	Accès	Appareil
_115_PLC_P355_1	Paramètre PLC entier P355 [-01]	-	INT	R	tous
_116_PLC_P355_2	Paramètre PLC entier P355 [-02]	-	INT	R	tous
_117_PLC_P355_3	Paramètre PLC entier P355 [-03]	-	INT	R	tous
_118_PLC_P355_4	Paramètre PLC entier P355 [-04]	-	INT	R	tous
_119_PLC_P355_5	Paramètre PLC entier P355 [-05]	-	INT	R	tous
_120_PLC_P355_6	Paramètre PLC entier P355 [-06]	-	INT	R	tous
_121_PLC_P355_7	Paramètre PLC entier P355 [-07]	-	INT	R	tous
_122_PLC_P355_8	Paramètre PLC entier P355 [-08]	-	INT	R	tous
_123_PLC_P355_9	Paramètre PLC entier P355 [-09]	-	INT	R	tous
_124_PLC_P355_10	Paramètre PLC entier P355 [-10]	-	INT	R	tous
_125_PLC_P356_1	Paramètre PLC LONG P356 [-01]	-	DINT	R	tous
_126_PLC_P356_2	Paramètre PLC LONG P356 [-02]	-	DINT	R	tous
_127_PLC_P356_3	Paramètre PLC LONG P356 [-03]	-	DINT	R	tous
_128_PLC_P356_4	Paramètre PLC LONG P356 [-04]	-	DINT	R	tous
_129_PLC_P356_5	Paramètre PLC LONG P356 [-05]	-	DINT	R	tous
_130_PLC_P360_1	Paramètre affichage PLC P360[-01]	-	DINT	R/W	tous
_131_PLC_P360_2	Paramètre affichage PLC P360[-02]	-	DINT	R/W	tous
_132_PLC_P360_3	Paramètre affichage PLC P360[-03]	-	DINT	R/W	tous
_133_PLC_P360_4	Paramètre affichage PLC P360[-04]	-	DINT	R/W	tous
_134_PLC_P360_5	Paramètre affichage PLC	-	DINT	R/W	tous

Nom	Fonction	Échelonnage	Type	Accès	Appareil
	P360[-05]				
_135_PLC_Scope_ Int_1	PLC Scope valeur d'affichage 1	-	INT	R/W	tous
_136_PLC_Scope_ Int_2	PLC Scope valeur d'affichage 2	-	INT	R/W	tous
_137_PLC_Scope_ Int_3	PLC Scope valeur d'affichage 3	-	INT	R/W	tous
_138_PLC_Scope_ Int_4	PLC Scope valeur d'affichage 4	-	INT	R/W	tous
_139_PLC_Scope_ Bool_1	PLC Scope valeur d'affichage 5	-	INT	R/W	tous
_140_PLC_Scope_ Bool_2	PLC Scope valeur d'affichage 6	-	INT	R/W	tous

3.6 Langues

3.6.1 Liste d’instructions (AWL / IL)

3.6.1.1 Généralités

Types de données

PLC prend en charge les types de données suivants.

Nom	Espace disque requis	Plage de valeurs
BOOL	1 bit	0 à 1
BYTE	1 octet	0 à 255
INT	2 octets	-32768 à 32767
DINT	4 octets	-2 147 483 648 à 2 147 483 647
LABEL_ADDRESS	2 octets	Étiquette

Littéraux

Pour plus de clarté, il est possible de saisir des constantes de tous les types de données dans différentes formes de représentation. Le tableau suivant présente une vue d'ensemble de toutes les variantes possibles.

