

BU 0550 – de

## PLC Funktionalität

Zusatanleitung für NORDAC - Geräte







## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b> .....	<b>7</b>
1.1	Allgemeines .....	7
1.1.1	Dokumentation .....	7
1.1.2	Dokumenthistorie.....	7
1.1.3	Urheberrechtsvermerk .....	8
1.1.4	Herausgeber.....	8
1.1.5	Zu diesem Handbuch .....	8
1.2	Mitgeltende Dokumente .....	9
1.3	Darstellungskonventionen.....	9
1.3.1	Warnhinweise .....	9
1.3.2	Andere Hinweise .....	9
1.4	Bestimmungsgemäße Verwendung .....	9
<b>2</b>	<b>Sicherheit</b> .....	<b>10</b>
2.1	Auswahl und Qualifikation des Personals .....	10
2.1.1	Qualifiziertes Personal.....	10
2.1.2	Elektrofachkraft.....	10
2.2	Sicherheitshinweise .....	10
<b>3</b>	<b>PLC</b> .....	<b>11</b>
3.1	Allgemeines .....	11
3.1.1	Spezifikation der PLC .....	12
3.1.2	PLC Aufbau .....	13
3.1.2.1	Speicher .....	13
3.1.2.2	Prozessabbild .....	13
3.1.2.3	Programm Task .....	14
3.1.2.4	Sollwert Verarbeitung .....	14
3.1.2.5	Datenverarbeitung über Akku .....	14
3.1.3	Funktionsumfang .....	15
3.1.3.1	Motion Control Lib .....	15
3.1.3.2	Elektronisches Getriebe mit Fliegender Säge .....	15
3.1.3.3	Visualisierung .....	15
3.1.3.4	Prozessregler .....	16
3.1.3.5	CANopen Kommunikation .....	16
3.2	Erstellen von PLC Programmen.....	17
3.2.1	Laden, Speichern & Drucken.....	17
3.2.2	Editor .....	18
3.2.2.1	Variablen und FB Deklaration .....	19
3.2.2.2	Eingabefenster .....	20
3.2.2.3	Watch- & Breakpoint Anzeigefenster .....	21
3.2.2.4	PLC Meldungsfenster .....	21
3.2.3	Programm zum Gerät übertragen.....	22
3.2.4	Debugging .....	23
3.2.4.1	Beobachtungspunkte (Watchpoints) .....	23
3.2.4.2	Haltepunkte (Breakpoints) .....	23
3.2.4.3	Einzelschritt (Single Step) .....	24
3.2.5	PLC Konfiguration .....	25
3.3	Funktionsblöcke .....	26
3.3.1	CANopen .....	26
3.3.1.1	Überblick .....	26
3.3.1.2	FB_NMT .....	27
3.3.1.3	FB_PDConfig .....	28
3.3.1.4	FB_PDORceive .....	31
3.3.1.5	FB_PDOSend .....	33
3.3.2	Elektronisches Getriebe mit Fliegender Säge.....	35
3.3.2.1	Überblick .....	36
3.3.2.2	FB_FlyingSaw .....	36
3.3.2.3	FB_Gearing .....	38
3.3.3	Motion Control .....	39
3.3.3.1	MC_Control .....	41
3.3.3.2	MC_Control_MS .....	43
3.3.3.3	MC_Home .....	44

3.3.3.4	MC_Home (SK 5xxP)	45
3.3.3.5	MC_MoveAbsolute	47
3.3.3.6	MC_MoveAdditive	49
3.3.3.7	MC_MoveRelative	50
3.3.3.8	MC_MoveVelocity	51
3.3.3.9	MC_Power	53
3.3.3.10	MC_ReadActualPos	55
3.3.3.11	MC_ReadParameter	56
3.3.3.12	MC_ReadStatus	57
3.3.3.13	MC_Reset	58
3.3.3.14	MC_Stop	59
3.3.3.15	MC_WriteParameter_16 / MC_WriteParameter_32	60
3.3.4	Standard .....	61
3.3.4.1	CTD Abwärtszähler	61
3.3.4.2	CTU Aufwärtszähler	62
3.3.4.3	CTUD Auf- und Abwärtszähler	63
3.3.4.4	R_TRIG und F_TRIG	65
3.3.4.5	R <sub>S</sub> Flip Flop	66
3.3.4.6	SR Flip Flop	67
3.3.4.7	TOF Ausschaltverzögerung	68
3.3.4.8	TON Einschaltverzögerung	69
3.3.4.9	TP Zeitimpuls	70
3.3.5	Zugriff auf Speicherbereiche des Frequenzumrichters .....	71
3.3.5.1	FB_ReadTrace	71
3.3.5.2	FB_WriteTrace	73
3.3.6	Visualisierung ParameterBox .....	75
3.3.6.1	Überblick Visualisierung	75
3.3.6.2	FB_DINTToPBOX	76
3.3.6.3	FB_STRINGToPBOX	79
3.3.7	FB_Capture (Erfassen schneller Ereignisse).....	81
3.3.8	FB_DinCounter.....	84
3.3.9	FB_FunctionCurve.....	86
3.3.10	FB_PIDT1.....	87
3.3.11	FB_ResetPosition.....	89
3.3.12	FB_Weigh.....	90
3.4	Operatoren.....	92
3.4.1	Arithmetische Operatoren.....	92
3.4.1.1	ABS	92
3.4.1.2	ADD und ADD(	93
3.4.1.3	DIV und DIV(	94
3.4.1.4	LIMIT	94
3.4.1.5	MAX	95
3.4.1.6	MIN	95
3.4.1.7	MOD und MOD(	96
3.4.1.8	MUL und MUL(	96
3.4.1.9	MUX	97
3.4.1.10	SUB und SUB(	97
3.4.2	Erweiterte mathematische Operatoren .....	98
3.4.2.1	COS, ACOS, SIN, ASIN, TAN, ATAN	98
3.4.2.2	EXP	99
3.4.2.3	LN	99
3.4.2.4	LOG	100
3.4.2.5	SQRT	100
3.4.3	Bit Operatoren .....	101
3.4.3.1	AND und AND(	101
3.4.3.2	ANDN und ANDN(	102
3.4.3.3	NOT	103
3.4.3.4	OR und OR(	104
3.4.3.5	ORN undORN(	105
3.4.3.6	ROL	106
3.4.3.7	ROR	106
3.4.3.8	S und R	107
3.4.3.9	SHL	107
3.4.3.10	SHR	108
3.4.3.11	XOR und XOR(	109
3.4.3.12	XORN und XORN(	110
3.4.4	Lade- und Speicheroperatoren.....	111
3.4.4.1	LD	111

3.4.4.2	LDN	111
3.4.4.3	ST	112
3.4.4.4	STN	112
3.4.5	Vergleichs Operatoren.....	113
3.4.5.1	EQ	113
3.4.5.2	GE	113
3.4.5.3	GT	114
3.4.5.4	LE	114
3.4.5.5	LT	115
3.4.5.6	NE	115
3.5	Prozesswerte .....	116
3.5.1	Ein- und Ausgänge .....	116
3.5.2	PLC Soll- und Istwerte.....	124
3.5.3	Bus Soll- und Istwerte.....	128
3.5.4	ControlBox und ParameterBox.....	133
3.5.5	Infoparameter .....	134
3.5.6	PLC Fehler .....	139
3.5.7	PLC Parameter.....	140
3.6	Sprachen.....	142
3.6.1	Anweisungsliste (AWL / IL).....	142
3.6.1.1	Allgemein	142
3.6.2	Strukturierter Text (ST).....	146
3.6.2.1	Allgemein	146
3.6.2.2	Anweisungen	148
3.7	Sprünge .....	152
3.7.1	JMP .....	152
3.7.2	JMPC.....	152
3.7.3	JMPCN .....	152
3.8	Typkonvertierung .....	153
3.8.1	BOOL_TO_BYTE .....	153
3.8.2	BYTE_TO_BOOL .....	153
3.8.3	BYTE_TO_INT .....	154
3.8.4	DINT_TO_INT .....	154
3.8.5	INT_TO_BYTE .....	155
3.8.6	INT_TO_DINT .....	155
3.9	PLC Störmeldungen.....	156
<b>4</b>	<b>Parameter.....</b>	<b>157</b>
<b>5</b>	<b>Anhang.....</b>	<b>158</b>
5.1	Service- und Inbetriebnahmehinweise .....	158
5.2	Dokumente und Software.....	158
5.3	Abkürzungen.....	159

# 1 Einleitung

## 1.1 Allgemeines

### 1.1.1 Dokumentation

Bezeichnung:	<b>BU 0550</b>
Materialnummer:	<b>6075501</b>
Reihe:	<b>PLC - Funktionalität für Frequenzumrichter und Motorstarter der Baureihen</b>
<b>NORDAC PRO</b>	(SK 500P ... SK 550P) (SK 520E ... SK 545E)
<b>NORDAC Flex</b>	(SK 200E ... SK 235E)
<b>NORDAC Base</b>	(SK 180E / SK 190E)
<b>NORDAC Link</b>	(SK 250E-FDS ... SK 280E-FDS)
<b>NORDAC Link</b>	(SK 155E-FDS / SK 175E-FDS)
<b>NORDAC ON/ON+</b>	(SK 300P)

### 1.1.2 Dokumenthistorie

Ausgabe	Baureihe	Version	Bemerkungen
Bestellnummer		Software	
<b>BU 0550</b> , September 2011 <b>6075501/ 3911</b>	SK 540E ... SK 545E	V 2.0 R0	Erste Ausgabe
Weitere Überarbeitungen: Oktober, 2011, Februar 2013, Februar 2017, Mai 2019 Eine Übersicht über den Änderungsinhalt o.g. Ausgaben ist im jeweiligen Dokument zu finden.			
<b>BU 0550</b> , Januar 2021 <b>6075501/ 0321</b>	SK 500P ... SK 550P SK 540E ... SK 545E	V 1.2 R2 V 2.4 R2	<ul style="list-style-type: none"> <li>• Implementierung der Gerätetypen NORDAC ON/ON+ SK 300P</li> <li>• Anpassungen und Korrekturen</li> </ul>
	SK 520E ... SK 535E	V 3.2 R2	
	SK 200E ... SK 235E	V 2.2 R1	
	SK 180E / SK 190E	V 1.3 R0	
	SK 250E-FDS ... SK 280E-FDS	V 1.3 R1	
	SK 155E-FDS / SK 175E-FDS	V 1.2 R1	
	SK 300P	V 1.0 R1	

### 1.1.3 Urheberrechtsvermerk

Das Dokument ist als Bestandteil des hier beschriebenen Gerätes bzw. der hier beschriebenen Funktionalität jedem Nutzer in geeigneter Form zur Verfügung zu stellen.

Jegliche Bearbeitung oder Veränderung des Dokuments ist verboten.

### 1.1.4 Herausgeber

#### **Getriebebau NORD GmbH & Co. KG**

Getriebebau-Nord-Straße 1

22941 Bargteheide, Germany

<http://www.nord.com/>

Fon +49 (0) 45 32 / 289-0

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


### 1.1.5 Zu diesem Handbuch

Dieses Handbuch soll Ihnen bei der Inbetriebnahme der PLC-Funktionalität eines Frequenzumrichters bzw. Motorstarters der Getriebebau NORD GmbH & Co. KG (kurz NORD) helfen. Es richtet sich an Elektrofachkräfte, die die PLC Programme für das Gerät planen, projektieren, installieren und einrichten (📖 Abschnitt 2.1 "Auswahl und Qualifikation des Personals"). Die in diesem Handbuch enthaltenen Informationen setzen voraus, dass die mit der Arbeit betrauten Elektrofachkräfte mit dem Umgang mit elektronischer Antriebstechnik, insbesondere den Geräten aus dem Hause NORD, vertraut sind.

Dieses Handbuch enthält ausschließlich Informationen und Beschreibungen der PLC-Funktionalität und die für die PLC-Funktionalität relevanten Zusatzinformationen zum Gerät der Getriebebau NORD GmbH & Co. KG.



## 1.2 Mitgeltende Dokumente

Dieses Handbuch ist nur zusammen mit der Betriebsanleitung des eingesetzten Gerätes gültig. Nur gemeinsam mit diesem Dokument stehen alle für eine sichere Inbetriebnahme der Antriebsaufgabe erforderlichen Informationen zur Verfügung. Eine Liste der Dokumente finden Sie im  Abschnitt 5.2 "Dokumente und Software".

Die erforderlichen Dokumente finden Sie unter [www.nord.com](http://www.nord.com).

## 1.3 Darstellungskonventionen

### 1.3.1 Warnhinweise

Warnhinweise für die Sicherheit der Benutzer und der Busschnittstellen sind wie folgt gekennzeichnet:

#### **GEFAHR**

Dieser Warnhinweis warnt vor Personengefährdungen, die zu schweren Verletzungen oder zum Tod führen.

#### **WARNUNG**

Dieser Warnhinweis warnt vor Personengefährdungen, die zu schweren Verletzungen oder zum Tod führen können.

#### **VORSICHT**

Dieser Warnhinweis warnt vor Personengefährdungen, die zu leichten bis mittelschweren Verletzungen führen können.

#### **ACHTUNG**


Dieser Warnhinweis warnt vor Sachschäden.

### 1.3.2 Andere Hinweise

#### **Information**

Dieser Hinweis zeigt Tipps und wichtige Informationen.

## 1.4 Bestimmungsgemäße Verwendung

Die PLC-Funktionalität der Getriebebau NORD GmbH & Co. KG ist eine softwaregestützte, funktionale Erweiterung für Frequenzumrichter und Motorstarter aus dem Hause NORD. Sie ist untrennbar mit dem jeweiligen Gerät verbunden und unabhängig von ihm nicht verwendbar. Es gelten somit uneingeschränkt die spezifischen Sicherheitshinweise des jeweiligen Gerätes, die dem betreffenden Handbuch zu entnehmen sind ( Abschnitt 5.2 "Dokumente und Software").

Die PLC-Funktionalität dient im Wesentlichen der Lösung komplexer Antriebsaufgaben mit einem oder mehreren Geräten der elektronischen Antriebstechnik, sowie der Vereinfachung antriebsnaher Ansteuerungs- und Überwachungsfunktionen durch ein entsprechend ausgestattetes Gerät.

## 2 Sicherheit

### 2.1 Auswahl und Qualifikation des Personals

Die PLC-Funktionalität darf nur von qualifizierten Elektrofachkräften in Betrieb genommen werden. Diese müssen das erforderliche Wissen über die PLC-Funktionalität, über die verwendete elektronische Antriebstechnik sowie die verwendeten Konfigurationshilfsmittel (z.B. NORD CON – Software) und die mit der Antriebsausgabe im Zusammenhang stehenden Peripherie (u. A. die Steuerung) haben.

Die Elektrofachkräfte müssen darüber hinaus mit der Installation, Inbetriebnahme und dem Betrieb von Sensoren und elektronischer Antriebstechnik vertraut sein und alle am Einsatzort geltenden Unfallverhütungsvorschriften, Richtlinien und Gesetze kennen und befolgen.

#### 2.1.1 Qualifiziertes Personal

Zum qualifizierten Personal gehören Personen, die aufgrund ihrer fachlichen Ausbildung und Erfahrung ausreichende Kenntnisse auf einem speziellen Sachgebiet haben und mit den entsprechenden einschlägigen Arbeitsschutz- und Unfallverhütungsvorschriften sowie den allgemein anerkannten Regeln der Technik vertraut sind.


Die Personen müssen vom Betreiber der Anlage berechtigt worden sein, die jeweils erforderlichen Tätigkeiten auszuführen.

#### 2.1.2 Elektrofachkraft

Eine Elektrofachkraft ist eine Person, die aufgrund ihrer fachlichen Ausbildung und Erfahrung ausreichende Kenntnisse besitzt hinsichtlich


- des Einschaltens, Abschaltens, Freischaltens, Erdens und Kennzeichnens von Stromkreisen und Geräten,
- der ordnungsgemäßen Wartung und Anwendung von Schutzeinrichtungen entsprechend festgelegter Sicherheitsstandards,
- der Notversorgung von Verletzten.

### 2.2 Sicherheitshinweise

Verwenden Sie die Technologiefunktion **PLC Funktionalität** und das Gerät der Getriebebau NORD GmbH & Co. KG ausschließlich bestimmungsgemäß,  Abschnitt 1.4 "Bestimmungsgemäße Verwendung".

Für einen gefahrlosen Einsatz der Technologiefunktion beachten Sie die Vorgaben in diesem Handbuch.

Nehmen Sie das Gerät nur technisch unverändert und nicht ohne erforderliche Abdeckungen in Betrieb. Achten Sie darauf, dass alle Anschlüsse und Kabel in einwandfreiem Zustand sind.

Arbeiten an und mit dem Gerät dürfen nur von qualifiziertem Personal ausgeführt werden,  Abschnitt 2.1 "Auswahl und Qualifikation des Personals".

## **3 PLC**

### **3.1 Allgemeines**

NORD Frequenzumrichter der Baureihen SK 180E/SK 190E, SK 2xxE, SK 2xxE-FDS, SK 300P, SK 520E – SK 545E und SK 5xxP sowie die Motorstarter der Baureihe SK 155E-FDS/SK 175E-FDS enthalten eine Logikverarbeitung, welche an die für Speicherprogrammierbare Steuerungen (SPS / PLC) geltende Norm IEC61131-3 angelehnt ist. Die Reaktionsgeschwindigkeit oder Rechenleistung dieser PLC ist geeignet kleinere Aufgaben im Umfeld des Umrichters zu übernehmen. So können Umrichter-Eingänge oder über einen Feldbus ankommende Informationen überwacht, ausgewertet und in entsprechende Sollwerte für den Frequenzumrichter weiterverarbeitet werden. Im Zusammengehen mit anderen NORD Geräten ist auch eine Visualisierung von Anlagenzuständen und Eingabe von speziellen Kundenparametern möglich. Somit ergibt sich im begrenzten Bereich ein Einsparungspotential über das Weglassen einer bisherigen externen PLC Lösung. Als Programmiersprache wird AWL unterstützt. AWL ist eine maschinennahe textbasierende Programmiersprache, deren Umfang und Anwendung in der IEC61131-3 festgelegt ist.

---

#### **Information**

Die Programmierung und der Download in das Gerät erfolgen ausschließlich über die NORD Software NORDCON.

---

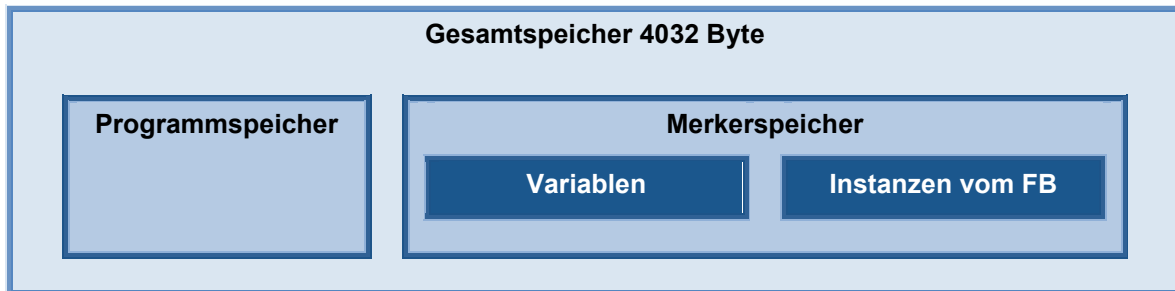
### 3.1.1 Spezifikation der PLC

Funktion	Spezifikation		
Standard	An IEC61131-3 angelehnt		
Sprache	Instruction List ( IL ), strukturierter Text (ST)		
Task	Ein zyklischer Task, Programmaufruf alle 5ms		
Rechenleistung	Zirka 200 AWL Befehle auf 1ms		
Programmspeicher	SK 5xxP, SK 520E ... SK 545E, SK 2xxE, SK 2x0E-FDS, On, On+	SK 190E / SK 180E	SK 155E-FDS / SK 175E-FDS
	8128 Byte für Merker, Funktionen und das PLC Programm	2032 Byte für Merker, Funktionen und das PLC Programm	2028 Byte für Merker, Funktionen und das PLC Programm
Max. mögliche Anzahl von Befehlen	ungefähr 2580 Befehle	ungefähr 660 Befehle	ungefähr 660 Befehle  <b>Hinweis:</b> Dies ist ein Durchschnittswert, eine starke Verwendung von Merkern, Prozessdaten und Funktionen minimiert die mögliche Zeilenanzahl erheblich, siehe Abschnitt Ressourcen.
Frei ansprechbare CAN Mailboxen	20 (außer On/On+)		
Unterstützte Geräte	SK 5xxP SK 54xE SK 53xE / SK 52xE ab V3.0 On/On+ SK 2xxE ab V2.0 SK 2x0E-FDS SK 180E / SK 190E SK 155E-FDS / SK 175E- FDS		

### 3.1.2 PLC Aufbau

#### 3.1.2.1 Speicher

Der Speicher in der PLC wird in Programm- und Merkerspeicher unterteilt. Im Bereich des Merkerspeichers werden neben den Variablen auch die Instanzen von Funktionsblöcken abgelegt. Eine Instanz ist ein Speicherbereich, in dem alle internen Ein- und Ausgabewerte eines FB abgelegt werden. Jede FB Deklaration benötigt eine eigene Instanz. Die Grenze zwischen Programm- und Merkerspeicher wird dynamisch festgelegt, abhängig von der Größe des Merkerbereiches.