Littéral	Exemple	Nombre sous forme décimale
Bool	**FALSE**	0
	TRUE	1
	BOOL#0	0
	BOOL#1	1
Binaire (base 2)	**2#01011111**	95
	2#0011_0011	51
	BYTE#2#00001111	15
	BYTE#2#0001_1111	31
Octal (base 8)	**8#0571**	377
	8#05_71	377
	BYTE#8#10	8
	BYTE#8#111	73
	BYTE#8#1_11	73
Hexadécimal (base 16)	**16#FFFF**	-1
	16#0001_FFFF	131071
	INT#16#1000	4096
	DINT#16#0010_2030	1056816
Décimal (base 10)	**10**	10
	-10	-10
	10_000	10000
	INT#12	12
	DINT#-100000	-100000
Temps	**TIME#10s50ms**	10,050 secondes
	T#5s500ms	5,5 secondes
	TIME#5.2s	5,2 secondes
	TIME#5D10H15M	5 jours+10 heures+15 minutes
	T#1D2H30M20S	1 jour +2 heures+30 minutes+20 secondes

Commentaires

Il est recommandé d'ajouter des explications aux sections du programme afin que le programme PLC soit compréhensible ultérieurement. Ces commentaires sont mis en évidence dans le programme utilisateur en commençant par la chaîne de caractères "(*" et se terminant par "*)" comme dans les exemples suivants.

```
(* Commentaire à propos d'un bloc du programme *)
LD 100 (* Commentaire après une commande *)
ADD 20
```

Étiquette

À l'aide des opérateurs JMP, JMPC ou JMPCN, des parties entières du programme peuvent être ignorées. Une étiquette est indiquée en tant qu'adresse cible. À l'exception des caractères comportant un tréma (par ex. ö, ä ou ü) et du caractère "ß", elle peut comporter toutes les lettres, les chiffres de 0 à 9 et les traits de soulignement. Les autres caractères ne sont pas autorisés. L'étiquette se termine par deux-points. Elle peut figurer seule ou se trouver sur la même ligne qu'une autre commande, après l'étiquette.

Les variantes possibles peuvent être semblables à celles-ci :

Exemple :

```
Étiquette :
LD 20

Voici_une_étiquette :
ADD 10

MainLoop: LD 1000
```

Une autre variante est le transfert d'une étiquette en tant que variable. Cette variable doit être définie en tant que type LABEL_ADDRESS dans le tableau des variables. Des étiquettes peuvent ensuite être chargées dans cette variable. Ainsi, des machines à états finis peuvent être facilement créées ; voir ci-après

Exemple :

```
LD FirstTime
JMPC AfterFirstTime
(* L’adresse de l’étiquette doit être initialisée au début. *)
LD Address_1
ST Address_Var
LD TRUE
ST FirstTime
AfterFirstTime:
JMP Address_Var
Address_1:
LD Address_2
ST Address_Var
JMP Ende
Address_2:
LD Address_3
ST Address_Var
JMP Ende
Address_3:
LD Address_1
ST Address_Var
End:
```

Appels de fonctions

L'éditeur prend en charge une forme d'appels de fonctions. Dans les variantes suivantes, la fonction CTD est appelée via l'instance I_CTD. Les résultats sont enregistrés dans les variables. La signification des fonctions utilisées ci-après est expliquée plus loin dans le manuel.

Exemple :

```
LD 10000
ST I_CTD.PV
LD LoadNewVar
ST I_CTD.LD
LD TRUE
ST I_CTD.CD
CAL I_CTD
LD I_CTD.Q
ST ResultVar
LD I_CTD.CV
ST CurrentCountVar
```

Accès par bits aux variables

Pour l'accès à un bit à partir d'une variable ou d'une variable de processus, une forme simplifiée est possible.

Commande	Signification
LD Var1.0	Charge le bit 0 de Var1 dans AE
ST Var1.7	Enregistre l'AE sur le bit 7 de Var1
EQ Var1.4	Compare l'AE avec le bit 4 de Var1

3.6.2 Littéral structuré (ST)

Le littéral structuré est constitué d'une série d'instructions qui peuvent être exécutées - tout comme dans les langages évolués - en condition ("IF..THEN..ELSE") ou en boucle d'itération (WHILE..DO).