Im Merkerspeicher werden im Bereich Variablen zwei verschiedene Klassen abgelegt:

#### [VAR]

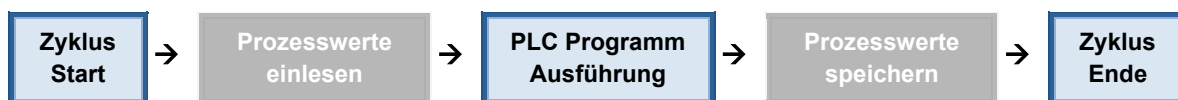
Speichervariable zum Ablegen von Hilfsinformationen und Zuständen. Variablen dieses Typs werden bei jedem Start der PLC neu initialisiert. Während des zyklischen Ablaufs der PLC bleiben die Speicherinhalte erhalten.

#### [VAR\_ACCESS]

Dient zum Einlesen und Beschreiben von Prozessdaten (Eingänge, Ausgänge, Sollwerte, usw.) des Frequenzumrichters. Diese Werte werden bei jedem PLC Zyklus neu erzeugt

#### 3.1.2.2 Prozessabbild

Das Gerät verfügt über etliche physikalische Größen wie Drehmoment, Drehzahl, Position, Eingänge, Ausgänge, usw. Diese Größen unterteilen sich in Ist- und Sollwerte. Sie können im Prozessabbild der PLC geladen und beeinflusst werden. Die benötigten Prozesswerte müssen in der Variablenliste unter der Klasse VAR\_ACCESS definiert werden. Mit jedem PLC Zyklus werden alle in der Variablenliste definierten Prozessdaten des Umrichters neu eingelesen. Am Ende jedes PLC Zyklus werden die beschreibbaren Prozessdaten wieder dem Umrichter übergeben, siehe nachfolgende Abbildung.



Aufgrund dieses Ablaufes ist es wichtig, einen zyklischen Programmablauf zu programmieren. Das Programmieren von Schleifen, um auf bestimmte Ereignisse zu warten (z.B. Pegeländerung an einem Eingang), führt nicht zum gewünschten Ergebnis. Bei Funktionsblöcken, die auf Prozesswerte zugreifen, ist dieses Verhalten anders. Hier werden die Prozesswert mit dem Aufruf des Funktionsblockes gelesen und bei Beendigung des Blockes werden die Prozesswerte sofort geschrieben.

## **i** Information

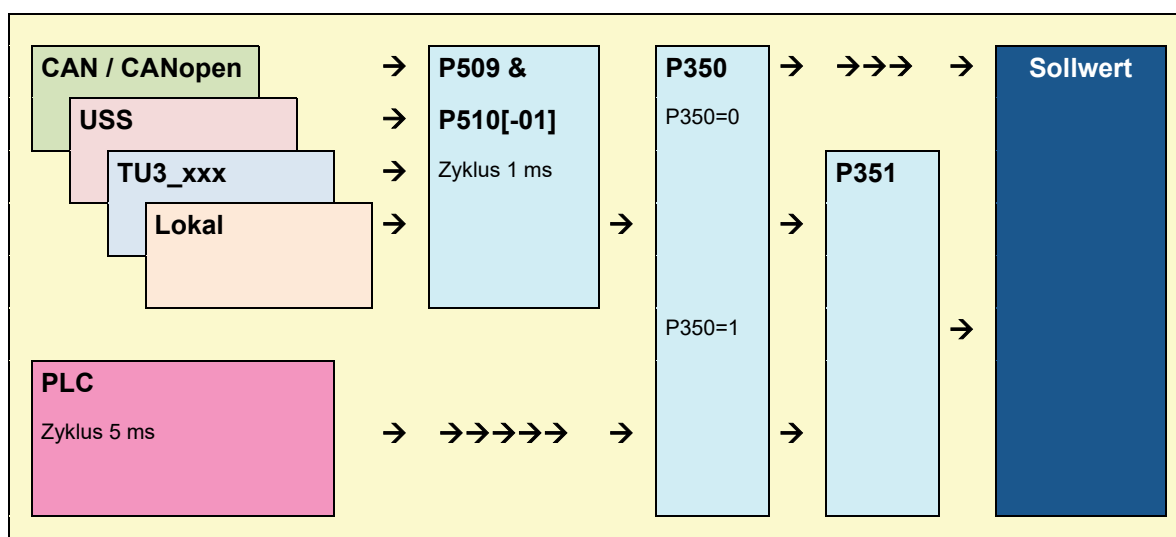
Werden Motion Blöcke MC\_Power, MC\_Reset, MC\_MoveVelocity, MC\_Move, MC\_Home oder MC\_Stop verwendet, dann dürfen die Prozesswerte „PLC\_Control\_Word“ und „PLC\_Set\_Val1“ bis „PLC\_Set\_Val5“ nicht verwendet werden. Anderenfalls würden die Werte in der Variablenliste immer die Änderung des Funktionsblockes überschreiben.

### 3.1.2.3 Programm Task

Die Programmausführung in der PLC erfolgt in einer einzigen Task. Die Task wird zyklisch alle 5 ms aufgerufen und ihre max. Bearbeitungsdauer beträgt 3 ms. Kann ein längeres Programm in dieser Zeit nicht abgearbeitet werden, dann wird die Programmausführung unterbrochen und in der nächsten 5 ms Task fortgeführt.

### 3.1.2.4 Sollwert Verarbeitung

Der Umrichter verfügt über eine Vielzahl von Sollwertquellen, die letztendlich über mehrere Parameter zu einem resultierenden Frequenzumrichter Sollwert miteinander verknüpft werden.



Bei aktivierter PLC (P350=1) erfolgt über die P509 & P510[-01] eine Vorselektion der von außen eingehenden Sollwerte (Hauptsollwerte). Über den P351 wird dann letztlich entschieden, welche Sollwerte von der PLC oder den über P509/P510[-01] eingehenden Werten genommen wird. Auch ein Mix aus beiden ist möglich. Bei den Nebensollwerten (P510[-02]) verändert sich im Zusammenhang mit der PLC Funktion nichts. Alle Nebensollwertquellen und die PLC übergeben ihre Nebensollwerte gleichberechtigt an den Frequenzumrichter.

### 3.1.2.5 Datenverarbeitung über Akku

Der Akkumulator bildet die zentrale Recheneinheit der PLC. Fast alle AWL-Befehle funktionieren nur im Zusammenhang mit dem Akkumulator. In der NORD PLC existieren gleich drei Akkumulatoren. Dabei handelt es sich um die 32Bit großen Akku1 und Akku2, sowie das AE im Format BOOL. Das AE wird für alle boolschen Lade-, Speicher- und Vergleichsoperationen herangezogen. Wird ein boolscher Wert geladen so wird er im AE dargestellt. Vergleichsoperanden liefern das Ergebnis im AE ab und bedingte Sprünge werden aufgrund des AE ausgelöst. Akku1 und Akku2 werden für alle Operanden im Datenformat BYTE, INT und DINT verwendet. Bei Akku1 handelt es sich um den Hauptakkumulator während Akku2 nur Hilfsfunktionen übernimmt. Alle Lade und Speicheroperanden laufen über Akku1. Alle arithmetischen Operatoren speichern ihr Ergebnis unter Akku1 ab. In Akku2

wird bei jedem Ladebefehl der Inhalt von Akku1 verschoben. Ein nachfolgender Operator kann dann beide Akkumulatoren miteinander verknüpfen oder auswerten und das Ergebnis wieder in Akku1, der im Folgenden auch allgemein als „Akku“ bezeichnet wird, speichern.

### **3.1.3 Funktionsumfang**

Die PLC unterstützt eine Vielzahl von Operatoren, Funktionen und Standardfunktionsbausteinen, die in der IEC1131-3 definiert sind. Eine detaillierte Darstellung ist in den nachfolgenden Kapiteln enthalten. Des Weiteren werden Funktionsblöcke erläutert, die zusätzlich unterstützt werden.

#### **3.1.3.1 Motion Control Lib**

Die Motion Control Lib ist an die PLCopen Specification „Function blocks for motion control“ angelehnt. In ihr sind hauptsächlich Funktionsblöcke zum Verfahren des Antriebs enthalten. Zusätzlich werden auch Funktionsblöcke zum Lesen und Schreiben von Geräteparametern bereitgestellt.

#### **3.1.3.2 Elektronisches Getriebe mit Fliegender Säge**

Der Frequenzumrichter verfügt über die Funktionen elektronisches Getriebe (Gleichlauf im Positioniermodus) und Fliegende Säge. Über diese Funktionen kann der Umrichter mit einem anderen Antrieb winkelsynchron mitfahren. Weiterhin ist es über die Zusatzfunktion Fliegende Säge möglich, sich positionsgenau auf einen fahrenden Antrieb zu synchronisieren. Der Betriebsmodus elektronisches Getriebe kann jederzeit gestartet und beendet werden. Damit ist eine Kombination von klassischer Lageregelung mit ihren Verfahrbefehlen und Getriebefunktion möglich. Für die Getriebefunktion wird an der Masterachse zwingend ein NORD Frequenzumrichter mit internem CAN-Bus benötigt.

#### **3.1.3.3 Visualisierung**

Mit Hilfe einer ControlBox bzw. einer ParameterBox sind die Visualisierung des Betriebszustandes und die Parametrierung des Frequenzumrichters möglich. Alternativ können auch über die CANopen Master Funktionalität der PLC CAN-Bus Panels zur Anzeige von Informationen verwendet werden.

##### **ControlBox**

Die einfachste Variante zur Visualisierung ist die ControlBox. Über zwei Prozesswerte kann auf das 4 stellige Display und den Zustand der Tastatur zugegriffen werden. Damit können sehr schnell einfache HMI Applikationen erstellt werden. Damit die PLC auf die Anzeige zugreifen kann muss der P001 auf „PLC-Controlbox Value“ eingestellt werden. Eine weitere Besonderheit ist, dass das Parametermenü nicht mehr über die Pfeiltasten erreicht wird. Stattdessen müssen die „On“ und „Enter“ Taste zeitgleich betätigt werden.

##### **ParameterBox**

Im Visualisierungsmodus kann über die PLC jedes der 80 Zeichen im P-Box Display (4 Zeilen a 20 Zeichen) gesetzt werden. Es ist möglich Zahlen wie auch Texte zu übertragen. Weiterhin können Tastatureingaben auf der P-Box von der PLC erfasst werden. Damit ist eine Realisierung komplexerer HMI Funktionen ( Anzeige von Istwerten, Bildwechsel, Übergabe von Sollwerten, usw. ) möglich. Der Zugriff auf die P-Box Anzeige erfolgt über Funktionsblöcke in der PLC. Die Visualisierung erfolgt über die Betriebswertanzeige der ParameterBox. Der Inhalt der Betriebswertanzeige wird über den P-Box Parameter P1003 eingestellt. Dieser Parameter befindet sich unter dem Hauptmenüpunkt „Anzeige“. P1003 muss auf den Wert „PLC-Anzeige“ eingestellt werden. Über die Pfeiltasten Rechts oder Links kann die Betriebswertanzeige danach wieder angewählt werden. Hier wird jetzt das von der PLC kontrollierte Display angezeigt. Diese Einstellung bleibt auch nach einem erneuten Einschalten erhalten.

### 3.1.3.4 Prozessregler

Der Prozessregler ist ein PID-T1 – Regler mit begrenzter Ausgangsgröße. Mit Hilfe dieses Funktionsbausteines können in der PLC auf einfache Weise komplexe Regelungen aufgebaut werden, über die sich etliche Prozesse, wie z.B. Druckregelungen, deutlich eleganter lösen lassen als mit den häufig verwendeten Zweipunktreglern.

### 3.1.3.5 CANopen Kommunikation

Neben den standardmäßig vorhandenen Kommunikationskanälen bietet die PLC noch weitere Möglichkeiten zu kommunizieren. Über die CAN Bus Schnittstelle des Frequenzumrichters bzw. über den Systembus kann dieser mit anderen Geräten zusätzliche Kommunikationsbeziehungen aufbauen. Das dabei verwendete Protokoll ist CANopen. Die Kommunikationsbeziehungen sind dabei auf den PDO Datentransfer und NMT Kommandos beschränkt. Die per Standard im Frequenzumrichter vorhandene CANopen Kommunikation über SDO, PDO1, PDO2 und Broadcast bleibt von dieser PLC - Funktion unbeeinträchtigt.

#### **PDO (Prozess Daten Objects)**


Über PDO können andere Frequenzumrichter gesteuert und überwacht werden. Es ist aber auch möglich Geräte anderer Anbieter an die PLC anzubinden. Dies können IO-Baugruppen, CANopen Geber, Panels, usw. sein. Damit kann die Anzahl der Ein/Ausgänge des Frequenzumrichters beliebig erweitert werden, auch analoge Ausgänge wären dann möglich.

#### **NMT (Network Management Objects)**

Alle CANopen Geräte müssen vom Busmaster in den CANopen Bus State „Operational“ gebracht werden. Erst in diesem Buszustand ist eine PDO Kommunikation möglich. Wenn sich kein Busmaster in dem CANopen Bus befindet, muss dies durch die PLC erfolgen. Für diesen Zweck gibt es den Funktionsbaustein FB\_NMT.



## 3.2 Erstellen von PLC Programmen

Die Erstellung der PLC Programme erfolgt ausschließlich über das PC-Programm NORDCON. Der PLC Editor wird entweder über den Menüpunkt „Datei/Neu/PLC Programm“ oder durch das Symbol  geöffnet. Diese Schaltfläche ist nur aktiv, wenn in der Geräteübersicht ein Gerät mit PLC Funktionalität den Fokus hat.

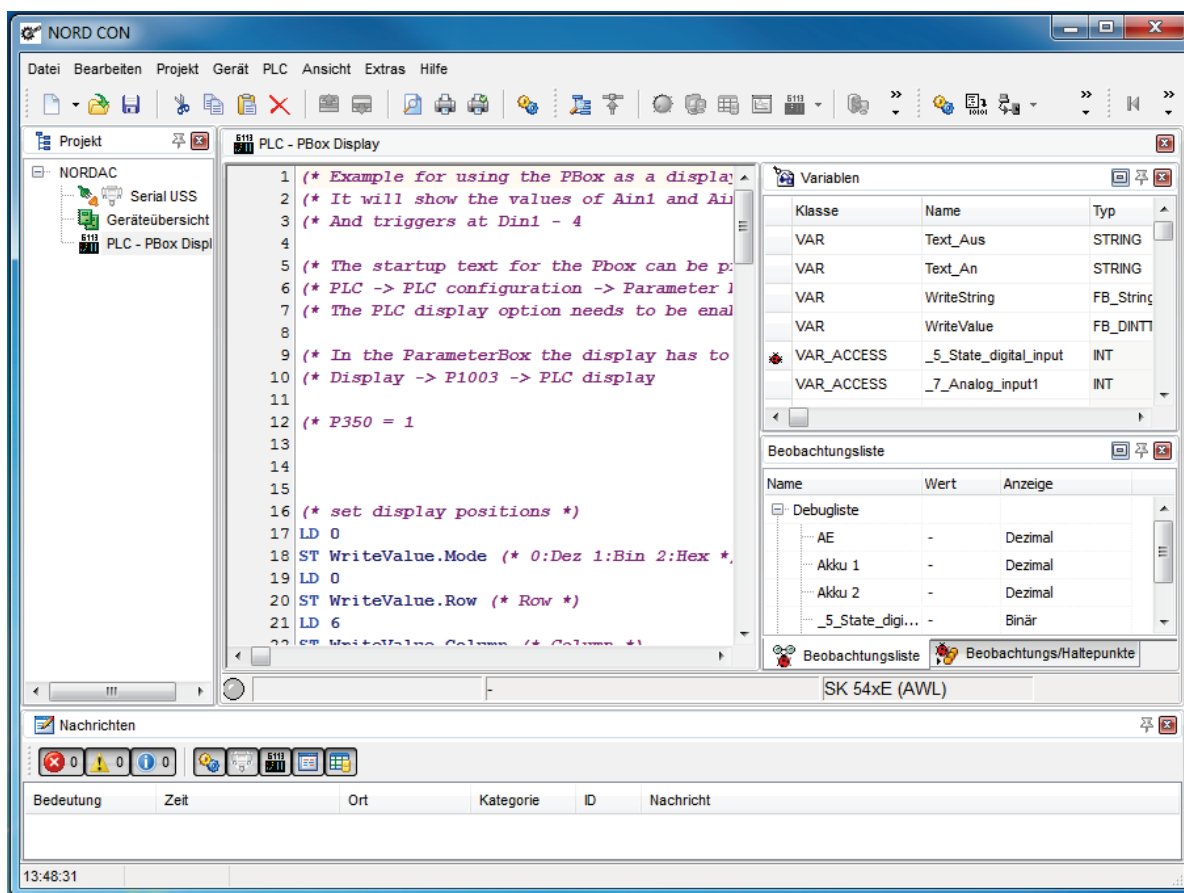
### 3.2.1 Laden, Speichern & Drucken

Die Funktionen Laden, Speichern und Drucken erfolgen über die entsprechenden Einträge im Hauptmenü oder die Symbolleisten. Beim Öffnen ist es empfehlenswert, im Dialog „Öffnen“ den Dateityp auf „PLC Programm“ (\*.awl, \*.nstx) zu setzen. Damit werden nur noch Dateien, die vom PLC Editor gelesen werden können, angezeigt. Soll das erstellte PLC Programm gespeichert werden, dann muss das Fenster vom PLC Editor aktiv sein. Das PLC Programm wird durch Betätigen von „Speichern“ oder „Speichern unter“ gesichert. Bei der Operation „Speichern unter“ kann dies auch am Eintrag des Dateityp (Programm PLC (\*.awl\*.nstx)) erkannt werden. Für das Drucken des PLC Programmes muss auch das entsprechende PLC Fenster aktiv sein. Der Ausdruck wird dann über „Datei/Drucken“ oder das passende Symbol gestartet.

PLC Programme können zusätzlich auch als gesichertes PLC Programm gespeichert werden. Hierfür muss der Benutzer im Dateiauswahldialog den Dateityp auf "AWL Dateien gesichert" oder "ST Dateien gesichert" einstellen. Anschließend wird das PLC Programm in einer verschlüsselten (\*.awls oder \*.nsts) und normalen Version (\*.awl, \*.nstx) abgespeichert. Das verschlüsselte PLC Programm kann nur noch zum Gerät übertragen werden (siehe ).

### 3.2.2 Editor

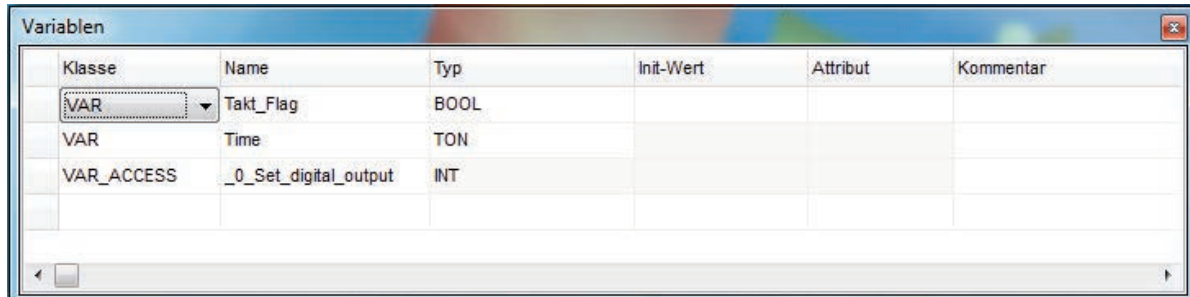
Der PLC – Editor ist in vier verschiedene Fenster aufgeteilt.



Die einzelnen Fenster werden in den nachfolgenden Abschnitten näher erläutert.

### 3.2.2.1 Variablen und FB Deklaration

In diesem Fenster werden alle im Programm benötigten Variablen, Prozesswerte und Funktionsblöcke deklariert.



#### Variablen

Variablen werden angelegt, indem die Klasse „VAR“ eingestellt wird. Der Name für die Variable ist frei wählbar. Im Feld Typ kann zwischen BOOL, BYTE, INT und DINT gewählt werden. Für die Variablen kann eine Startinitialisierung unter Init-Wert eingetragen werden.

#### Prozesswerte

Diese werden angelegt indem unter Klasse der Eintrag „VAR\_ACCESS“ selektiert wird. Der Name ist nicht frei wählbar und das Feld Init-Wert ist für diesen Typ gesperrt.

#### Funktionsbausteine

Unter Klasse wird der Eintrag „VAR“ selektiert. Der Name für die jeweilige Instanz des Funktionsbausteins (FB) ist frei wählbar. Der gewünschte FB wird unter Typ selektiert. Ein Init-Wert ist für FB nicht einstellbar.

Alle Menüpunkte, die das Variablenfenster betreffen, werden über das Kontextmenü aufgerufen. Hierüber können Einträge hinzugefügt und gelöscht werden. Sowie Variablen und Prozessvariablen zur Beobachtung (Watchpoint Funktion) oder zum Debuggen (Breakpoint) aktiviert werden.

### 3.2.2.2 Eingabefenster

Das Eingabefenster dient zur Programmeingabe und auch Darstellung des AWL-Programmes. Es verfügt über folgende Funktionen:

- Syntax Hervorhebung
- Lesezeichen
- Variablen Deklaration
- Debugging

#### **Syntax Hervorhebung**

Werden der Befehl und die ihm zugeordnete Variable vom Editor erkannt, dann wird der Befehl blau und die Variable schwarz dargestellt. Solange dies nicht der Fall ist, erfolgt die Darstellung in dünner, schräger, schwarzer Schrift.

#### **Lesezeichen**

Da Programme im Editor durchaus eine beträchtliche Länge erreichen können, ist es möglich über die Funktion Lesezeichen wichtige Stellen im Programm zu markiert und gezielt anzuspringen. Zur Markierung einer Zeile muss sich der Cursor in der betreffenden Zeile befinden. Über den Menüpunkt „Lesezeichen umschalten“ (rechte Maustaste Menü) wird die Zeile mit dem gewünschten Lesezeichen markiert. Angesprungen werden die Lesezeichen über den Menüpunkt „Gehe zu Lesezeichen“.

#### **Variablen Deklaration**

Über das Editor Menü „Variable hinzufügen“ (rechte Maustaste) können vom Editor aus neue Variablen deklariert werden.

#### **Debugging**

Für die Funktion Debugging werden im Editor die Position der Break- und Watchpoints festgelegt. Dies kann über die Menüpunkte „Haltepunkt umschalten“ (Breakpoints) und „Beobachtungspunkt umschalten“ (Watchpoints) passieren. Die Position von Breakpoints kann zusätzlich über einen Klick auf der linken Randleise des Editorfensters festgelegt werden. Variablen und Prozesswerte, die während des Debuggings aus dem Frequenzumrichter ausgelesen werden sollen, müssen markiert werden. Dies kann im Editor über die Menüpunkte „Variable debuggen“ und „Variable beobachten“ erfolgen. Dazu muss die entsprechende Variable markiert sein, bevor der gewünschte Menüpunkt angewählt wird.

### 3.2.2.3 Watch- & Breakpoint Anzeigefenster

Dieses Fenster verfügt über zwei Tab Reiter die nachfolgend erläutert werden.

#### **Haltepunkte**

In diesem Fenster sind alle gesetzten Breakpoint und Watchpoints zu sehen. Sie können über die Checkboxes ein-/ausgeschaltet und über die „Entfernen Taste“ gelöscht werden. Über die rechte Maustaste kann ein entsprechendes Menü aufgerufen werden.

#### **Beobachtungsliste**

Hier werden alle zur Beobachtung ausgewählten Variablen dargestellt. In der Spalte Wert wird ihr aktueller Inhalt dargestellt. Über die Spalte Anzeige kann das Darstellungsformat ausgewählt werden.

### 3.2.2.4 PLC Meldungsfenster

In diesem Fenster werden alle Status- und Fehlermeldungen der PLC eingetragen. Für ein korrekt übersetztes Programm erscheint die Meldung „Fehlerfrei übersetzt“. Eine Zeile tiefer wird der Ressourcenverbrauch angezeigt. Bei Fehlern im PLC Programm erscheint die Meldung „Fehler X“, in X wird die Anzahl der Fehler dargestellt. In den folgenden Zeilen erscheint die konkrete jeweilige Fehlermeldung im Format:

[ Zeilennummer]: Fehlerbeschreibung

### 3.2.3 Programm zum Gerät übertragen

Es gibt mehrere Wege, um ein PLC Programm zum Gerät zu übertragen.


#### PLC Programm direkt übertragen:

1. Gerät im Projektbaum auswählen.
2. Kontextmenü öffnen (rechte Maustaste drücken)
3. Funktion "PLC Programm zum Gerät übertragen" ausführen
4. Datei im Dateiauswahldialog auswählen und "Öffnen" drücken

#### PLC Programm mit den PLC Editor übertragen (Offline):

1. PLC Programm mit der Funktion "Öffnen" (Datei->Öffnen) öffnen
2. PLC Editor mit einem Gerät verbinden (PLC->Verbinden)
3. PLC Programm übersetzen
4. PLC Programm zum Gerät übertragen

#### PLC Programm mit den PLC Editor übertragen (Online):

1. Gerät im Projektbaum markieren
2. PLC Editor starten
3. PLC Programm öffnen
4. PLC Programm in die Online-Ansicht importieren
5. PLC Programm übersetzen
6. PLC Programm zum Gerät übertragen 



#### Information

#### SK 1xxE-FDS - begrenzte Anzahl an Schreibzyklen

In den Geräten SK 155E-FDS / SK 175E-FDS wird als Speichermedium ein Flash eingesetzt. Die Anzahl der Schreibzyklen eines Flashspeichers ist stark begrenzt. Deshalb wird standardmäßig das Programm nur in den RAM geladen. Es kann anschließend gestartet und getestet werden. Soll die PLC anschließend neu gestartet werden, muss das Programm erneut zum Gerät geladen werden, um die PLC Variablen zu initialisieren. Soll das Programm dauerhaft im Gerät gespeichert werden, muss der Benutzer die Aktion "Programm zum Gerät übertragen und speichern" ausführen.

---

### 3.2.4 Debugging

Da Programme nur in seltenen Fällen auf Anhieb funktionieren bietet die NORD PLC einige Möglichkeiten zur Fehlerfindung. Diese Möglichkeiten lassen sich grob in zwei Punkte unterteilen, auf die jetzt nachfolgend eingegangen wird.

#### 3.2.4.1 Beobachtungspunkte (Watchpoints)

Die einfachste Debugging Variante ist die Watchpoint Funktion. Sie bietet einen schnellen Überblick über das Verhalten einiger Variablen. Dazu wird an beliebiger Stelle im Programm ein Beobachtungspunkt gesetzt. Wenn die PLC diese Programmzeile abarbeitet, werden bis zu 5 Werte gespeichert und in der Beobachtungsliste angezeigt (Fenster „Beobachtungsliste“). Die 5 zu beobachtenden Werte können im Eingabefenster oder Variablenfenster über das Kontextmenü ausgewählt werden. Wurde ein Watchpoint an eine Stelle ohne Programmcode gesetzt, sucht NORDCON die vorherige Codezeile. Wird diese Codezeile im Programmablauf erreicht, wird die Aktualisierung der Werte ausgeführt. Wird ein Watchpoint durch einen Sprung (JMP, IF, Switch Anweisung) übersprungen, werden keine Werte aktualisiert.

---


#### Information


Variablen von Funktionsblöcken können in der aktuellen Version nicht zur Watchliste hinzugefügt werden!

---

#### 3.2.4.2 Haltepunkte (Breakpoints)


Über Haltepunkte ist es möglich das PLC Programm gezielt an einer gewünschten Programmzeile zu stoppen. Wenn die PLC in einen Haltepunkt hineinläuft werden das AE, Akku1 und Akku2 ausgelesen, sowie alle Variablen, die über den Menüpunkt „Variable debuggen“ (Kontextmenü) selektiert wurden. Es können bis zu 5 Breakpoints im PLC Programm gesetzt werden. Gestartet wird diese Funktion


über das Symbol . Das Programm läuft nun solange bis ein Haltepunkt ausgelöst wird. Eine erneute Betätigung der Symbolleiste lässt das Programm wieder frei laufen bis der nächste Haltepunkt



kommt. Soll das Programm wieder frei laufen, so wird das Symbol  betätigt.

### 3.2.4.3 Einzelschritt (Single Step)

Mit dieser Debugging Methode ist es möglich das PLC Programm Zeile für Zeile in Einzelschritten abzuarbeiten. Mit jedem Einzelschritt werden alle ausgewählten Variablen aus der Geräte-PLC ausgelesen und im Fenster „Beobachtungsliste“ angezeigt. Die zu beobachtenden Werte können im Eingabefenster oder Variablenfenster über das rechte Maustastenmenü ausgewählt werden. Voraussetzung für das Debugging in Einzelschritten ist, dass vor dem Start des Debugging


mindestens ein Haltepunkt gesetzt wurde. Durch Betätigung des Symbols  wird der Debugging Mode eingeschaltet. Erst wenn das Programm in den ersten Haltepunkt gelaufen ist, kann über das

Symbol  in Einzelschritten durch die nachfolgenden Zeilen debuggt werden. Hinter einigen Befehlszeilen verbergen sich mehrere einzelne Befehle. Dadurch kann es passieren das zwei oder mehr Einzelschritte abgearbeitet werden bevor im Eingabefenster die Schrittanzeige weiterspringt. Die aktuelle Position wird über einen kleinen Pfeil am linken PLC Editorfenster angezeigt. Bei Betätigung

des Symbols  läuft das Programm bis zum nächsten Haltepunkt weiter. Soll das Programm wieder frei laufen, so wird das Symbol  betätigt.



### 3.2.5 PLC Konfiguration

Über das Symbol  wird der PLC Konfigurationsdialog geöffnet. Hier können einige grundsätzliche Einstellungen für die PLC vorgenommen werden, auf die nachfolgend eingegangen wird.

#### **Überwachung der Zykluszeit**

Diese Funktion überwacht die max. Bearbeitungszeit für einen PLC Zyklus. Somit können ungewollt programmierte Dauerschleifen im PLC Programm abgefangen werden. Im Falle einer Überschreitung wird im Frequenzumrichter der Fehler E22.4 ausgelöst.

#### **ParameterBox Funktionsbaustein zulassen**

Soll im PLC Programm eine Visualisierung über die ParameterBox erfolgen, dann muss diese Option aktiviert sein. Andernfalls erzeugen die entsprechenden Funktionsblöcke beim Start des Frequenzumrichters einen Compiler Fehler.

#### **Ungültige Steuerdaten**

Die PLC kann die über die möglichen Bussysteme eingehenden Steuerwörter auswerten. Jedoch kommen die Steuerwörter nur durch, wenn das Bit „PZD gültig“ (Bit 10) gesetzt ist. Sollen auch nicht USS Protokoll konforme Steuerwörter von der PLC ausgewertet werden können, dann muss diese Option aktiviert sein. Bit 10 im ersten Wort wird dann nicht mehr abgefragt.

#### **Warmstart nach Fehler**

Alle Variablen werden beim Start der PLC immer mit „0“ oder ihren Initialisierungswert geladen. Dabei ist es egal ob der Start nach einem Stopp, Programmdownload oder PLC Fehler erfolgt. Über diese Option wird bei einem Warmstart der Inhalt der Variablen nicht verändert. Ein Warmstart erfolgt nach einem PLC Stopp Kommando oder einem PLC Fehler.

#### **Systemzeit beim Haltepunkt nicht anhalten**

Während des Debuggings, wenn die PLC im Haltepunkt oder sich im Einzelschrittmode befindet, wird die Systemzeit angehalten. Die Systemzeit bildet die Grundlage für alle Timer in der PLC. Soll die Systemzeit auch während des Debuggings weiterlaufen, dann ist diese Funktion zu aktivieren.

### 3.3 Funktionsblöcke

Funktionsblöcke sind kleinere Programme, die ihre Zustandswerte in internen Variablen ablegen können. Aus diesem Grund muss für jeden Funktionsblock eine eigene Instanz in der Variablenliste von NORDCON erzeugt werden. Soll z.B. ein Timer parallel 3 Zeiten überwachen, so muss er in der Variablenliste auch dreimal angelegt werden.

#### **i** Information

#### Erkennen einer Signalflanke

Damit die nachfolgenden Funktionsblöcke eine Flanke am Eingang erkennen können, ist es notwendig, dass der Funktionsaufruf zwei Mal mit unterschiedlichen Zuständen am Eingang durchlaufen wird.

#### 3.3.1 CANopen

Die PLC kann über Funktionsblöcke PDO-Kanälen konfigurieren, überwachen und auf ihnen senden. Über ein PDO können von der PLC bis zu 8 Byte Prozessdaten gesendet oder empfangen werden. Jedes dieser PDO wird über eine eigene Adresse (COB-ID) angesprochen. In der PLC können bis 20 PDO's konfiguriert werden. Zur einfacheren Bedienung wird nicht die COB-ID direkt eingegeben. Stattdessen werden Geräteadresse und die PDO Nummer an den FB übergeben. Die resultierende COB-ID wird auf Basis des Pre-Defined Connection Set (CiA DS301) ermittelt. Dadurch ergeben sich folgende mögliche COB-ID's für die PLC.

Sende PDO		Überwachte PDO	
PDO	COB-ID	PDO	COB-ID
PDO1	200h + Geräteadresse	PDO1	180h + Geräteadresse
PDO2	300h + Geräteadresse	PDO2	280h + Geräteadresse
PDO3	400h + Geräteadresse	PDO3	380h + Geräteadresse
PDO4	500h + Geräteadresse	PDO4	480h + Geräteadresse

NORD Frequenzumrichter benutzen zur Prozessdatenübermittlung PDO1, nur für Soll-/Istwert 4 und 5 wird PDO2 verwendet.

##### 3.3.1.1 Überblick

Funktionsbaustein	Erläuterung
FB_PDConfig	PDO Konfiguration
FB_PDOSend	PDO senden
FB_PDORceive	PDO empfangen
FB_NMT	PDO freigeben und sperren

3.3.1.2 FB\_NMT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X		X	X	X	X

Nach einem *Power UP* befinden sich alle CAN Teilnehmer im Bus-Zustand Pre-Operational. In diesem Zustand können sie weder PDO empfangen noch senden. Damit die PLC mit anderen Teilnehmern auf dem CAN Bus kommunizieren kann, müssen diese in den Zustand Operational gesetzt werden. Im Regelfall übernimmt dies der Busmaster. Sollte es keinen Busmaster geben, so kann diese Aufgabe vom FB\_NMT übernommen werden. Über die Eingänge **PRE**, **OPE** oder **STOP** kann der Zustand aller am Bus angeschlossenen Teilnehmer beeinflusst werden. Die Eingänge werden mit einer positiven Flanke an **EXECUTE** übernommen. Die Funktion muss solange aufgerufen werden, bis der Ausgang **DONE** oder **ERROR** auf 1 gesetzt wurde.

Wenn der Ausgang **ERROR** auf 1 gesetzt wurde, dann liegt entweder keine 24V Versorgung an der RJ45 CAN Buchse des Umrichters an oder der CAN – Treiber des Umrichters ist im Status *Bus off*. Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
EXECUTE	Ausführen	BOOL	DONE	NMT Befehl wird gesendet	BOOL
PRE	Setze alle Teilnehmer in den State Pre-Operational	BOOL	ERROR	Fehler im FB	BOOL
OPE	Setze alle Teilnehmer in den State Operational	BOOL			
STOP	Setze alle Teilnehmer in den State Stopped	BOOL			

### 3.3.1.3 FB\_PDOConfig

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X		X	X	X	X

Über diesen FB werden die PDO's konfiguriert. Mit einer Instanz dieser Funktion können alle gewünschten PDO's konfiguriert werden. Für jedes PDO muss der FB nur einmal aufgerufen werden. Es können bis zu 20 PDO eingerichtet werden. Jedes PDO hat seine eigene Parametrierung. Die Zuordnung der PDO's in den anderen CANopen FB's erfolgt über die Messagebox Number. Die **TARGETID** stellt die Geräteadresse dar. Bei NORD Frequenzumrichter wird diese im P515 oder über DIP Schalter eingestellt. Unter PDO wird die gewünscht Messagebox-Nummer eingetragen (siehe Einleitung). **LENGTH** legt die Sendelänge eines PDO fest. Über **DIR** wird die Send-/Empfangsrichtung festgelegt. Mit der positiven Flanke am **EXECUTE** Eingang werden die Daten übernommen. Der **DONE** Ausgang kann sofort nach Aufruf des FB abgefragt werden. Wenn **DONE** auf 1 gesetzt ist, dann wurde der PDO-Kanal konfiguriert. Bei **ERROR** = 1 gab es ein Problem, die genaue Ursache ist in **ERRORID** abgelegt. Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

Sende PDO		Überwachte PDO	
PDO	COB-ID	PDO	COB-ID
PDO1	200h + Geräteadresse	PDO1	180h + Geräteadresse
PDO2	300h + Geräteadresse	PDO2	280h + Geräteadresse
PDO3	400h + Geräteadresse	PDO3	380h + Geräteadresse
PDO4	500h + Geräteadresse	PDO4	480h + Geräteadresse
PDO5	180h + Geräteadresse	PDO5	200h + Geräteadresse
PDO6	280h + Geräteadresse	PDO6	300h + Geräteadresse
PDO7	380h + Geräteadresse	PDO7	400h + Geräteadresse
PDO8	480h + Geräteadresse	PDO8	500h + Geräteadresse

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Ausführen	BOOL	<b>DONE</b>	PDO konfiguriert	BOOL
<b>NUMBER</b>	Messagebox Nummer Wertebereich = 0 bis 19	BYTE	<b>ERROR</b>	Fehler im FB	BOOL
<b>TARGETID</b>	Geräteadresse Wertebereich = 1 bis 127	BYTE	<b>ERRORID</b>	Fehlercode	INT
<b>PDO</b>	PDO Wertebereich = 1 bis 4	BYTE			
<b>LENGTH</b>	PDO Länge Wertebereich = 1 bis 8	BYTE			
<b>DIR</b>	Senden oder Empfangen Senden = 1 / Empfangen = 0	BOOL			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1800h	Wertebereich Number überschritten				
1801h	Wertebereich TARGETID überschritten				
1802h	Wertebereich PDO überschritten				
1803h	Wertebereich LENGT überschritten				

** Information**
**Keine doppelte Verwendung der CAN ID**

Es dürfen keine CAN-ID parametrisiert werden, die das Gerät schon benutzt!