Exemple :

```
IF value < 7 THEN
  WHILE value < 8 DO
    value := value + 1;
  END_WHILE;
END_IF;
```

3.6.2.1 Généralités

Types de données dans ST

PLC prend en charge les types de données suivants.

Nom	Espace disque requis	Plage de valeurs
BOOL	1 bit	0 à 1
BYTE	1 octet	0 à 255
INT	2 octets	-32768 à 32767
DINT	4 octets	-2 147 483 648 à 2 147 483 647

Informations

Pour les nombres, il est judicieux d'indiquer le type de données pour générer un programme PLC efficace, par ex. : VarInt := INT#-32768, VarDINT := DINT#-2147483648.

Opérateur d'affectation

À gauche d'une affectation se trouve un opérande (variable, adresse) auquel la valeur de l'expression est affectée sur la droite avec l'opérateur d'affectation ":=".

Exemple :

```
Var1 := Var2 * 10;
```

Après l'exécution de cette ligne, Var1 a la valeur de Var2 multipliée par 10.

Appel des blocs fonctionnels dans ST

Un bloc fonctionnel est appelé dans ST en écrivant le nom de l'instance du bloc fonctionnel et en affectant ensuite entre parenthèses les valeurs souhaitées aux paramètres. Dans l'exemple suivant, un temporisateur est appelé avec des affectations pour ses paramètres IN et PT. Ensuite, la variable de résultat Q est affectée à la variable A.

L'accès à la variable de résultat est effectué comme dans AWL avec le nom du bloc fonctionnel, un point qui suit et le nom des variables.

Exemple :

```
Timer(IN := TRUE, PT := 300);
A := Timer.Q;
```

Évaluation des expressions

L'évaluation d'une expression est effectuée par le traitement des opérateurs selon des règles de liaison. L'opérateur avec la plus forte liaison est tout d'abord traité, puis l'opérateur avec la plus forte liaison suivante, etc. jusqu'à ce que tous les opérateurs soient traités. Les opérateurs avec la même force de liaison sont traités de gauche à droite.

Ci-après, un tableau des opérateurs ST indique l'ordre de leur force de liaison :

Opération	Symbole	Force de liaison
Entre parenthèses	(Expression)	Liaison la plus forte
Appel de fonction	Nom de la fonction (liste des paramètres)	
Négation, formation complémentaire	**NOT**	
Multiplication Division Modulo AND	***** **/** **MOD** **AND**	
Addition Soustraction OR XOR	**+** **-** **OR** **XOR**	
Comparaison Égalité Inégalité	**<,>,<=,>=** **=** **<>**	Liaison la plus faible

3.6.2.2 Instructions

Return

L'instruction RETURN peut être utilisée pour passer à la fin du programme, en fonction d'une condition.

IF

Avec l'instruction IF, une condition peut être vérifiée et en fonction de cette condition, des instructions sont exécutées.

Syntaxe :

```
IF <Expression_booléenne1> THEN
  <Instructions_IF>
ELSIF <Expression_booléenne2> THEN
  <Instructions_ELSIF1>
ELSIF <Expression_booléenne n> THEN
  <Instructions_ELSIF n-1>
ELSE
  <Instructions_ELSE>}
END_IF;
```

La partie entre accolades {} est facultative.

Si <Expression_booléenne1> a pour résultat TRUE, seules les <Instructions_IF> sont exécutées mais aucune autre instruction. Si ce n'est pas le cas, les expressions booléennes commençant par <Expression_booléenne2> sont évaluées dans l'ordre jusqu'à ce que l'une des expressions aboutisse à TRUE. Ensuite, seules les instructions après cette expression booléenne et avant l'instruction ELSE ou ELSIF suivante sont évaluées. Si aucune des expressions booléennes n'a TRUE comme résultat, seules les <Instructions_ELSE> sont alors évaluées.

Exemple :

```
IF temp < 17 THEN
  Bool1 := TRUE;
ELSE
  Bool2 := FALSE;
END_IF;
```

CASE

Avec l'instruction CASE, plusieurs instructions conditionnelles sont combinées en une structure avec la même variable de condition.