Betreffende Empfangsadressen:

- CAN ID = 0x180 + P515[-01]                      PDO1
- CAN ID = 0x180 + P515[-01]+1                CAN ID für Absolutwertgeber
- CAN ID = 0x280 + P515[-01]                      PDO2

Betreffende Sendeadressen:

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

**Beispiel in ST:**

```
(* PDO Konfigurieren *)
PDOConfig(
  Execute := TRUE,
  (* Messagebox 1 konfigurieren *)
  Number := 1,
  (* CAN Knotennummer setzen *)
  TargetID := 50,
  (* PDO wählen (Standard für PDO1 Steuerwort, Sollwert1, Sollwert2, Sollwert3) *)
  PDO := 1,
  (* Länge der Daten festlegen (Standard für PDO1 gleich 8 *)
  LENGTH := 8,
  (* Senden *)
  Dir := 1);
```

oder

```
(* PDO Konfigurieren *)
PDOConfig(
  Execute := TRUE,
  (* Messagebox 1 konfigurieren *)
  Number := 2,
  (* CAN Knotennummer setzen *)
  TargetID := 50,
  (* PDO wählen (Standard für PDO2 Sollwert4, Sollwert5 SK540E) *)
  PDO := 2,
  (* Länge der Daten festlegen (Standard für PDO2 gleich 4 *)
  LENGTH := 4,
  (* Senden *)
  Dir := 1);
```

oder

```
(* PDO Konfigurieren *)
PDOConfig(
  Execute := TRUE,
  (* Messagebox 2 konfigurieren *)
  Number := 2,
  (* CAN Knotennummer setzen *)
  TargetID := 50,
  (* PDO wählen (Standard für PDO1 Statuswort, Istwert1, Istwert2, Istwert3) *)
  PDO := 1,
  (* Länge der Daten festlegen (Standard für PDO1 gleich 8 *)
  LENGTH := 8,
  (* Empfangen *)
  Dir := 0);
```

3.3.1.4 FB\_PDORceive

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X		X	X	X	X

Dieser FB überwacht einen vorher konfigurierten PDO Kanal auf eingehende Botschaften. Die Überwachung startet, wenn der **ENABLE** Eingang auf 1 steht. Nach dem Aufruf der Funktion ist der **NEW** Ausgang zu prüfen. Wenn er auf 1 geht, dann ist eine neue Botschaft angekommen. Der **NEW** Ausgang wird mit dem nächsten Aufruf der Funktion gelöscht. In **WORD1** bis **WORD4** stehen die empfangenen Daten. Über **TIME** kann der PDO Kanal auf zyklischen Empfang überwacht werden. Wird in **TIME** ein Wert zwischen 1 und 32767 ms eingetragen, dann muss in dieser Zeitspanne eine Botschaft empfangen werden. Anderenfalls geht der FB in den Fehlerzustand (**ERROR** = 1). Über den Wert 0 kann diese Funktion ausgeschaltet werden. Der Überwachungstimer läuft in 5 ms Schritten. Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig. Bei einer negativen Flanke an **ENABLE** werden **DONE**, **ERROR** und **ERRORID** zurückgesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Ausführen	BOOL	<b>NEW</b>	Neues PDO empfangen	BOOL
<b>NUMBER</b>	Messagebox Nummer Wertebereich = 0 bis 19	BYTE	<b>ERROR</b>	Fehler im FB	BOOL
<b>TIME</b>	Watchdog-Funktion Wertebereich = 0 bis 32767 0 = ausgeschaltet 1 bis 32767 = Überwachungszeit	INT	<b>ERRORID</b>	Fehlercode	INT
			<b>WORD1</b>	Empfangsdaten Wort 1	INT
			<b>WORD2</b>	Empfangsdaten Wort 2	INT
			<b>WORD3</b>	Empfangsdaten Wort 3	INT
			<b>WORD4</b>	Empfangsdaten Wort 4	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1800h	Wertebereich Number überschritten				
1804h	Angewählte Box ist nicht korrekt konfiguriert				
1805h	24 V für Bustreiber Fehlen oder Bustreiber ist im State „Bus off“				
1807h	Empfangs Timeout ( Watchdog Funktion)				

**i Information**

**PLC Zykluszeit**

Der PLC Zyklus liegt bei 5 ms, d.h. bei einem Aufruf der Funktion im PLC Programm kann nur alle 5 ms eine CAN Botschaft ausgelesen werden. Werden mehrere Botschaften schnell aufeinander gesendet, können Botschaften überschrieben werden.

**Beispiel in ST:**

```
IF bFirstTime THEN
  (* Geräte in den Status Pre-Operational setzen *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* PDO Konfigurieren *)
  PDOConfig(
    Execute := TRUE,
    (* Messagebox 2 konfigurieren *)
    Number := 2,
    (* CAN Knotennummer setzen *)
    TargetID := 50,
    (* PDO wählen (Standard für PDO1 Statuswort, Istwert1, Istwert2, Istwert3) *)
    PDO := 1,
    (* Länge der Daten festlegen (Standard für PDO1 gleich 8 *)
    Length := 8,
    (* Empfangen *)
    Dir := 0);
  END_IF;

  (* Status und Istwerte auslesen *)
  PDOResceive(Enable := TRUE, Number := 2);
  IF PDOResceive.New THEN
    State := PDOResceive.Word1;
    Sollwert1 := PDOResceive.Word2;
    Sollwert2 := PDOResceive.Word3;
    Sollwert3 := PDOResceive.Word4;
  END_IF
```



3.3.1.5 FB\_PDOSend

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X		X	X	X	X

Mit diesem FB können PDO's auf einem vorher konfigurierten Kanal gesendet werden. Es ist möglich diese einmalig oder zyklisch zu senden. Die zu sendenden Daten werden in **WORD1** bis **WORD4** eingetragen. Ein Senden der PDO's ist unabhängig vom CANopen State des Frequenzumrichters möglich. Über **NUMBER** wird der vorher konfigurierte PDO Kanal ausgewählt. In **WORD1** bis **WORD4** werden die zu sendenden Daten eingetragen. Über **CYCLE** kann zwischen einmaligen Senden (Einstellung=0) oder zyklischen Senden gewählt werden. Über eine positive Flanke an **EXECUTE** wird das PDO abgeschickt. Bei **DONE** = 1 waren alle Eingaben korrekt und das PDO wird gesendet. Bei **ERROR** = 1 gab es ein Problem. Die genaue Ursache ist in **ERRORID** abgelegt. Alle Ausgänge werden mit negativer Flanke an **EXECUTE** zurückgesetzt. Die Zeitbasis der PLC ist 5ms, dies gilt auch für den Eingang **CYCLE**. Es sind nur Sendezyklen mit einem Vielfachen von 5ms realisierbar.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Ausführen	<b>BOOL</b>	<b>DONE</b>	PDO gesendet = 1	<b>BOOL</b>
<b>NUMBER</b>	Messagebox Nummer Wertebereich = 0 bis 19	<b>BYTE</b>	<b>ERROR</b>	Fehler im FB	<b>BOOL</b>
<b>CYCLE</b>	Sendezyklus Wertebereich = 0 bis 255 0 = ausgeschaltet 1 bis 255 = Sendezyklus in ms	<b>BYTE</b>	<b>ERRORID</b>	Fehlercode	<b>INT</b>
<b>WORD1</b>	Sendedaten Wort 1	<b>INT</b>			
<b>WORD2</b>	Sendedaten Wort 2	<b>INT</b>			
<b>WORD3</b>	Sendedaten Wort 3	<b>INT</b>			
<b>WORD4</b>	Sendedaten Wort 4	<b>INT</b>			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1800h	Wertebereich Number überschritten				
1804h	Angewählte Box ist nicht korrekt konfiguriert				
1805h	24 V für Bustreiber Fehlen oder Bustreiber ist im State „Bus off“				

Wenn **DONE** auf 1 geht, dann wurde die zu sendende Botschaft vom CAN Modul übernommen, aber noch nicht gesendet. Das eigentliche Senden läuft parallel im Hintergrund. Sollen jetzt über einen FB mehrere Botschaften direkt hintereinander gesendet werden, dann kann es zu passieren, dass bei dem neuen Aufruf die vorherige Botschaft noch nicht gesendet wurde. Dies kann daran erkannt werden, dass weder das **DONE** noch das **ERROR** Signal nach den **CAL** Aufruf auf 1 gesetzt wurde. Der **CAL** Aufruf kann jetzt einfach so oft wiederholt werden, bis eines der beiden Signale auf 1 geht. Sollen über einen einzigen FB mehrere verschiedene CAN-ID's beschrieben werden, so ist dies über

eine Neukonfiguration des FB's möglich. Diese darf jedoch nicht im selben PLC Zyklus wie das Senden erfolgen. Da sonst die Gefahr besteht, dass die zu sendende Botschaft bei der Konfiguration über den FB\_PDOConfig gelöscht wird.

### Beispiel in ST:

```

IF bFirstTime THEN
  (* Geräte in den Status Pre-Operational setzen *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Configure PDO*)
  PDOConfig(
    Execute := TRUE,
    (*Messagebox 1 konfigurieren*)
    Number := 1,
    (* CAN Knotennummer setzen *)
    TargetID := 50,
    (* PDO wählen (Standard für PDO1 Statuswort, Istwert1, Istwert2, Istwert3) *)
    PDO := 1,
    (*Länge der Daten festlegen (Standard für PDO1 gleich 8 *)
    LENGTH := 8,
    (* Senden *)
    Dir := 1);

  IF not PDOConfig.Done THEN
    RETURN;
  END_IF;

  (* Transmit PDO - Steuerwort Gerät in den Status „Einschalt bereit“ versetzen *)
  PDOSend(Execute := TRUE, Number := 1, Word1 := 1150, Word2 := 0, Word3 := 0, Word4 := 0);
  IF NOT PDOSend.Done THEN
    RETURN;
  END_IF;

  PDOSend(Execute := FALSE);
  bFirstTime := FALSE;
END_IF;

CASE State OF
  0:
    (* Ist der digitale Eingang 1 gesetzt? *)
    IF _5_State_digital_input.0 THEN
      (*Transmit PDO - Steuerwort Gerät in den Status „Einschalt bereit“ versetzen *)
      PDOSend(Execute := TRUE, Number := 1, Word1 := 1150, Word2 := 0, Word3 := 0,
        Word4 := 0);
      State := 10;
      RETURN;
    END_IF;

    (*Ist der digitale Eingang 2 gesetzt? *)
    IF _5_State_digital_input.1 THEN
      (* Transmit PDO - Gerät mit 50% Max. Frequenz freigeben *)
      PDOSend(Execute := TRUE, Number := 1, Word1 := 1151, Word2 := 16#2000, Word3 := 0,
        Word4 := 0);
      State := 10;
      RETURN;
    END_IF;

  10:
    PDOSend;
    IF PDOSend.Done THEN
      PDOSend(Execute := FALSE);
      State := 0;
    END_IF;
END_CASE;

```

### 3.3.2 Elektronisches Getriebe mit Fliegender Säge

Für das *elektronische Getriebe* („winkelsynchroner Gleichlauf“) und die Unterfunktion *Fliegende Säge* gibt es zwei Funktionsblöcke, die eine Steuerung dieser Funktionen erlauben. Weiterhin müssen für einen korrekten Ablauf der beiden Funktionsblöcke im Master- und Slave- Frequenzumrichter diverse Parameter eingestellt werden. Exemplarisch ist dies in der nachfolgenden Tabelle am Beispiel eines SK 540E aufgeführt.

Master FU			Slave FU		
Parameter	Einstellung	Bedeutung	Parameter	Einstellung	Bedeutung
P502[-01]	20	Sollfreq. nach Freq.Rampe	P509	10 *	CANopen Broadcast *
P502[-02]	15	Istpos in Inc. High – Word	P510[-01]	10	CANopen Broadcast
P502[-03]	10	Istpos in Inc. Low – Word	P510[-02]	10	CANopen Broadcast
P503	3	CANopen	P505	0	0,0 Hz
P505	0	0,0 Hz	P515[-02]	P515[-03] <sub>Master</sub>	Broadcast Slave Adresse
P514	5	250 kBaud (min. 100 kBaud)	P546[-01]	4	Frequenzaddition
P515[-03]	P515[-02] <sub>Slave</sub>	Broadcast Master Adresse	P546[-02]	24	Sollpos. Inc. High – Word
			P546[-03]	23	Sollpos. Inc. Low – Word
			P600	1,2	Lageregelung an
			Nur für den FB_Gearing		
			P553[-01]	21	Pos. Sollpos Low Word
			P553[-02]	22	Pos. Sollpos High Word

\* (P509) muss nicht zwingend auf {10} „CANopen Broadcast“ stehen. Dann jedoch ist am Master (P502 [-01]) auf die Einstellung {21} "Istfrequenz ohne Schlupf" zu stellen.

#### Information

#### Istlage - Übertragungsformat

Die Istlage des Masters muss zwingend im Format „Inkrement“ (Inc) übergeben werden.

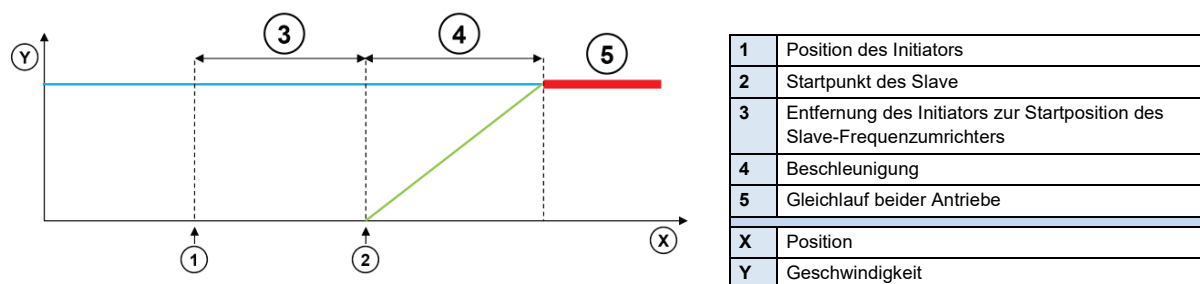
### 3.3.2.1 Überblick

Funktionsbaustein	Erläuterung
FB_Gearing	FB für die einfache Getriebefunktion
FB_FlyingSaw	FB für Getriebefunktion mit fliegender Säge

### 3.3.2.2 FB\_FlyingSaw

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	On+	X	X	X	

Die Funktion Fliegende Säge stellt eine Erweiterung zur Getriebefunktion dar. Mit Hilfe dieser Funktion ist es möglich auf einen fahrenden Antrieb positionsgenau zu synchronisieren. Die Synchronisierung erfolgt im Gegensatz zu FB\_Gearing relativ, d.h. die Slave Achse verfährt synchron zu der Position des Masters, die beim Start der „Fliegenden Säge“ anlag. Der Vorgang der Synchronisierung ist im nachfolgenden Bild dargestellt.



Wird die Funktion gestartet, dann beschleunigt der Slave Frequenzumrichter auf die Geschwindigkeit der Masterachse. Die Beschleunigungsrampe wird über den Weg **ACCELERATION** festgelegt. Bei niedrigen Geschwindigkeit ist die Rampe so flacher und bei hohen Master Geschwindigkeiten ergibt sich eine steiler Rampe für den Slave Frequenzumrichter. Der Beschleunigungsweg wird in Umdrehungen (1000 = 1,000 rev) angegeben, wenn P553 als Sollposition angegeben ist. Wird für P553 Sollposition INC verwendet, dann wird der Beschleunigungsweg in Inkrementen angegeben.

Wird der Initiator mit der in **ACCELERATION** gespeicherten Entfernung vor die Position des Slave Antriebes gesetzt, dann wird der Slave präzise mit der auslösenden Position auf dem Masterantrieb synchronisiert.

Der FB muss über den **ENABLE** Eingang eingeschaltet werden. Der Start der Funktion kann entweder über einen digitalen Eingang (P420[-xx]=64, *Start Fliegende Säge*) oder **EXECUTE** erfolgen. Der Frequenzumrichter beschleunigt dann auf die Geschwindigkeit der Masterachse. Bei Erreichen der Synchronität zur Masterachse wird der **DONE** Ausgang auf 1 geschaltet.

Über den **STOP** Eingang oder die digitale Eingangsfunktion P420[-xx] = 77, *Fliegende Säge anhalten*, erfolgt ein Ausschalten der Getriebefunktion, der Frequenzumrichter bremst auf 0Hz und bleibt stehen. Über den **HOME** Eingang wird der Umrichter veranlasst auf die absolute Position 0 zu fahren. Nach Beendigung des **HOME** oder **STOP** Befehls ist der jeweils zugeordnete Ausgang aktiv. Über eine erneute Betätigung von **EXECUTE** oder den digitalen Eingang kann die Getriebefunktion wieder gestartet werden. Mit der digitalen Eingangsfunktion (P420[-xx] = 63, *Gleichlauf ausschalten*) kann die Getriebefunktion angehalten, und anschließend auf die absolute Position 0 gefahren werden.

Wird die Funktion durch die MC\_Stop Funktion unterbrochen, dann wird **ABORT** auf 1 gesetzt. Im Fehlerfall wird **ERROR** auf 1 und in **ERRORID** der Errorcode gesetzt. Diese drei Ausgänge werden zurückgesetzt wenn **ENABLE** auf 0 geschaltet wird.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>VALID</b>	Vorgegebene Sollfrequenz erreicht	BOOL
<b>EXECUTE</b>	Start der Synchronisierung	BOOL	<b>DONEHOME</b>	Home Fahrt beendet	
<b>STOP</b>	Stop der Synchronisierung	BOOL	<b>DONESTOP</b>	Stop Kommando ausgeführt	
<b>HOME</b>	Verfährt auf Position 0	BOOL	<b>ABORT</b>	Befehl abgebrochen	BOOL
<b>ACCELERATION</b>	Beschleunigungsweg (1rev. = 1.000)	DINT	<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				

### 3.3.2.3 FB\_Gearing

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	On+	X	X	X	

Über den Funktionsbaustein FB\_Gearing kann die Position und die Drehzahl des Frequenzumrichters auf die eines Masterumrichters synchronisiert werden. Der Slave, der diese Funktion verwendet, folgt immer den Bewegungen des Masterumrichters.

Die Synchronisierung erfolgt absolut, d.h. Slave- und Masterposition sind immer gleich.

#### Information

Wird der Slave mit einer anderen Position als der Master in den Getriebemodus geschaltet, dann verfährt der Slave mit max. Frequenz zur Masterposition.

Wird ein Übersetzungsverhältnis angegeben, ergibt sich nach dem Wiedereinschalten auch eine neue Position.

Der Positionswert, auf den synchronisiert wird, sowie die Drehzahl, müssen über den Broadcast Kanal übertragen werden. Über den Eingang **ENABLE** wird die Funktion aktiviert, dabei muss die Lageregelung aktiv und die Endstufe freigegeben sein. Die Endstufe kann z.B. mit der Funktion MC\_Power freigegeben werden. Wird **ENABLE** auf 0 gesetzt, dann bremst der Frequenzumrichter auf 0Hz und bleibt stehen. Der Umrichter befindet sich jetzt wieder im Mode Lageregelung. Wird der MC\_Stop aktiviert, dann verlässt der Frequenzumrichter den Getriebemodus und der **ABORT** Ausgang geht auf 1. Bei Fehlern im FB geht **ERROR** auf 1 und die Fehlerursache steht in **ERRORID**. Über ein Setzen von **ENABLE** auf 0 kann **ERROR**, **ERRORID** und **ABORT** wieder zurückgesetzt werden.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Gleichlauf aktiv	BOOL	<b>VALID</b>	Getriebefunktion ist aktiv	BOOL
<b>RELATIVE</b>	Relative Mode (ab V2.1)	BOOL	<b>ABORT</b>	Befehl abgebrochen	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				
1201h	Der PLC Sollwert Position High ist nicht parametrier				
1202h	Der PLC Sollwert Position Low ist nicht parametrier				

### 3.3.3 Motion Control

Die Motion Control Lib ist an die PLCopen Specification „Function blocks for motion control“ angelehnt. Sie enthält Funktionsblöcke zum Steuern und Verfahren eines Frequenzumrichters und bietet Zugriff auf seine Parameter. Damit die Motion Blöcke funktionieren, müssen einige Einstellungen in den Parametern des Gerätes vorgenommen werden.

Funktionsblock	Benötigte Einstellungen
MC_MoveVelocity	<ul style="list-style-type: none"> <li>• P350 = PLC aktiv</li> <li>• P351 = Hauptsollwert kommt von der PLC</li> <li>• P553 [-xx] = Sollfrequenz</li> <li>• P600 = Lageregelung (Positioniermode) ist ausgeschaltet</li> </ul>
MC_MoveAbsolute	<ul style="list-style-type: none"> <li>• P350 = PLC aktiv</li> <li>• P351 = Hauptsollwert kommt von der PLC</li> </ul>
MC_MoveRelative	<ul style="list-style-type: none"> <li>• P600 = Lageregelung (Positioniermode) ist eingeschaltet</li> </ul>
MC_MoveAdditive	<ul style="list-style-type: none"> <li>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollposition High Word parametrisiert sei</li> <li>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollposition Low Word parametrisiert sei</li> </ul>
MC_Home	<ul style="list-style-type: none"> <li>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollfrequenz parametrisiert sei</li> </ul>
MC_Power	<ul style="list-style-type: none"> <li>• P350 = PLC aktiv</li> </ul>
MC_Reset	<ul style="list-style-type: none"> <li>• P351 = Steuerwert kommt von der PLC</li> </ul>
MC_Stop	

#### Information

Die PLC\_Sollwert 1 bis 5 und das PLC Steuerwort lassen sich auch über Prozessvariablen beschreiben. Sollen jedoch die Motion Control FB's verwendet werden, dürfen keine entsprechenden Prozessvariablen in der Variablen-tabelle deklariert sein, da sonst die Ausgaben der Motion Control FB's überschrieben werden.

#### Information

#### Erkennen einer Signalflanke

Damit die nachfolgenden Funktionsblöcke eine Flanke am Eingang erkennen können, ist es notwendig, dass der Funktionsaufruf zwei Mal mit unterschiedlichen Zuständen am Eingang durchlaufen wird.

Funktionsblock	Erläuterung
MC_ReadParameter	Lesezugriff auf die Parameter des Gerätes
MC_WriteParameter	Schreibzugriff auf die Parameter des Gerätes
MC_MoveVelocity	Verfahrenbefehl im Drehzahlmode
MC_MoveAbsolute	Verfahrenbefehl mit absoluter Positionsangabe
MC_MoveRelative	Verfahrenbefehl mit relativer Positionsangabe
MC_MoveAdditive	Verfahrenbefehl mit additiver Positionsangabe
MC_Home	Startet eine Homefahrt
MC_Power	Ein-/Ausschalten der Motorspannung
MC_ReadStatus	Gerätestatus
MC_ReadActualPos	Liest die aktuelle Position aus
MC_Reset	Fehlerreset im Gerät
MC_Stop	Stoppt alle aktiven Verfahrenbefehle



### 3.3.3.1 MC\_Control

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	

Dieser FB dient zum Steuern des FU und bildet die Möglichkeiten des FU Steuerwortes etwas detaillierter nach wie der MC\_Power. Über die Eingänge **ENABLE (ENABLE\_RIGHT)**, **ENABLE\_LEFT**, **DISABLEVOLTAGE** und **QUICKSTOP** wird der FU gesteuert, siehe nachfolgende Tabelle.

Baustein Eingänge				Verhalten Frequenzumrichter
ENABLE (RIGHT)	ENABLE_LEFT	QUICKSTOP	DISABLE VOLTAGE	
High	Low	Low	Low	Der Frequenzumrichter wird eingeschaltet (Freigabe rechts).
X	High	Low	Low	Der Frequenzumrichter wird eingeschaltet (Freigabe links).
Low	Low	Low	Low	Der Frequenzumrichter bremst auf 0Hz (P103) und schaltet dann den Motor spannungsfrei.
X	X	X	High	Der Frequenzumrichter wird sofort spannungsfrei geschaltet, der Motor dreht ungebremst aus.
X	X	High	Low	Der Frequenzumrichter fährt einen Schnellstop (P426) und schaltet dann den Motor spannungsfrei

Über den Eingang **PARASET** kann der aktive Parametersatz eingestellt werden.

Wenn der Ausgang **STATUS = 1** ist, dann ist der FU eingeschaltet und der Motor wird bestromt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>STATUS</b>	Motor wird bestromt	BOOL
<b>DISABLEVOLTA GE</b>	Spannungsfrei schalten	BOOL	<b>ERROR</b>	Fehler im FB	BOOL
<b>QUICKSTOP</b>	Schnellstop	BOOL	<b>ERRORID</b>	Fehlercode	INT
<b>PARASET</b>	Aktiver Parametersatz Wertebereich: 0 - 3	BYTE			
<b>ENABLE_RIGHT</b>	Freigabe rechts (wie <b>ENABLE</b> ) (SK5xxP)	BOOL			
<b>ENABLE_LEFT</b>	Freigabe links (SK5xxP)	BOOL			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1001h	Stop Funktion ist aktiv				
1300h	Der FU befindet sich in einem Zustand, in dem die ausgewählte Funktion nicht ausgeführt werden kann				

### Beispiel in ST:

```

(* Gerät freigeben mit Dig3*)
Control.Enable := _5_State_digital_input.2;
(* Parametersätze werden über Dig1 und Dig2 festgelegt. *)
Control.ParaSet := INT_TO_BYTE(_5_State_digital_input and 2#11);
Control;
(* Ist Gerät freigegeben? *)
if Control.Status then
  (* Soll eine andere Position angefahren werden? *)
  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;

(* Position anfahren wenn das Gerät freigegeben ist. *)
Move(Execute := Control.Status);

```

3.3.3.2 MC\_Control\_MS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit								X

Dieser FB dient zum Steuern des Starters (MS).

Baustein Eingänge				Verhalten Frequenzumrichter
ENABLE_RIG HT	ENABLE_LEFT	QUICKSTOP	DISABLEVOLTAGE	
High	Low	Low	Low	MS wird eingeschaltet, rechtsdrehend
Low	High	Low	Low	MS wird eingeschaltet, linksdrehend
High	High	Low	Low	MS wird ausgeschaltet
Low	Low	Low	Low	MS bremst auf 0 Hz (P103) und schaltet dann den Motor spannungsfrei
X	X	X	High	MS wird sofort spannungsfrei geschaltet, der Motor dreht ungebremst aus
X	X	High	Low	MS fährt einen Schnellstopp (P426) und schaltet dann den Motor spannungsfrei

(X = der Pegel am Eingang ist unwichtig )

Wenn der Ausgang **STATUS** = 1 ist, dann ist der MS eingeschaltet und der Motor wird bestromt.

Wird **OPENBRAKE** auf 1 gesetzt, dann wird die Bremse geöffnet.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
ENABLE_RIGHT	Freigabe rechts	BOOL	STATUS	Motor wird bestromt	BOOL
ENABLE_LEFT	Freigabe links	BOOL	ERROR	Fehler im FB	BOOL
DISABLEVOLTA GE	Spannungsfrei schalten	BOOL	ERRORID	Fehlercode	INT
QUICKSTOP	Schnellstopp	BOOL			
OPENBRAKE	Bremse öffnen	BOOL			
ERRORID	Erläuterung				
0	Kein Fehler				
1001h	Stopp Funktion ist aktiv				
1300h	MS befindet sich in einem unerwarteten State				

### 3.3.3.3 MC\_Home

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>		X	X		X	X	X	X

Veranlasst den Frequenzumrichter eine Referenzpunktfahrt zu starten, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Der Frequenzumrichter verfährt mit der in **VELOCITY** eingetragenen Sollfrequenz. Wenn der Eingang mit dem Positionsreferenzsignal (P420[-xx] = Referenzpunkt) aktiv wird, dann erfolgt eine Drehrichtungsumkehr. Bei der negativen Flanke des Positionsreferenzsignals wird der in **POSITION** stehende Wert übernommen. Anschließend bremst der Frequenzumrichter auf 0Hz ab, das Signal **DONE** geht auf 1. Während der gesamten **HOME** Fahrt ist der **BUSY** Ausgang aktiv. Wird der Eingang **MODE** auf **True** gesetzt, übernimmt der Antrieb beim Überfahren des Referenzpunktschalters während der Referenzpunktfahrt (positive Flanke → negative Flanke) den Mittelwert beider Positionen und setzt diesen als Referenzpunkt. Der Antrieb reversiert und bleibt auf dem so ermittelten Referenzpunkt stehen. Der Eingang **POSITION** kann nicht verwendet werden.

Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Vorgegebene Sollposition erreicht	BOOL
<b>POSITION</b>	Sollposition	DINT	<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>ERROR</b>	Fehler im FB	BOOL
<b>MODE</b> (ab V2.1)	siehe unten	BOOL	<b>ERRORID</b>	Fehlercode	INT
			<b>BUSY</b>	Home Fahrt aktiv	BOOL
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				
1201h	In den PLC Sollwerten ist die High Position nicht eingetragen (P553)				
1202h	In den PLC Sollwerten ist die Low Position nicht eingetragen (P553)				
1D00h	Absolutwertgeber werden nicht unterstützt				

**3.3.3.4 MC\_Home (SK 5xxP)**

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X			On+				

Veranlasst den Frequenzumrichter eine Referenzpunktfahrt zu starten, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Der Frequenzumrichter verfährt mit der in **VELOCITY** eingetragenen Sollfrequenz. Wenn der Eingang mit dem Positionsreferenzsignal (P420[-xx] = Referenzpunkt) aktiv wird, dann erfolgt eine Drehrichtungsumkehr. Bei der negativen Flanke des Positionsreferenzsignals wird der in **POSITION** stehende Wert übernommen. Anschließend bremst der Frequenzumrichter auf 0Hz ab, das Signal **DONE** geht auf 1. Während der gesamten **HOME** Fahrt ist der **BUSY** Ausgang aktiv.

Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Vorgegebene Sollposition erreicht	BOOL
<b>POSITION</b>	Sollposition	DINT	<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>ERROR</b>	Fehler im FB	BOOL
<b>MODE</b>	siehe unten	BYTE	<b>ERRORID</b>	Fehlercode	INT
			<b>BUSY</b>	Home Fahrt aktiv	BOOL
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				
1201h	In den PLC Sollwerten ist die High Position nicht eingetragen (P553)				
1202h	In den PLC Sollwerten ist die Low Position nicht eingetragen (P553)				
1D00h	Absolutwertgeber werden nicht unterstützt				
1D01h	Wertebereich von Eingang „Mode“ über- bzw. unterschritten (P623)				

**Mode**

Wert	Erläuterung
1..14	Referenzpunktmethode siehe P623
15	<p>Wird der Referenzpunktschalter erreicht, reversiert der Antrieb. Beim Verlassen des Referenzpunktschalters (negative Flanke) wird dies als Referenzpunkt übernommen. Der Referenzpunkt liegt somit typischer Weise auf der Seite des Referenzpunktschalters, auf der die Referenzpunktfahrt begonnen wurde.</p> <p><b>Hinweis:</b> Wird der Referenzpunktschalter „überfahren“ (zu schmaler Schalter, zu hohe Geschwindigkeit), wird ebenfalls beim Verlassen des Referenzpunktschalters (negative Flanke) dies als Referenzpunkt übernommen. Der Referenzpunkt liegt somit nicht auf der Seite des Referenzpunktschalters, auf der die Referenzpunktfahrt begonnen wurde. (P623 = [15] Nord Methode 1)</p>
16	<p>Wie 15, jedoch führt ein Überfahren des Referenzpunktschalters nicht zur Übernahme als Referenzpunkt. Erst nach abgeschlossenem Reversieren führt eine negative Flanke zur Übernahme als Referenzpunkt.</p> <p>Der Referenzpunkt liegt somit sicher auf der Seite des Referenzpunktschalters, auf der die Referenzpunktfahrt begonnen wurde. (P623 = [16] Nord Methode 2)</p>
17	<p>Beim Überfahren des Referenzpunktschalters während der Referenzpunktfahrt (positive Flanke → negative Flanke) übernimmt der Antrieb den Mittelwert beider Positionen und setzt diesen als Referenzpunkt. Der Antrieb reversiert und bleibt auf dem so ermittelten Referenzpunkt stehen. (P623 = [17] Nord Methode 3)</p>
18..34	Referenzpunktmethode siehe P623

3.3.3.5 MC\_MoveAbsolute

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	On+	X	X	X	

Schreibt einen Positions- und Geschwindigkeitssollwert zum Frequenzumrichter, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Die Sollfrequenz **VELOCITY** wird nach der im MC\_MoveVelocity erläuterten Skalierung übergeben.

**POSITION:**

**MODE** = False:

Die Sollposition ergibt sich aus dem in **POSITION** übergebenen Wert.

**MODE** = True:

Der in **POSITION** übergebene Wert entspricht um 1 erhöht dem Index aus Parameter P613. Die in diesem Parameterindex hinterlegte Position entspricht der Sollposition.

Beispiel:

Mode = True; Position = 12

Der FB fährt die Position, die im aktuellen Parametersatz von P613[-13] steht, an.

Hat der Umrichter die Sollposition erreicht, so wird **DONE** auf 1 gesetzt. **DONE** wird mit dem Rücksetzen von **EXECUTE** gelöscht. Wenn **EXECUTE** vor dem Erreichen der Zielposition gelöscht wird, so wird **DONE** für einen Zyklus auf 1 gesetzt. Während des Verfahrens zur Sollposition ist **BUSY** aktiv. Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt. Im Fehlerfall wird **ERROR** auf 1 und in **ERRORID** der entsprechende Fehlercode gesetzt. **DONE** ist in diesem Fall 0. Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Vorgegebene Sollposition erreicht	BOOL
<b>POSITION</b>	Sollposition	DINT	<b>BUSY</b>	Sollposition nicht erreicht	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
<b>MODE</b>	Modus Quelle Sollposition	BOOL	<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
0x1000	FU ist nicht freigegeben				
0x1200	Lageregelung ist nicht aktiviert				
0x1201	In den PLC Sollwerten ist die High Position nicht eingetragen (P553)				
0x1202	In den PLC Sollwerten ist die Low Position nicht eingetragen (P553)				

### Beispiel in ST:

```
(* Das Gerät wird freigegeben, wenn DIG1 = TRUE *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* Das Gerät ist freigegeben und fährt auf Position 20000 mit 50% max. Frequenz.
  Der Motor benötigt für diese Aktion ein Geber und Lageregelung muss aktive sein. *)
  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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	On+	X	X	X	

Entspricht bis auf den Eingang **DISTANCE** in allen Punkten dem MC\_MoveAbsolute. Die Sollposition ergibt sich aus der Addition von aktueller Sollposition und der übergebenen **DISTANCE**.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Vorgegebene Sollposition erreicht	BOOL
<b>DISTANCE</b>	Sollposition	DINT	<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>ERROR</b>	Fehler im FB	BOOL
<b>MODE</b>	Modus Quelle Sollposition	BOOL	<b>ERRORID</b>	Fehlercode	INT
			<b>BUSY</b>	Sollposition nicht erreicht	BOOL
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				
1201h	In den PLC Sollwerten ist die High Position nicht eingetragen (P553)				
1202h	In den PLC Sollwerten ist die Low Position nicht eingetragen (P553)				

### 3.3.3.7 MC\_MoveRelative

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	On+	X	X	X	

Entspricht bis auf den Eingang **DISTANCE** in allen Punkten dem MC\_MoveAbsolute. Die Sollposition ergibt sich aus der Addition von aktueller Istposition und der übergebenen **DISTANCE**.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Vorgegebene Sollposition erreicht	BOOL
<b>DISTANCE</b>	Sollposition	DINT	<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>ERROR</b>	Fehler im FB	BOOL
<b>MODE</b>	Modus Quelle Sollposition	BOOL	<b>ERRORID</b>	Fehlercode	INT
			<b>BUSY</b>	Sollposition nicht erreicht	BOOL
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1200h	Lageregelung ist nicht aktiviert				
1201h	In den PLC Sollwerten ist die High Position nicht eingetragen (P553)				
1202h	In den PLC Sollwerten ist die Low Position nicht eingetragen (P553)				

3.3.3.8 MC\_MoveVelocity

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	

Setzt die Sollfrequenz für den Frequenzumrichter, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Hat der Frequenzumrichter die Sollfrequenz erreicht, so wird **INVELOCITY** auf 1 gesetzt. Während der FU auf die Sollfrequenz beschleunigt, ist der **BUSY** Ausgang aktiv. Wurde **EXECUTE** bereits auf 0 gesetzt, dann wird **INVELOCITY** nur für einen Zyklus auf 1 gesetzt. Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

**VELOCITY** wird skaliert nach folgender Formel eingegeben:

$$\text{VELOCITY} = ( \text{ Sollfrequenz (Hz)} \times 0x4000 ) / P105$$

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>INVELOCITY</b>	Vorgegebene Sollfrequenz erreicht	BOOL
<b>VELOCITY</b>	Sollfrequenz	INT	<b>BUSY</b>	Sollfrequenz noch nicht erreicht	BOOL
			<b>COMMAND-ABORTED</b>	Befehl abgebrochen	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1000h	FU ist nicht freigegeben				
1100h	FU nicht im Drehzahl Mode (Lageregelung aktive)				
1101h	Keine Sollfrequenz parametrier (P553)				

### Beispiel AWL:

```
CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* 20 Hz einstellen (Max. 50 Hz) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute
```

### Beispiel in ST:

```
(* Gerät betriebsbereit wenn DIG1 gesetzt *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* Gerät freigeben mit 50% der max. Frequenz wenn DIG2 gesetzt *)
  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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Über diese Funktion kann die Endstufe des Gerätes ein- oder ausgeschaltet werden. Wird der **ENABLE** Eingang auf 1 gesetzt, dann wird die Endstufe freigegeben. Voraussetzung dafür ist das sich das Gerät im State „Einschaltsperr“ oder „Einschaltbereit“ befindet. Sollte das Gerät im State „Störung“ oder „Störungsreaktion aktiv“ sein, muss zuerst die Störung beseitigt und quittiert werden. Erst dann kann eine Freigabe über diesen Block erfolgen. Befindet sich das Gerät im State „Nicht Einschaltbereit“, ist ein Einschalten auch nicht möglich. In allen Fällen geht der FB in den Fehlerstate und **ENABLE** muss auf 0 gesetzt werden, um den Fehler zu quittieren.

Wird der **ENABLE** Eingang auf 0 gesetzt, dann wird das Gerät ausgeschaltet. Geschieht dies bei laufendem Motor, so wird dieser über die in P103 eingestellte Rampe vorher auf 0 Hz heruntergefahren.

Der Ausgang **STATUS** ist 1 wenn die Endstufe des Gerätes eingeschaltet ist, andernfalls ist er 0.

**ERROR** und **ERRORID** werden zurückgesetzt, wenn **ENABLE** auf 0 geschaltet wird.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>STATUS</b>	Motor wird bestromt	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
<b>ERRORID</b>	Erläuterung				
0	Kein Fehler				
1001h	Stopp Funktion ist aktiv				
1300h	Gerät befindet sich nicht im State „Einschaltbereit“ oder „Einschaltsperr“				

**Beispiel in AWL:**

```

CAL Power
CAL Move

LD TRUE
ST Power.Enable

(* 20 Hz einstellen (Max. 50 Hz) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute

```

**Beispiel in ST:**

```
(* Power Block aktivieren *)  
Power(Enable := TRUE);  
IF Power.Status THEN  
    (* Das Gerät ist einschaltbereit *)  
END_IF
```

### 3.3.3.10 MC\_ReadActualPos

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	On+	X	X	X	

Liefert kontinuierlich die aktuelle Istposition des Frequenzumrichters, wenn **ENABLE** auf 1 steht. Sobald eine gültige Istposition am Ausgang anliegt wird **VALID** auf gültig gesetzt. Im Fehlerfall wird **ERROR** auf 1 gesetzt und **VALID** ist in diesem Fall 0.