Syntaxe :

```
CASE <Var1> OF
  <Valeur 1> : <Instruction 1>
  <Valeur 2> : <Instruction 2>
  <Valeur3, Valeur4, Valeur5 : <Instruction 3>
  <Valeur6 .. Valeur10 : <Instruction 4>
  ...
  <Valeur n>: <Instruction n>
ELSE <Instruction ELSE>
END_CASE;
```

Une instruction CASE est traitée selon le schéma suivant :

- Si la variable dans <Var1> a la valeur <Valeur i>, l'instruction <Instruction i> est exécutée
- Si <Var 1> n'a aucune des valeurs indiquées, <Instruction-ELSE> est exécutée.
- Si pour plusieurs valeurs des variables, la même instruction doit être exécutée, il est possible d'écrire successivement ces valeurs séparées par des virgules, en tant que condition pour l'instruction commune.
- Si pour une plage de valeurs des variables, la même instruction doit être exécutée, il est possible d'écrire successivement ces valeurs de début et de fin séparées par deux points, en tant que condition pour l'instruction commune.

Exemple :

```
CASE INT1 OF
  1, 5:
    BOOL1 := TRUE;
    BOOL3 := FALSE;
  2:
    BOOL2 := FALSE;
    BOOL3 := TRUE;
  10..20:
    BOOL1 := TRUE;
    BOOL3:= TRUE;
  ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Boucle FOR

Avec la boucle FOR, il est possible de programmer des processus répétitifs.

Syntaxe :

```
FOR <INT_Var> := <VALEUR_INIT> TO <VALEUR_FIN> {BY <Incrément>} DO
  <Instructions>
END_FOR;
```

La partie entre accolades {} est facultative. Les <Instructions> sont exécutées tant que le compteur <INT_Var> n'est pas supérieur à la <VALEUR_END>. Ceci est vérifié avant l'exécution des <Instructions> de sorte que les <Instructions> ne soient jamais exécutées si <VALEUR_INIT> est supérieure à <VALEUR_END>. En cas d'exécution de <Instructions>, <INT_Var> est toujours augmentée de <Incrément>. L'incrément peut avoir chaque nombre entier. S'il manque, il est défini sur 1. L'exécution de la boucle doit être terminée car <INT_Var> devient seulement plus grande.

Exemple :

```
Compteur FOR :=1 TO 5 BY 1 DO
  Var1 := Var1 * 2;
  END_FOR;
```

Boucle REPEAT

La boucle REPEAT se différencie des boucles WHILE par le fait que la condition de terminaison n'est vérifiée qu'après l'exécution de la boucle. Cela a pour conséquence que la boucle sera exécutée au moins une fois quelle que soit la condition de terminaison.

Syntaxe :

```
REPEAT
  <Instructions>
UNTIL <Expression booléenne>
END_REPEAT;
```

Les <Instructions> sont exécutées jusqu'à ce que l'<Expression booléenne> soit TRUE. Si l'<Expression booléenne> est déjà TRUE lors de la première évaluation, les <Instructions> sont exécutées précisément une fois. Si l'<Expression booléenne> n'a jamais la valeur TRUE, toutes les <Instructions> sont répétées sans fin. Une erreur d'exécution est créée à cet effet.

Informations

Le programmeur doit veiller lui-même à ce qu'il n'y ait pas de boucle sans fin en modifiant la condition dans la partie des instructions de la boucle, par exemple un compteur effectuant un comptage ascendant ou descendant.

Exemple :

```
REPEAT
  Var1 := Var1 * 2;
  Compteur := Compteur - 1;
UNTIL
  Compteur = 0
END_REPEAT
```

Boucle WHILE

La boucle WHILE peut être utilisée comme la boucle FOR à la différence que la condition de terminaison peut être une expression booléenne quelconque. Cela signifie qu'une condition est indiquée, qui si elle s'applique, a pour conséquence l'exécution de la boucle.

Syntaxe :

```
WHILE <Expression booléenne> DO
  <Instructions>
END_WHILE;
```

Les <Instructions> sont exécutées jusqu'à ce que l'<Expression booléenne> soit FALSE. Si l'<Expression booléenne> est déjà FALSE pendant la première évaluation, les <Instructions> sont exécutées précisément une fois. Si l'<Expression booléenne> n'a jamais la valeur FALSE, toutes les <Instructions> sont répétées sans fin. Une erreur d'exécution est créée à cet effet.

Informations

Le programmeur doit veiller lui-même à ce qu'il n'y ait pas de boucle sans fin en modifiant la condition dans la partie des instructions de la boucle, par exemple un compteur effectuant un comptage ascendant ou descendant.

Exemple :

```
Compteur WHILE >0 DO
  Var1 := Var1 * 2;
  Compteur := Compteur - 1;
END_WHILE
```

Exit

Si l'instruction EXIT survient dans une boucle FOR, WHILE ou REPEAT, la boucle la plus interne est terminée, indépendamment de la condition de terminaison.

3.7 Sauts

3.7.1 JMP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Saut inconditionnel vers une étiquette.

Exemple dans AWL :

```
JMP NextStep (* Saut inconditionnel vers NextStep *)
ADD 1

NextStep:
ST Value1
```

3.7.2 JMPC

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Saut conditionnel (Jump Conditional) vers une étiquette. Si AE = TRUE, l'instruction JMPC saute jusqu'à l'étiquette indiquée.

Exemple dans AWL :

```
LD 10
JMPC NextStep (* AE = TRUE à Saut du programme *)
ADD 1

NextStep:
ST Value1
```

3.7.3 JMPCN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

Saut conditionnel (Jump Conditional) vers une étiquette. JMPCN saute si AE Register = FALSE. Sinon le programme se poursuit avec l'instruction suivante.

Exemple dans AWL :

```
LD 10
JMPCN NextStep (* AE = TRUE à Pas de saut du programme *)
ADD 1

NextStep:
ST Value1
```

3.8 Conversion de type

3.8.1 BOOL_TO_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données	X			

Convertit le type de données AE de BOOL vers BYTE. Si l'AE est égal à FALSE, l'accumulateur est converti à 0. Si l'AE est égal à TRUE, l'accumulateur est converti à 1.

Exemple dans AWL :

```
LD TRUE
BOOL_TO_BYTE (* AE = 1 *)
```

Exemple dans ST :

```
Résultat := BOOL_TO_BYTE(TRUE); (* Résultat = 1 *)
```

3.8.2 BYTE_TO_BOOL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X		

Convertit le type de données de BYTE vers BOOL. Tant que BYTE est différent de zéro, un résultat TRUE est toujours disponible pour la conversion.

Exemple dans AWL :

```
LD 10
BYTE_TO_BOOL (* AE = TRUE *)
```

Exemple dans ST :

```
Résultat := BYTE_TO_BOOL(10); (* Résultat = TRUE *)
```

3.8.3 BYTE_TO_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données		X		

Convertit le type de données de BYTE vers INT. BYTE est copié dans le composant Low de INT et le composant High de INT est mis à 0.

Exemple dans AWL :

```
LD 10
BYTE_TO_INT (* Accumulateur = 10 *)
```

Exemple dans ST :

```
Résultat := BYTE_TO_INT(10); (* Résultat = 10 *)
```

3.8.4 DINT_TO_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données				X

Convertit le type de données de DINT vers INT. Le composant High de la valeur DINT n'est pas transféré.

Exemple dans AWL :

```
LD 200000
DINT_TO_INT     (* Accumulateur = 3392 *)

LD DINT# -5000
DINT_TO_INT     (* Accumulateur = -5000 *)

LD DINT# -50010
DINT_TO_INT     (* Accumulateur = 15526 *)
```

Exemple dans ST :