Skalierung Position: 1 Motorumdrehung = 1000

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>VALID</b>	Ausgang ist gültig	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>POSITION</b>	Aktuelle Istposition des FU	DINT

#### Beispiel in 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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Liest einen Parameter zyklisch aus dem Gerät, sofern **ENABLE** auf 1 gesetzt ist. Der gelesene Parameter wird in Value abgelegt und ist gültig, wenn **DONE** auf 1 gesetzt ist. Für die Dauer des Lesevorgangs wird der Ausgang **BUSY** auf 1 gesetzt. Bleibt **ENABLE** auf 1 dann wird der Parameter ständig zyklisch ausgelesen. Parameternummer und Index können jederzeit bei aktivem **ENABLE** geändert werden. Jedoch ist schwierig zu erkennen, wann der neue Wert ausgelesen ist, da das **DONE** Signal die gesamte Zeit 1 ist. In diesem Fall ist es empfehlenswert das **ENABLE** Signal für einen Zyklus auf 0 zu setzen, da das **DONE** Signal dann zurückgesetzt wird. Der Parameterindex ergibt sich aus dem Index in der Dokumentation minus 1. So wird z.B. P700 Index 3 („Grund Einschaltsperr“) über den Parameterindex 2 abgefragt. Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0 und die **ERRORID** enthält den Fehlercode. Wird das **ENABLE** Signal auf 0 gesetzt, dann werden alle Signale und die **ERRORID** gelöscht.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>DONE</b>	Value ist gültig	BOOL
<b>PARAMETERNUMBER</b>	Parameternummer	INT	<b>ERROR</b>	Lesevorgang ist fehlgeschlagen	BOOL
<b>PARAMETERINDEX</b>	Parameterindex	INT	<b>BUSY</b>	Der Vorgang ist nicht abgeschlossen	BOOL
			<b>ERRORID</b>	Fehlercode	INT
			<b>VALUE</b>	Ausgelesener Parameter	DINT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	unzulässige Parameternummer				
3	fehlerhafter Parameterindex				
4	kein Array				
201	Ungültiges Auftragsselement im zuletzt empfangenen Auftrag				
202	Interne Antwortkennung nicht abbildbar				

#### Beispiel in ST:

```
(* Motionbaustein 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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Liest den Status des Gerätes aus. Die Statusmaschine orientiert sich an der PLCopen Spezifikation „Function blocks for motion control“. Solange **ENABLE** auf 1 steht wird der Zustand ausgelesen.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>VALID</b>	Ausgang ist gültig	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORSTOP</b>	Das Gerät hat einen Fehler	BOOL
			<b>DISABLED</b>	Die Endstufe des Gerätes ist ausgeschaltet	BOOL
			<b>STOPPING</b>	Ein Stopp Befehl ist aktiv	BOOL
			<b>DISCRETE MOTION</b>	Einer der drei Positionier FB ist aktiv	BOOL
			<b>CONTINUOUS MOTION</b>	Der MC_Velocity ist aktiv	BOOL
			<b>HOMING</b>	Der MC_Home ist aktiv	BOOL
			<b>STANDSTILL</b>	Das Gerät hat keinen aktiven Verfahrbefehl. Es steht mit Drehzahl 0 U/min und eingeschalteter Endstufe.	BOOL

**Beispiel in 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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Rücksetzen eines Fehlers im Gerät (Störungsquittierung), bei einer steigenden Flanke von **EXECUTE**. Im Fehlerfall wird **ERROR** auf 1 gesetzt und die Fehlerursache in **ERRORID** eingetragen. Bei einer negativen Flanke an **EXECUTE** werden alle Fehler zurückgesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Start	BOOL	<b>DONE</b>	Gerätefehler zurückgesetzt	BOOL
			<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
			<b>BUSY</b>	Resetvorgang ist noch aktiv	BOOL
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1001h	Stopp Funktion ist aktiv				
1700h	Ein Fehler – Reset konnte nicht ausgeführt werden, die Ursache für den Fehler liegt noch an				

#### Beispiel in 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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Bei steigender Flanke (0 auf 1) wird das Gerät in den Zustand **STANDINGSTILL** gesetzt. Alle gerade aktiven Motion Funktionen werden abgebrochen. Das Gerät bremst auf 0 Hz ab und schaltet die Endstufe aus. Solange der Stopp Befehl aktiv ist (**EXECUTE** = 1), werden alle anderen Motion FB geblockt. Der **BUSY** Ausgang wird mit der steigenden Flanke an **EXECUTE** aktiv und bleibt dies solange bis eine fallende Flanke an **EXECUTE** erfolgt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Start	BOOL	<b>DONE</b>	Befehl ist ausgeführt	BOOL
			<b>BUSY</b>	Befehl ist aktiv	BOOL

### 3.3.3.15 MC\_WriteParameter\_16 / MC\_WriteParameter\_32

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Schreibt einen 16/32 Bit Parameter in das Gerät, wenn **EXECUTE** von 0 auf 1 wechselt (Flanke). Der Parameter wurde geschrieben, wenn **DONE** auf 1 gesetzt ist. Für die Dauer des Lesevorgangs wird der Ausgang **BUSY** auf 1 gesetzt. Im Fehlerfall wird **ERROR** auf 1 gesetzt und die **ERRORID** enthält den Fehlercode. Die Signale **DONE**, **ERROR**, **ERRORID** bleiben solange gesetzt, bis **EXECUTE** wieder auf 0 wechselt. Wechselt das **EXECUTE** Signal auf 0, dann wird der Schreibprozess nicht abgebrochen. Nur das **DONE** Signal bleibt nur für 1 PLC Zyklus gesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Freigabe	BOOL	<b>DONE</b>	Value ist gültig	BOOL
<b>PARAMETERNUMBER</b>	Parameternummer	INT	<b>BUSY</b>	Der Schreibvorgang ist aktiv	BOOL
<b>PARAMETERINDEX</b>	Parameterindex	INT	<b>ERROR</b>	Lesevorgang ist fehlgeschlagen	BOOL
<b>VALUE</b>	Zu schreibender Wert	INT	<b>ERRORID</b>	Fehlercode	INT
<b>RAMONLY</b>	Speichere den Wert nur im RAM (ab Version V2.1)	BOOL			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	unzulässige Parameternummer				
1	Parameterwert nicht änderbar				
2	untere oder obere Wertgrenze überschritten				
3	fehlerhafter Parameterindex				
4	kein Array				
5	Unzulässiger Datentyp				
6	Nur Rücksetzbar (es darf nur 0 geschrieben werden)				
7	Beschreibungselement nicht änderbar				
201	Ungültiges Auftragselement im zuletzt empfangenen Auftrag				
202	Interne Antwortkennung nicht abbildbar				

#### Beispiel in ST:

```

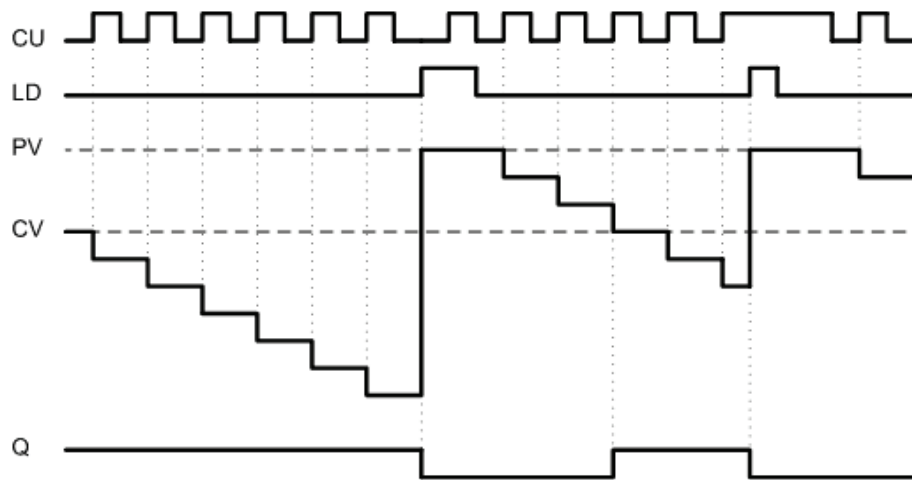
WriteParam16(Execute := TRUE, ParameterNumber := 102, ParameterIndex := 0, Value := 300);
IF WriteParam16.Done THEN
  WriteParam16(Execute := FALSE);
END_IF;
  
```

### 3.3.4 Standard

#### 3.3.4.1 CTD Abwärtszähler

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Bei steigender Flanke an **CD** wird der Zähler des Funktionsblockes **CV** um eins verringert, solange CV größer als -32768 ist. Wenn **CV** kleiner oder gleich 0 ist, bleibt der Ausgang **Q** auf TRUE. Über **LD** kann der Zähler **CV** auf den in **PV** gespeicherten Wert gesetzt werden.



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
CD	Zählereingang	BOOL	Q	TRUE, wenn CV ≤ 0	BOOL
LD	Lade Startwert	BOOL	CV	Aktueller Zählerstand	INT
PV	Startwert	INT			

#### Beispiel in 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
```

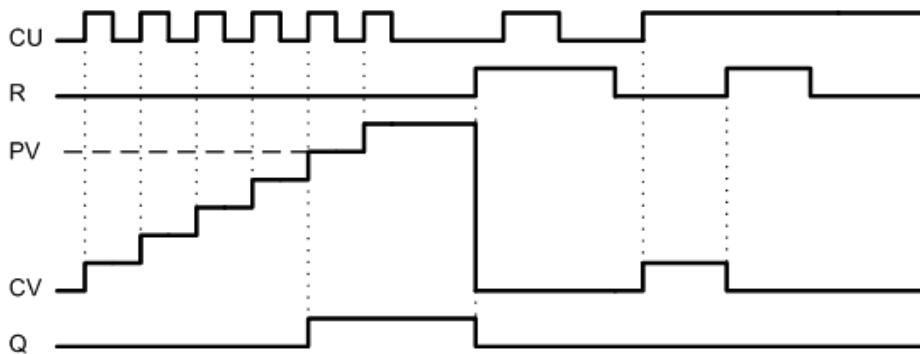
#### Beispiel in ST:

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

### 3.3.4.2 CTU Aufwärtszähler

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Bei steigender Flanke an **CU** wird der Zähler des Funktionsblockes **CV** um eins erhöht. **CV** kann bis auf den Wert 32767 gezählt werden. Solange **CV** größer oder gleich **PV** ist, bleibt der Ausgang **Q** auf TRUE. Über **R** kann der Zähler **CV** auf den Wert null zurückgesetzt werden.



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
CU	Zählereingang	BOOL	Q	TRUE, wenn CV >= PV	BOOL
R	Reset Zählerstand	BOOL	CV	Aktueller Zählerstand	INT
PV	Grenzwert	INT			

#### Beispiel in 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
```

#### Beispiel in ST:

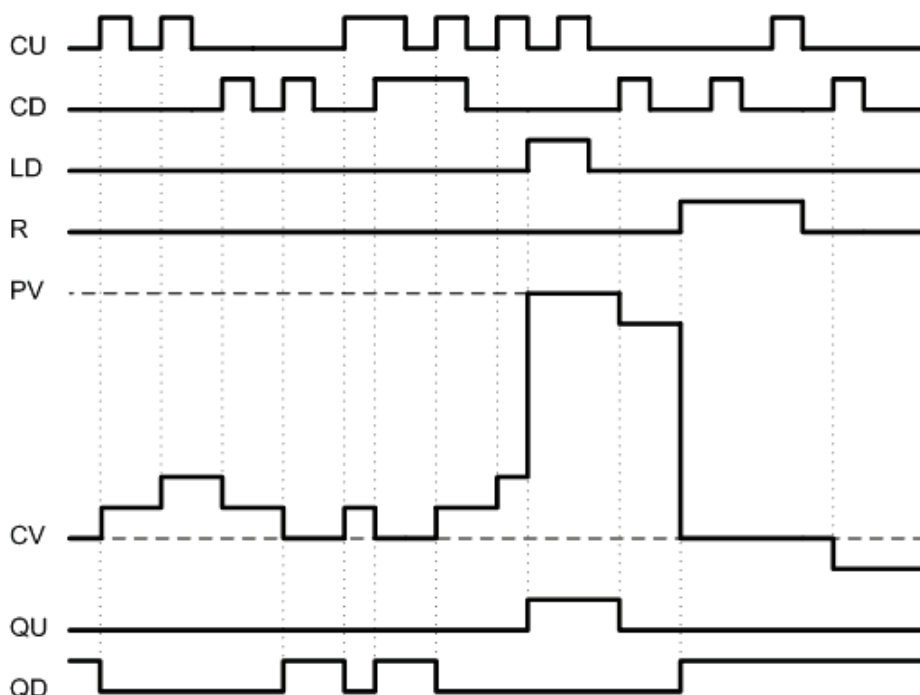
```
CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTUInst.Q;
VarINT2 := CTUInst.CV;
```

### 3.3.4.3 CTUD Auf- und Abwärtszähler

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Bei steigender Flanke an **CU** wird der Zähler **CV** um eins erhöht, solange **CV** kleiner als 32767 ist. Bei steigender Flanke an **CD** wird der Zähler **CV** um eins verringert, solange **CV** größer als -32768 ist. Über **R** kann der Zähler **CV** auf den Wert Null gesetzt werden. Über **LD** wird der in **PV** gespeicherte Wert in **CV** kopiert.

**R** hat Vorrang gegenüber **LD**, **CU** und **CV**. **PV** kann jederzeit verändert werden, **QU** bezieht sich immer auf den aktuell eingestellten Wert.



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>CU</b>	Aufwärtszählen	BOOL	<b>QU</b>	TRUE, wenn $CV \geq PV$	BOOL
<b>CD</b>	Abwärtszählen	BOOL	<b>QD</b>	TRUE, wenn $CV \leq 0$	BOOL
<b>R</b>	Reset Zählerstand	BOOL	<b>CV</b>	Aktueller Zählerstand	INT
<b>LD</b>	Lade Startwert	BOOL			
<b>PV</b>	Startwert / Grenzwert	INT			

**Beispiel in 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
```

**Beispiel in ST:**

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

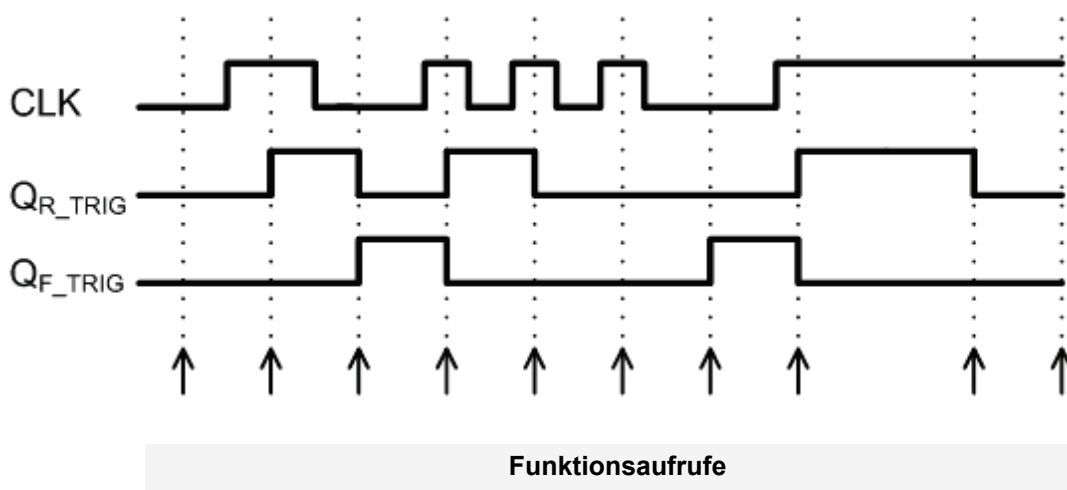


### 3.3.4.4 R\_TRIG und F\_TRIG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Beide Funktionen dienen der Flankenerkennung. Wird eine Flanke auf **CLK** erkannt geht **Q** bis zum nächsten Funktionsaufruf auf TRUE, danach wieder auf FALSE. Erst mit einer neuen Flanke kann **Q** wieder für einen Zyklus TRUE werden.

- R\_TRIG = steigende Flanke
- F\_TRIG = fallende Flanke



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
CLK	Setzen	BOOL	Q	Ausgang	BOOL

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### Information

Die Ausgabe der Funktion ändert sich nur, wenn die Funktion aufgerufen wird. Aus diesem Grund ist es ratsam, die Flankendetektion kontinuierlich mit dem SPS-Zyklus aufzurufen.

### 3.3.4.5 RS Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Bistabile Funktion, über **S** wird der Ausgang **Q1** gesetzt und über **R1** wieder gelöscht. Liegt an **R1** und **S** zeitgleich ein TRUE an, so ist **R1** dominant.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>S</b>	Setzen	BOOL	<b>Q1</b>	Ausgang	BOOL
<b>R1</b>	Reset	BOOL			

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.3.4.6 SR Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Bistabile Funktion, über **S1** wird der Ausgang **Q1** gesetzt und über **R** wieder gelöscht. Liegt an **R1** und **S** zeitgleich ein TRUE an, so ist **S1** dominant.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>S1</b>	Setzen	BOOL	<b>Q1</b>	Ausgang	BOOL
<b>R</b>	Reset	BOOL			

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

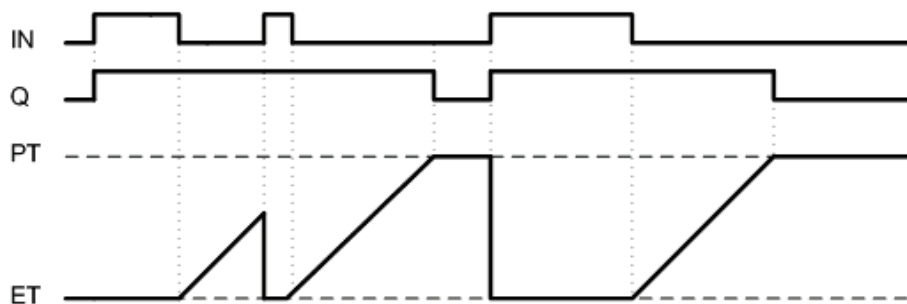
### 3.3.4.7 TOF Ausschaltverzögerung

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Wird **IN** = TRUE, dann wird **Q** auf TRUE gesetzt. Geht **IN** auf FALSE, läuft der Timer hoch. Solange der Timer läuft (**ET** < **PT**) bleibt **Q** auf TRUE gesetzt. Ist (**ET** = **PT**) bleibt der Timer stehen, **Q** wird dann FALSE. Bei einer neuen steigenden Flanke auf **IN**, wird der Timer **ET** wieder auf null gesetzt.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
IN	Timer aktiv	BOOL	Q	TRUE ß ( ET < PT )	BOOL
PT	Zeitdauer	DINT	ET	Aktueller Stand des Timers	DINT

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### Information

#### Timer ET

Die Zeit ET läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit IN und das Setzen des Ausgangs Q werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5 ms sein, sodass zeitlich ein Jitter entstehen kann.

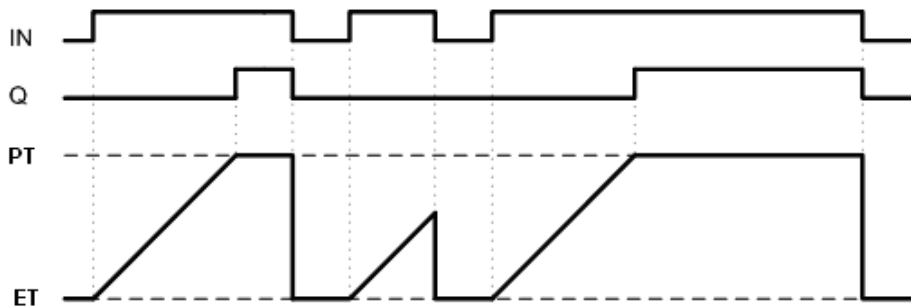
### 3.3.4.8 TON Einschaltverzögerung

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Wird **IN** = TRUE gesetzt, dann läuft der Timer hoch. Wenn **ET** = **PT** ist, wird **Q** auf TRUE gesetzt und der Timer bleibt stehen. **Q** bleibt solange TRUE wie **IN** auch TRUE ist. Bei einer neuen steigenden Flanke auf **IN** fängt der Timer wieder bei null an zu laufen. **PT** kann verändert werden während der Timer läuft. Die Zeitdauer wird in **PT** in Millisekunden eingegeben. Damit ist eine Zeitverzögerung zwischen 5ms und 24,8 Tagen möglich. Da die Zeitbasis der PLC bei 5ms liegt, ist die minimale Zeitverzögerung auch 5ms.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
IN	Timer aktiv	BOOL	Q	TRUE & ( IN=TRUE & ET=PT )	BOOL
PT	Zeitdauer	DINT	ET	Aktueller Stand des Timers	DINT

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### Information

### Timer ET

Die Zeit ET läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit IN und das Setzen des Ausgangs Q werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5 ms sein, sodass zeitlich ein Jitter entstehen kann.