```
Résultat := DINT_TO_INT(200000); (* Résultat = 3392 *)
Résultat := DINT_TO_INT(-5000); (* Résultat = -5000 *)
Résultat := DINT_TO_INT(-50010); (* Résultat = 15526 *)
```

3.8.5 INT_TO_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données			X	

Convertit le type de données de INT vers BYTE. Le composant High de la valeur INT n'est pas transféré. Les signes sont perdus car le type BYTE est sans signe.

Exemple dans AWL :

```
LD 16#5008
INT_TO_BYTE (* Accumulateur = 8 *)
```

Exemple dans ST :

```
Résultat := INT_TO_BYTE(16#5008); (* Résultat = 8 *)
```

3.8.6 INT_TO_DINT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Disponibilité	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Type de données			X	

Convertit le type de données de INT vers DINT. INT est copié dans le composant Low de DINT et le composant High de DINT est mis à 0.

Exemple dans AWL :

```
LD 10
INT_TO_DINT (* Accumulateur = 10 *)
```

Exemple dans ST :

```
Résultat := INT_TO_DINT(10); (* Résultat = 10 *)
```

3.9 Messages de dysfonctionnement PLC

Les messages de dysfonctionnement provoquent l'arrêt de l'appareil afin d'éviter tout endommagement de l'appareil. Dans le cas des messages de dysfonctionnement du PLC, le traitement du PLC est arrêté et le PLC passe dans l'état "PLC Error". Dans le cas des messages de dysfonctionnement, le PLC continue à fonctionner. Après l'acquittement de l'erreur, le PLC redémarre automatiquement.

Dans le cas de l'erreur PLC défaut client 23.X, le PLC continue à fonctionner !

Affichage dans la SimpleBox		**Défaut** **Texte dans la ParameterBox**	**Cause** **Remède**
Groupe	**Détails dans P700[-01] / P701**		
E022	**22.0**	**Pas de programme PLC**	Le PLC a démarré. Il n'y a cependant pas de programme PLC dans le VF. - Charger le programme PLC dans l'appareil
	22.1	**Programme PLC défectueux**	La vérification des sommes de contrôle via le programme PLC a détecté une erreur. - Redémarrer l'appareil (Power ON) et réessayer - Ou bien, charger de nouveau le programme PLC
	22.2	**Adresse de saut incorrecte**	Erreur du programme, comportement semblable à celui de l'erreur 22.1
	22.3	**Dépassement pile**	Plus de 6 niveaux de parenthèses ont été ouverts pendant l'exécution du programme - Vérifier si le programme comporte des erreurs d'exécution
	22.4	**Cycles PLC max. dépassés**	Le temps de cycle max. indiqué du programme PLC a été dépassé - Adapter le temps de cycle ou vérifier le programme
	22.5	**Code de commande inconnu**	Un code de commande disponible dans le programme ne peut pas être exécuté car il est inconnu - Erreur de programme, comportement semblable à celui de l'erreur 22.1 - La version du PLC et la version de NORD CON ne coïncident pas
	22.6	**Accès écriture PLC**	Pendant l'exécution d'un programme PLC, le contenu du programme a été modifié
	22.9	**Erreur de regroupement PLC**	La cause de l'erreur ne peut pas être précisément déterminée - Comportement semblable à celui de l'erreur 22.1
E023	**23.0**	**PLC défaut client 1**	Cette erreur peut être résolue par le programme PLC afin de signaler des problèmes d'exécution du programme PLC de façon externe. Le déclenchement est effectué par la description de la variable de processus "ErrorFlags".
	23.1	**PLC défaut client 2**	
	23.2	**PLC défaut client 3**	

4 Paramètres

Les paramètres d’appareil pour les fonctions PLC sont décrits de façon détaillée dans le manuel du variateur de fréquence ou du démarreur correspondant.

5 Annexe

5.1 Instructions d'entretien et de mise en service