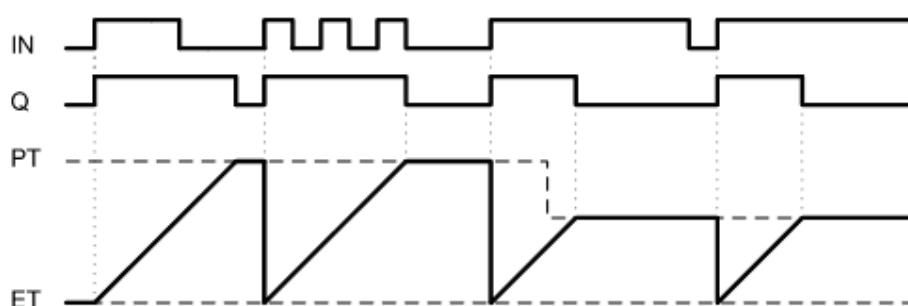
### 3.3.4.9 TP Zeitimpuls

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Bei einer positiven Flanke an **IN** wird der Timer mit dem Wert 0 gestartet. Der Timer zählt bis auf den in **PT** eingetragenen Wert hoch und bleibt dann stehen. Dieser Vorgang ist nicht unterbrechbar! **PT** kann während des Hochzählens verändert werden. Der Ausgang **Q** ist TRUE, solange der Timer **ET** kleiner als **PT** ist. Wenn **ET = PT** ist und eine steigende Flanke an **IN** erkannt wird, wird der Timer wieder bei 0 gestartet.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
IN	Timer aktiv	BOOL	Q	TRUE & ( ET < PT )	BOOL
PT	Zeitdauer	DINT	ET	Aktueller Stand des Timers	DINT

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### Information

#### Timer ET

Die Zeit ET läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit IN und das Setzen des Ausgangs Q werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5 ms sein, sodass zeitlich ein Jitter entstehen kann.



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Ausführen	BOOL	<b>VALID</b>	Lesevorgang erfolgreich	BOOL
<b>SIZE</b>	Speicherformat	BOOL	<b>READY</b>	Der gesamte Speicher ist ausgelesen	BOOL
<b>MEMORY</b>	Auswahl Speicherbereich	BYTE	<b>ERROR</b>	der FB hat einen Fehler	BOOL
<b>STARTINDEX</b>	Zeigt auf die zu beschreibende Speicherzelle	INT	<b>ERRORID</b>	Fehlercode	INT
			<b>ACTINDEX</b>	Aktueller Speicherindex, aus dem im nächsten Zyklus gelesen wird	INT
			<b>VALUE</b>	Ausgelesener Wert	DINT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1A00h	Wertebereich STARTINDEX wurde überschritten				
1A01h	Wertebereich MEMORY wurde überschritten				



**3.3.5.2 FB\_WriteTrace**

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X		

Über diesen FB können einzelne oder auch größere Mengen an Werten im FU zwischengespeichert werden. Das Speichern der Werte erfolgt nicht dauerhaft, d.h. nach einem Neustart des FU gehen die Werte verloren.

Wird vom FB eine positive Flanke am **ENABLE** Eingang erkannt, dann werden alle am Eingang anliegenden Parameter übernommen. Der in **VALUE** stehende Wert wird auf die durch **STARTINDEX** und **MEMORY** gekennzeichnete Speicherstelle geschrieben. Bei einem erfolgreichen Schreibvorgang geht der Ausgang **VALID** auf 1.

Wird der FB jetzt mehrfach aufgerufen und der **ENABLE** Eingang bleibt auf 1, dann wird bei jedem FB Aufruf der Eingang **VALUE** gelesen und gespeichert, sowie die Speicheradresse um 1 erhöht. Der aktuelle Speicherindex für den nächsten Zugriff kann unter dem Ausgang **ACTINDEX** ausgelesen werden. Wird das Speicherende erreicht, dann geht der Ausgang **FULL** auf 1 und der Speichervorgang wird gestoppt. Ist jedoch der Eingang **OVERWRITE** auf 1 gesetzt ist, so wird der Speicherindex wieder auf den **STARTINDEX** gesetzt und es werden die vorher gespeicherten Werte überschrieben.

Es können Werte im INT oder DINT Format gespeichert werden. Bei INT Werten, wird vom Eingang **VALUE** nur der Low Teil ausgewertet. Die Zuordnung erfolgt über den Eingang **SIZE**, eine 0 steht für INT und eine 1 für DINT Werte.

Die Zuordnung der Speicherbereiche erfolgt über den Eingang **MEMORY**:

**MEMORY** = 1 à P613[0-251]    entspricht 504 INT oder 252 DINT Werten

**MEMORY** = 0 à P900[0-247] bis P906[0-111]                            entspricht 3200 INT oder 1600 DINT Werten

Der FB kann nicht durch andere Blöcke unterbrochen werden.

Mit einer negativen Flanke an **ENABLE** werden alle Ausgänge auf 0 gesetzt und die Funktion des FB beendet.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Ausführen	BOOL	<b>VALID</b>	Schreibvorgang erfolgreich	BOOL
<b>SIZE</b>	Speicherformat	BOOL	<b>FULL</b>	Komplette Speicher ist voll	BOOL
<b>OVERWRITE</b>	Speicher überschreibbar	BOOL	<b>ERROR</b>	der FB hat einen Fehler	BOOL
<b>MEMORY</b>	Auswahl Speicherbereich	BYTE	<b>ERRORID</b>	Fehlercode	INT
<b>STARTINDEX</b>	Zeigt auf die zu beschreibende Speicherzelle	INT	<b>ACTINDEX</b>	Aktueller Speicherindex, auf dem im nächsten Zyklus gespeichert wird	DINT
<b>VALUE</b>	Zu speichernder Wert	DINT			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1A00h	Wertebereich STARTINDEX wurde überschritten				
1A01h	Wertebereich MEMORY wurde überschritten				


### Information

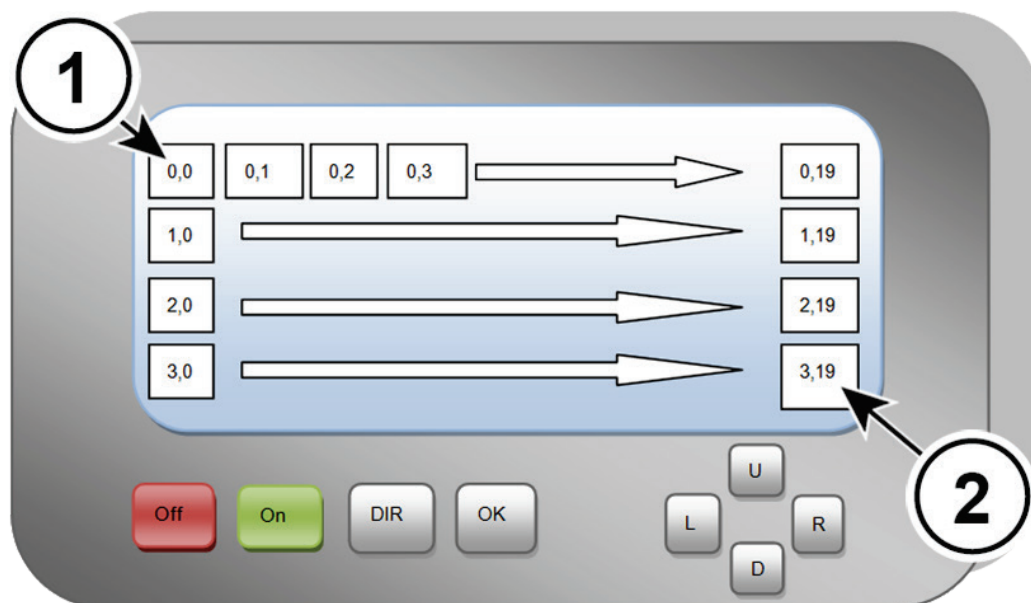
**Beachte!** Der Speicherbereich in der Einstellung MEMORY = 0 wird auch von der Scope Funktion genutzt. Ein Verwenden der Scope Funktion überschreibt die gespeicherten Werte!

### 3.3.6 Visualisierung ParameterBox

In der ParameterBox kann der komplette Displayinhalt für eigene Informationsdarstellungen benutzt werden. Dazu muss die ParameterBox in den Visualisierungsmodus geschaltet werden. Dies ist ab der Firmwareversion V4.3 der ParameterBox (Parameter P1308) möglich und geschieht wie folgt:

- Im Menüpunkt „Anzeige“ den Parameter P1003 auf „PLC-Anzeige“ einstellen
- Über die rechte oder linke Pfeiltaste auf die Betriebswertanzeige wechseln
- PLC Anzeige ist jetzt in der P-Box aktiv und bleibt dies auch dauerhaft

Im Visualisierungsmodus der P-Box kann über die zwei nachfolgend erläuterten FB der Displayinhalt beschrieben werden. Vorab muss jedoch im PLC Konfigurationsdialog (Schaltfläche ) der Punkt „Parameterbox Funktionsbausteine zulassen“ aktiviert sein. Über den Prozesswert „Parameterbox\_key\_state“ kann zusätzlich der Tastaturzustand der Box abgefragt werden. Damit können Eingaben in das PLC Programm realisiert werden. Der nachfolgenden Abbildung kann der Displayaufbau und die Position der auszulesenden Tasten für die ParameterBox entnommen werden.



1	Erstes Zeichen	(0,0 → Zeile = 0 , Spalte = 0)
2	Letztes Zeichen	(3,19 → Zeile = 3 , Spalte = 19)

#### 3.3.6.1 Überblick Visualisierung

Funktionsbaustein	Erläuterung
FB_STRINGToPBox	Kopiert einen String in die P-Box
FB_DINTToPBox	Kopiert einen DINT Wert zur P-Box

### 3.3.6.2 FB\_DINTToPBOX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

Dieser Funktionsbaustein konvertiert einen DINT Wert in einen ASCII String und kopiert diesen in die ParameterBox. Die Ausgabe kann im dezimalen, binären oder hexadezimalen Format erfolgen, die Selektion wird über **MODE** durchgeführt. Über **ROW** und **COLUMN** wird die Startposition des Strings im P-Box Display gesetzt. Der Parameter **LENGTH** übergibt die Länge des Strings in Zeichen. Im **MODE** Dezimal positioniert der Parameter **POINT** ein Komma in die darzustellende Zahl. In **POINT** wird angegeben wie viele Zeichen rechts vom Komma stehen. Bei der Einstellung 0 ist die Funktion **POINT** ausgeschaltet. Sollte die Zahl mehr Zeichen enthalten als es die Länge zulässt und ist außerdem kein Komma gesetzt, so wird der Überlauf durch das Zeichen „#“ angezeigt. Befindet sich ein Komma in der Zahl, so können bei Bedarf alle Zahlen hinter dem Komma entfallen. Im **MODE** hexadezimal und binär werden immer die niederwertigsten Bits dargestellt, wenn die eingestellte Länge zu kurz ist. Solange **ENABLE** auf 1 gesetzt ist, werden alle Änderungen an den Eingängen sofort übernommen. Geht **VALID** auf 1, dann ist der String korrekt übertragen worden. Im Fehlerfall wird **ERROR** auf 1 gesetzt. **VALID** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig. Bei einer negativen Flanke an **ENABLE** werden **VALID**, **ERROR** und **ERRORID** zurückgesetzt.

#### Beispiele:

Einstellung	Darzustellende Zahl	P-Box Anzeige
Length = 5	12345	12345
Point = 0		
Length = 5	-12345	#####
Point = 0		
Length = 10	123456789	123456,789
Point = 3		
Length = 8	123456789	123456,7
Point = 3		

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Übergabe des Strings	BOOL	<b>VALID</b>	String übergeben	BOOL
<b>MODE</b>	Darstellungsformat 0 = Dezimal 1 = Binäre 2 = Hexadezimal Wertebereich = 0 bis 2	BYTE	<b>ERROR</b>	Fehler im FB	BOOL
<b>ROW</b>	Zeile des Display Wertebereich = 0 bis 3	BYTE	<b>ERRORID</b>	Fehlercode	INT
<b>COLUMN</b>	Spalte des Display Wertebereich = 0 bis 19	BYTE			
<b>POINT</b>	Position des Komma Wertebereich = 0 bis 10 0 = Funktion ist ausgeschaltet	BYTE			
<b>LENGTH</b>	Ausgabelänge Wertebereich = 1 bis 11	BYTE			
<b>VALUE</b>	Auszugebende Zahl	DINT			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1500h	String überschreibt den Speicherbereich des P-Box Arrays				
1501h	beim Eingang LINE wurde der Wertebereich überschritten				
1502h	beim Eingang ROW wurde der Wertebereich überschritten				
1504h	beim Eingang POINT wurde der Wertebereich überschritten				
1505h	beim Eingang LENGTH wurde der Wertebereich überschritten				
1506h	beim Eingang MODE wurde der Wertebereich überschritten				

**Beispiel in ST:**

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

(* Aktuelle Position abfragen *)
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;

(* Gerät über DIG1 ein oder ausschalten *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Ist das Gerät eingeschaltet? *)
  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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Dieser Funktionsbaustein kopiert einen String (Zeichenkette) in das P-Box Speicherarray. Über **ROW** und **COLUMN** wird die Startposition des Strings im P-Box Display gesetzt. Der Parameter **TEXT** übergibt den gewünschten String an den Funktionsbaustein, der Stringname kann aus der Variablen-tabelle entnommen werden. Solange **ENABLE** auf 1 ist, werden alle Änderungen an den Eingängen sofort übernommen. Beim gesetzten **CLEAR** Eingang wird der gesamte Display Inhalt mit Leerzeichen überschrieben, bevor der selektierte String geschrieben wird. Geht **VALID** auf 1, dann ist der String korrekt übertragen worden. Im Fehlerfall wird **ERROR** auf 1 gesetzt. **VALID** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig. Bei einer negativen Flanke an **ENABLE** werden **VALID**, **ERROR** und **ERRORID** zurückgesetzt.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Übergabe des String	BOOL	<b>VALID</b>	String übergeben	BOOL
<b>CLEAR</b>	Display löschen	BOOL	<b>ERROR</b>	Fehler im FB	BOOL
<b>ROW</b>	Zeile des Display Wertebereich = 0 bis 3	BYTE	<b>ERRORID</b>	Fehlercode	INT
<b>COLUMN</b>	Spalte des Display Wertebereich = 0 bis 19	BYTE			
<b>TEXT</b>	anzuweisender Text	STRING			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1500h	String überschreibt den Speicherbereich des P-Box Arrays				
1501h	beim Eingang ROW wurde der Wertebereich überschritten				
1502h	beim Eingang COLUMN wurde der Wertebereich überschritten				
1503h	Die gewählte String Nummer existiert nicht				
1506h	In der PLC Konfiguration ist die Option „Parameterbox Funktionsbausteine zulassen“ nicht aktiviert.				

### Beispiel in ST:

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

(* Aktuelle Position abfragen *)
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;

(* Gerät über DIG1 ein oder ausschalten *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Ist das Gerät eingeschaltet? *)
  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 (Erfassen schneller Ereignisse)

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

Die Zykluszeit der PLC beträgt 5ms, dieser Zyklus ist zur Erfassung sehr schneller externer Ereignisse mitunter zu groß. Über der FB Capture ist es möglich auf Flanken an den FU Eingängen bestimmte physikalische Größen zu erfassen. Die Überwachung der Eingänge erfolgt in einem 1ms Zyklus. Die so gespeicherten Werte können später von der PLC ausgelesen werden.

Bei einer positiven Flanke an **EXECUTE** werden alle Eingänge eingelesen und die Capture Funktion scharf geschaltet. Über den Eingang **INPUT** wird der zu überwachende FU Eingang selektiert. Über **EDGE** werden die Art der Flanke und das Verhalten des Bausteins ausgewählt.

**EDGE = 0** Mit der ersten positiven Flanke wird der selektierte Wert unter **OUTPUT1** gespeichert und **DONE1** auf 1 gesetzt. Die nächste positive Flanke speichert unter **OUTPUT2** und **DONE2** wird auf 1 gesetzt. Der FB wird dann deaktiviert.

**EDGE = 1** Verhalten wie unter **EDGE = 0**, mit dem Unterschied das die negative Flanke auslöst.

**EDGE = 2** Mit der ersten positiven Flanke wird der selektierte Wert unter **OUTPUT1** gespeichert und **DONE1** auf 1 gesetzt. Die nächste negative Flanke speichert unter **OUTPUT2** und **DONE2** wird auf 1 gesetzt. Der FB wird dann deaktiviert.

**EDGE = 3** Verhalten wie unter **EDGE = 2**, mit dem Unterschied das zuerst die negative und dann die positive Flanke auslöst.

Wird der Eingang **CONTINUOUS** auf 1 gesetzt, dann ist für **EDGE** nur noch die Einstellung 0 und 1 relevant. Der FB läuft kontinuierlich weiter und speichert das letzte auslösende Ereignis immer unter **OUTPUT1** ab. **DONE1** bleibt ab dem ersten Ereignis aktiv. **DONE2** und **OUTPUT2** werden nicht verwendet.

Der **BUSY** Ausgang bleibt solange aktiv bis beide Capture Ereignisse (**DONE1** und **DONE2**) eingetreten sind.

Die Funktion des Bausteins kann jederzeit durch eine negative Flanke an **EXECUTE** beendet werden. Alle Ausgänge behalten dabei ihre Werte. Mit einer positiven Flanke an **EXECUTE** werden zuerst alle Ausgänge gelöscht und dann die Funktion des Bausteins gestartet.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Ausführen	BOOL	<b>DONE1</b>	Wert in OUTPUT1 gültig	BOOL
<b>CONTINUOUS</b>	Einmalige Ausführung o. Dauerbetrieb	BOOL	<b>DONE2</b>	Wert in OUTPUT2 gültig	BOOL
<b>INPUT</b>	<b>SK54xE</b> Zu überwachender Eingang 0 = Eingang 1 ---- 7 = Eingang 8  <b>SK52xE, SK53xE, SK2xxE, SK2xx-EFDS</b> Zu überwachender Eingang 0 = Eingang 1 ---- 3 = Eingang 4	BYTE	<b>BUSY</b>	FB wartet noch auf Capture Ereignisse	BOOL
<b>EDGE</b>	Auslösende Flanke	BYTE	<b>ERROR</b>	der FB hat einen Fehler	BOOL
<b>SOURCE</b>	Zu speichernde Größe 0 = Position in Umdrehungen 1 = Istfrequenz 2 = Moment	BYTE	<b>ERRORID</b>	Fehlercode	INT
			<b>OUTPUT1</b>	Wert für 1. Capture Ereignisses	DINT
			<b>OUTPUT2</b>	Wert für 2. Capture Ereignisses	DINT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1900h	Wertebereich INPUT wurde überschritten				
1901h	Wertebereich EDGE wurde überschritten				
1902h	Wertebereich SOURCE wurde überschritten				
1903h	Es sind mehr als zwei Instanzen aktiv				

### Beispiel in ST:

```

Power(ENABLE := TRUE);
IF Power.STATUS THEN
  Move(EXECUTE := TRUE, POSITION := Pos, VELOCITY := 16#2000);
  (* Der Capture FB wartet am DIG1 auf ein High Signal. Wird das
    erkannt, speichert der FB die aktuelle Position. Mit Hilfe
    der Eigenschaft "OUTPUT1" kann der Wert abgefragt werden. *)
  Capture(EXECUTE := TRUE, INPUT := 0);

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

```

---

** Information**

Von diesem FB können mehrere Instanzen im PLC Programm existieren. Aber es dürfen zur selben Zeit nur zwei Instanzen aktiv sein!