En cas de problèmes, par ex. pendant la mise en service, prenez contact avec notre service après-vente.

☎ +49 4532 289-2125

Notre service est disponible 24h sur 24, 7 jours sur 7 et peut vous aider à trouver les informations suivantes sur l'appareil et ses accessoires:

- désignation du type,
- numéro de série,
- version du microprogramme.

5.2 Documents et logiciels

Les documents et logiciels peuvent être téléchargés à partir de notre site Internet www.nord.com .

Documents complémentaires

Documentation	Table des matières
BU 0155	Manuel pour démarreurs du module de répartition NORDAC *LINK* **SK 180E / SK 190E**
BU 0180	Manuel pour variateurs de fréquence NORDAC *BASE* **SK 180E / SK 190E**
BU 0200	Manuel pour variateurs de fréquence NORDAC *FLEX* **SK 200E .. SK 235E**
BU 0250	Manuel pour variateurs de fréquence NORDAC *LINK* **SK 250E-FDS .. SK 280E-FDS**
BU 0500	Manuel pour variateurs de fréquence NORDAC *PRO* **SK 500E .. SK 535E**
BU 0505	Manuel pour variateurs de fréquence NORDAC *PRO* **SK 540E .. SK 545E**
BU 0600	Manuel pour variateurs de fréquence NORDAC *PRO* **SK 500P .. SK 550P**
BU 0000	Manuel pour l'utilisation du logiciel NORDCON
BU 0040	Manuel pour l'utilisation des consoles de paramétrage NORD

Logiciel

Logiciel	Description
NORDCON	Logiciel de paramétrage et de diagnostic

5.3 Abréviations

- **AE** Résultat actuel
- **AIN** Entrée analogique
- **AOUT** Sortie analogique
- **AWL** Liste d'instructions (voir également IL)
- **COB-ID** Identifiant d'objet de communication
- **DI / DIN** Entrée digitale
- **DO / DOUT** Sortie digitale
- **E/S ou I/O** Entrée / Sortie
- **EEPROM** Mémoire non volatile
- **CEM** Compatibilité électromagnétique
- **FB** Bloc fonctionnel
- **VF** Variateur de fréquence
- **HSW** Valeur de consigne principale
- **IL** Liste d'instructions (voir également AWL)
- **ISD** Courant de champ (réglage du vecteur de courant)
- **DEL** Diode électroluminescente
- **MC** Motion Control
- **NSW** Valeur de consigne secondaire
- **P** Paramètre dépendant du jeu de paramètres, autrement dit, un paramètre auquel dans chacun des 4 jeux de paramètres de l'appareil différentes fonctions ou valeurs peuvent être affectées.
- **P-BOX** ParameterBox
- **PDO** Process Data Object (objet de données de processus)
- **PLC** Programmable Logic Controller (Automate Programmable Industriel, API)
- **S** Paramètre superviseur, autrement dit, un paramètre qui est uniquement visible lorsque le code superviseur correct est saisi dans le paramètre **P003**
- **SW** Version logiciel (voir le paramètre **P707**)
- **STW** Mot de commande
- **ZSW** Mot d'état (Status word)

Index

U

NORD DRIVESYSTEMS Group

Headquarters and Technology Centre
in Bargteheide, close to Hamburg

Innovative drive solutions
for more than 100 branches of industry

Mechanical products
parallel shaft, helical gear, bevel gear and worm gear units

Electrical products
IE2/IE3/IE4 motors

Electronic products
centralised and decentralised frequency inverters,
motor starters and field distribution systems

7 state-of-the-art production plants
for all drive components

Subsidiaries and sales partners
in 98 countries on 5 continents
provide local stocks, assembly, production,
technical support and customer service

More than 4,000 employees throughout the world
create customer oriented solutions

www.nord.com/locator

Headquarters:

Getriebebau NORD GmbH & Co. KG
Getriebebau-Nord-Straße 1
22941 Bargteheide, Germany
T: +49 (0) 4532 / 289-0
F: +49 (0) 4532 / 289-22 53
info@nord.com, www.nord.com

Member of the NORD DRIVESYSTEMS Group

NORD® DRIVESYSTEMS

6075504 / 1919