---

### 3.3.8 FB\_DinCounter

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	ab V1.1	

Dieser FB dient zum Zählen von Impulsen über die Digitaleingänge. Es werden alle Flanken (Low – High und High – Low) gezählt. Die minimale Impulsbreite ist 1 ms.

Der FB wird über ENABLE aktiviert. Mit der positiven Flanke werden die Eingänge PV, UD, DIN und MODE übernommen und alle Ausgänge gelöscht.

**UD** definiert die Zählrichtung

- 0 = größer Zählen
- 1 = kleiner Zählen

In PV kann ein Zählerwert eingetragen werden. Je nach setzen des MODE Eingangs wirkt sich dies verschieden aus.

**MODE**

- 0 = Überlauf, der Zähler wird als Dauerzähler betrieben. Er kann in positiver und negativer Richtung überlaufen. Beim Start der Funktion wird CV = PV gesetzt. In diesem Mode bleibt BUSY immer 1 und Q immer 0.
- 1 = ohne Überlauf
  - Vorwärtszählen à CV startet bei 0, BUSY = 1, und läuft bis CV=>PV. Dann geht BUSY auf 0 und Q auf 1. Der Zählvorgang stoppt.
  - Rückwärtszählen à CV startet mit PV und läuft bis CV<=0. Während dieser Zeit ist BUSY = 1 und geht auf 0 wenn das Zählende erreicht ist. Im Gegenzug geht Q auf 1.
  - Neustart des Zählers wird über einen erneute Flanke am ENABLE Eingang erreicht

**DIN** definiert den Messeingang. Die Anzahl der Eingänge hängt vom jeweiligen FU ab (max. 4).

- Eingang 1 = 0
- Eingang 2 = 1
- Eingang 3 = 2
- Eingang 4 = 3

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Freigabe	BOOL	<b>Q</b>	Zählung beendet	BOOL
<b>UD</b>	Zählrichtung  0 = größer Zählen 1 = kleiner Zählen	BOOL	<b>BUSY</b>	Zähler läuft	BOOL
<b>PV</b>	Zählerwert	INT	<b>ERROR</b>	der FB hat einen Fehler	BOOL
<b>MODE</b>	Modus	BYTE	<b>ERRORID</b>	Fehlercode	INT
<b>DIN</b>	Messeingang	BYTE	<b>CV</b>	Zählerwert	INT
			<b>CF</b>	Zählfrequenz (Auflösung 0,1) <sup>1)</sup>	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
0x1E00	Digitaler Eingang wird bereits vom anderen Zähler verwendet				
0x1E01	Digitaler Eingang existiert nicht				
0x1E02	Wertebereich MODE überschritten				

1) Messbereich 0,1 Hz bis 1 kHz

### 3.3.9 FB\_FunctionCurve

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	

Der Funktionsbaustein stellt eine Kennfeldsteuerung dar. Es können dem Funktionsblock definierte Punkte übergeben werden, durch die er eine Funktion emuliert. Der Ausgang verhält sich dann entsprechend des hinterlegten Kennfeldes. Zwischen den einzelnen Stützpunkten wird linear interpoliert. Die Stützstellen werden mit X und Y-Werten definiert. Die X-Werte sind dabei immer vom Typ **INT**, die Y-Werte können alle entweder vom Typ **INT** oder **DINT** sein, je nach Größe der größten Stützstelle. Wird **DINT** verwendet verbraucht dies auch mehr Speicherplatz. Die Stützstellen werden im Variablenfenster in der Spalte „Init-Wert“ eingetragen. Wird am Eingang **ENABLE** ein TRUE erkannt wurde, wird anhand des Eingangswerts **INVALUE** der entsprechende Ausgangswert **OUTVALUE** berechnet. **VALID** signalisiert mit einem TRUE, dass der Ausgangswert **OUTVALUE** gültig ist. Solange **VALID** FALSE ist, hat der Ausgang **OUTVALUE** den Wert 0. Überschreitet der Eingangswert **INVALUE** das obere oder untere Ende des Kennfeldes, bleibt der erste oder letzte Ausgangswert des Kennfeldes am Ausgang stehen, solange bis sich **INVALUE** wieder im Bereich des Kennfeldes befindet. Bei Über- oder Unterschreitung des Kennfeldes wird der entsprechende Ausgang **MINLIMIT** oder **MAXLIMIT** auf TRUE gesetzt. **ERROR** wird TRUE, wenn die Abszissenwerte (X-Werte) des Kennfeldes nicht fortlaufen größer werden, oder keine Tabelle initialisiert wird. Dabei wird der entsprechende Fehler auch über **ERRORID** ausgegeben und der Ausgangswert wird 0. Der Fehler wird zurückgesetzt, wenn **ENABLE** = FALSE wird.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Ausführen	BOOL	<b>VALID</b>	Ausgangswert ist gültig	BOOL
<b>INVALUE</b>	Eingangswert ( x )	INT	<b>ERROR</b>	Fehler im FB	BOOL
			<b>ERRORID</b>	Fehlercode	INT
			<b>MAXLIMIT</b>	Maximales Limit erreicht	BOOL
			<b>MINLIMIT</b>	Minimales Limit erreicht	BOOL
			<b>OUTVALUE</b>	Ausgangswert ( y )	DINT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1400h	Abszissenwerte (X-Werte) des Kennfeldes nicht immer steigend				
1401h	Kein Kennfeld initialisiert				

### 3.3.10 FB\_PIDT1

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

Der P-I-DT1 stellt einen frei parametrierbaren diskreten Regler dar. Werden einzelne Anteile nicht benötigt, sowie der P, der I oder der DT1 Anteil, wird dessen Parameter mit 0 beschrieben. Der T1 Anteil arbeitet nur mit dem D Anteil zusammen. Es lässt sich also kein PT1 Regler parametrieren. Auf Grund von interner Speicherbegrenzung sind die Regelungsparameter auf folgende Bereiche begrenzt:

Zulässiger Wertebereich für Regelungsparameter			
Parameter	Wertebereich	Skalierung	resultierender Wertebereich
P (Kp)	0 – 32767	1/100	0,00 – 327,67
I (Ki)	0 – 10240	1/100	0,00 – 102,40
D (Kd)	0 – 32767	1/1000	0,000 – 32,767
T1 (ms)	0 – 32767	1/1000	0,000 – 32,767
Max	-32768 – 32767		
Min	-32768 – 32767		

Wenn der Eingang **ENABLE** auf TRUE gesetzt wird, beginnt der Regler zu rechnen. Die Regelungsparameter werden nur bei der steigenden Flanke von **ENABLE** übernommen. Während **ENABLE** auf TRUE ist, bleibt ein Verändern der Regelungsparameter wirkungslos. Wird **ENABLE** auf FALSE gesetzt, bleibt der Ausgang auf dem letzten Wert stehen.

Das Ausgangsbit **VALID** wird gesetzt, solange sich der Ausgangswert Q innerhalb der Grenzen Min und Max bewegt und der Eingang **ENABLE** auf TRUE steht.

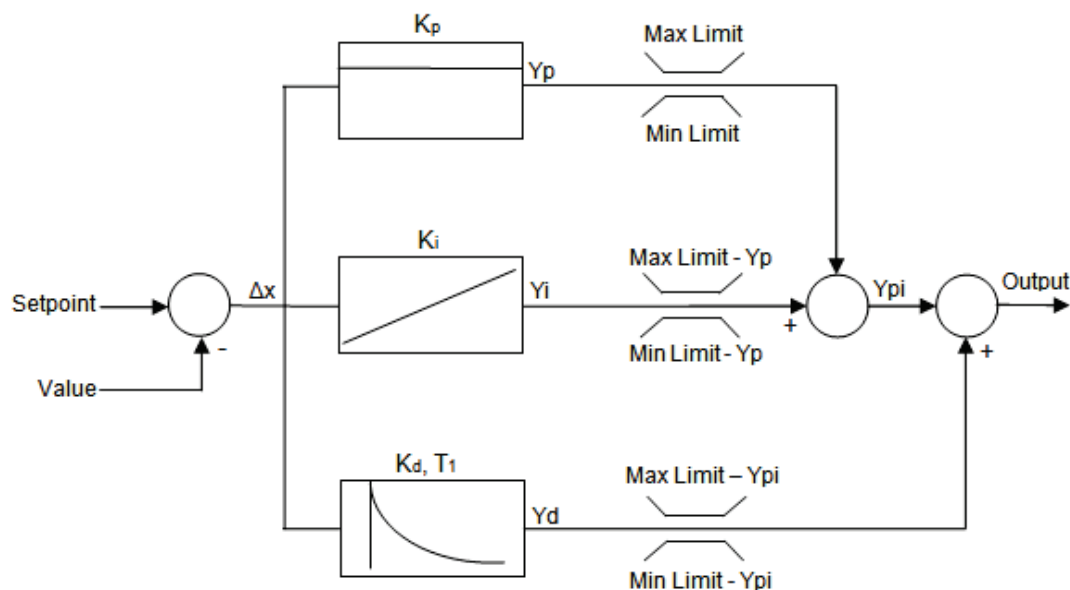
**ERROR** wird gesetzt, sobald ein Fehler aufgetreten ist. Das Bit **VALID** ist dann FALSE und die Fehlerursache ist über **ERRORID** (siehe Tabelle unten) zu erkennen.

Wird das Bit **RESET** auf TRUE gesetzt, werden der Integrator- und der Differenziatorinhalt auf 0 gesetzt. Ist der Eingang **ENABLE** auf FALSE, wird auch der Ausgang **OUTPUT** auf 0 gesetzt. Ist der Eingang **ENABLE** auf TRUE gesetzt, wirkt nur der P-Anteil auf den Ausgang **OUTPUT**.

Überschreitet der Ausgangswert **OUTPUT** die maximalen oder minimalen Ausgangswerte, wird das entsprechende Bit **MAXLIMIT** bzw. **MINLIMIT** gesetzt und das Bit **VALID** wird auf FALSE gesetzt.

#### Information

Kann das gesamte Programm nicht innerhalb von einem PLC Zyklus abgearbeitet werden, rechnet der Regler den Ausgangswert ein zweites Mal mit den alten Abtastwerten. Dadurch wird eine konstante Abtastrate erreicht. Aus diesem Grund ist es notwendig, dass der CAL Befehl für den PIDT1 Regler in jedem PLC Zyklus und nur am Ende des PLC Programms ausgeführt wird!



VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>ENABLE</b>	Ausführen	BOOL	<b>VALID</b>	Ausgangswert ist gültig	BOOL
<b>RESET</b>	Ausgangswerte zurücksetzen	BOOL	<b>ERROR</b>	Fehler im FB	BOOL
<b>P</b>	P-Anteil ( $K_p$ )	INT	<b>ERRORID</b>	Fehlercode	INT
<b>I</b>	I-Anteil ( $K_i$ )	INT	<b>MAXLIMIT</b>	Maximales Limit erreicht	BOOL
<b>D</b>	D-Anteil ( $K_d$ )	INT	<b>MINLIMIT</b>	Minimales Limit erreicht	BOOL
<b>T1</b>	T1-Anteil in ms	INT	<b>OUTPUT</b>	Ausgangswert	INT
<b>MAX</b>	Maximaler Ausgangswert	INT			
<b>MIN</b>	Minimaler Ausgangswert	INT			
<b>SETPOINT</b>	Sollwert	INT			
<b>VALUE</b>	Istwert	INT			
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
1600h	P-Anteil nicht im Wertebereich				
1601h	I-Anteil nicht im Wertebereich				
1602h	D-Anteil nicht im Wertebereich				
1603h	T1-Anteil nicht im Wertebereich				



### 3.3.11 FB\_ResetPostion

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	ab V2.3	ab V3.1	On+	ab V2.1	X	ab V1.2	

Bei einer Flanke auf den Eingang **EXECUTE**, wird die aktuelle Position (P601) auf den in Position eingetragenen Wert gesetzt. Ist im Parameter P609 ein Positionsoffset eingetragen, wird dieser Offset von der Position abgezogen.

Bei Absolutwertgebern kann die aktuelle Position nur auf 0 zurückgesetzt werden. Der Wert in Position wird nicht verwendet.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Ausführen	BOOL			
<b>Position</b>	Position	DINT			

### 3.3.12 FB\_Weigh

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	ab V2.3	ab V3.1	X	ab V2.1	X	ab V1.2	

Dieser Baustein dient zur Ermittlung des durchschnittlichen Drehmoments während einer Fahrt mit konstanter Drehzahl. Aus diesem Wert können dann z.B. physikalische Größen wie das bewegte Gewicht ermittelt werden.

Über eine positive Flanke am **EXECUTE** Eingang wird der FB gestartet. Mit der Flanke werden alle Eingänge am FB übernommen. Der FU verfährt mit der unter **SPEED** gesetzten Drehzahl. Nach Ablauf der unter **STARTTIME** gesetzten Zeit wird mit der Messung begonnen. Die Messdauer wird unter **MEASURETIME** definiert. Nach Ablauf der Messzeit stoppt der FU. Wenn der Eingang **REVERSE** = 1 ist, dann startet der Messvorgang erneut jedoch mit negierter Drehzahl. Ansonsten ist die Messung beendet, der Ausgang **DONE** geht auf 1 und in **VALUE** steht das Messergebnis.

Solange der Messvorgang läuft ist **BUSY** aktiv.

Die Skalierung des Messergebnis **VALUE** ist  $1 = 0,01\%$  vom Nenndrehmoment des Motors.

Der Aufruf eines anderen Motion FB stoppt die Messfunktion und der Ausgang **ABORT** geht auf 1.

Alle Ausgänge des FB werden mit einer neuen positiven Flanke an **EXECUTE** resetet.

VAR_INPUT			VAR_OUTPUT		
Eingang	Erläuterung	Typ	Ausgang	Erläuterung	Typ
<b>EXECUTE</b>	Ausführen	BOOL	<b>DONE</b>	Messung beendet	BOOL
<b>REVERSE</b>	Drehrichtungswechsel	BOOL	<b>BUSY</b>	Messung läuft	BOOL
<b>STARTTIME</b>	Zeit bis Messbeginn in ms	INT	<b>ABORT</b>	Messung abgebrochen	BOOL
<b>MEASURETIME</b>	Messzeit in ms	INT	<b>ERROR</b>	der FB hat einen Fehler	BOOL
<b>SPEED</b>	Messgeschwindigkeit in % (normiert auf die Maximalfrequenz, 16#4000 entspricht 100%)	INT	<b>ERRORID</b>	Fehlercode	INT
			<b>VALUE</b>	Messergebnis	INT
<b>ERRORID</b>	<b>Erläuterung</b>				
0	Kein Fehler				
0x1000	FU nicht eingeschaltet				
0x1101	Sollfrequenz nicht als Sollwert parametrier (P553)				
0x1C00	Wertebereich STARTTIME wurde überschritten				
0x1C01	Wertebereich MEASURETIME wurde überschritten				
0x1C02	Die Toleranz der Messwerte zueinander, ist größer als 1/8				

**Beispiel in ST:**

```
(* Gerät freigeben *)
Power(Enable := TRUE);
(* Ist das Gerät freigegeben? *)
if Power.Status then
  (* Startezeit festlegen 2000 ms *)
  Weigh.STARTTIME := 2000;
  (* Messzeit festlegen 1000 ms *)
  Weigh.MEASURETIME := 1000;
  (* Geschwindigkeit festlegen 25% der Maximalgeschwindigkeit *)
  Weigh.SPEED := 16#1000;
end_if;

Weigh(EXECUTE := Power.Status);
(* Wurde das Wiegen beendet? *)
if Weigh.done then
  Value := Weigh.Value;
end_if;
```

---

** Information**

Von diesem FB ist nur eine Instance im PLC Programm zulässig!

---

## 3.4 Operatoren

### 3.4.1 Arithmetische Operatoren

#### **i** Information

Einzelne der folgenden Operatoren können auch weiterführende Befehle beinhalten. Diese sind in Klammern hinter den Operator zu setzen. Dabei ist zu beachten, dass hinter der eröffnenden Klammer ein Leerzeichen stehen muss. Die schließende Klammer ist auf eine separate Programmzeile zu setzen.

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

#### 3.4.1.1 ABS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>			X	X

Bildet aus dem Akku den absoluten Betrag.

#### Beispiel in AWL:

```
LD -10 (* Lädt den Wert -10 *)
ABS (* Akku = 10 *)
ST Value1 (* Speichert den Wert 10 in Value1 ab *)
```

#### Beispiel in ST:

```
Value1 := ABS(-10); (* Das Ergebnis ist 10 *)
```

### 3.4.1.2 ADD und ADD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Addiert vorzeichenrichtig Variablen und Konstanten miteinander. Der erste Wert zur Addition befindet sich im Akku und der zweite wird mit dem ADD Befehl geladen oder er befindet sich innerhalb der Klammer. Es können auch mehrere Variablen oder Konstanten an den ADD Befehl angefügt werden. Bei der Klammer Addition wird der Akku mit dem Ergebnis des Klammersausdrucks addiert. Es sind bis zu 6 Klammerebenen möglich. Die zu addierenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 10
ADD 204 (* Addition zweier Konstanten *)
ST Value
LD 170 (* Addition einer Konstanten und 2 Variablen. *)
ADD Var1, Var2 (* 170dez + Var1 + Var2 *)
ST Value
LD Var1
ADD( Var2
SUB Var3 (* Var1 + ( Var2 - Var3 ) *)
)
ST Value
```

#### Beispiel in ST:

```
Ergebnis := 10 + 30; (* Das Ergebnis ist 40 *)
Ergebnis := 10 + Var1 + Var2;
```

### 3.4.1.3 DIV und DIV(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Dividiert den Akku durch den Operanden. Bei Divisionen durch null wird das maximal mögliche Ergebnis in den Akku eingetragen, z.B. bei einer Division mit INT Werten ist das der Wert 0x7FFF oder wenn der Divisor negativ ist dann ist es der Wert 0x8000. Bei der Klammer Division wird der Akku durch das Ergebnis des Klammersausdrucks dividiert. Es sind bis zu 6 Klammerebenen möglich. Die zu dividierenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 10
DIV 3 (* Division zweier Konstanten *)
ST iValue (* Das Ergebnis ist 9 *)
LD 170 (* Division einer Konstanten und 2 Variablen. *)
DIV Var1, Var2 (* (170dez : Var1) : Var2 *)
ST Value
LD Var1 (* Dividiere Var1 durch den Inhalt der Klammer *)
DIV( Var2
SUB Var3
) (* Var1 : ( Var2 - Var3 ) *)
ST Value
```

#### Beispiel in ST:

```
Ergebnis := 30 / 10; (* Das Ergebnis ist 3 *)
Ergebnis := 30 / Var1 / Var2;
```

### 3.4.1.4 LIMIT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Der Befehl begrenzt den im Akku stehenden Wert auf die übergebenen min. und max. Werte. Werte. Bei Überschreitung wird im Akku der max. Wert eingetragen und bei Unterschreitung der min. Wert. Liegt der Wert zwischen den Limits, so erfolgt keine Beeinflussung.

#### Beispiel in AWL:

```
LD 10 (* Lädt den Wert 10 in den Akku *)
LIMIT 20, 30 (* Der Wert wird mit den Grenzen 20 und 30 verglichen. *)
(* Der Wert im Akku ist kleiner, der Akku wird mit 20 überschrieben*)
ST iValue (* Speichert den Wert 20 in Value1 ab *)
```

#### Beispiel in ST:

```
Ergebnis := Limit(10, 20, 30); (* Das Ergebnis ist 20 *)
```

### 3.4.1.5 MAX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Dieser Befehl ermittelt den maximalen Wert von zwei Variablen oder Konstanten. Dazu wird der aktuelle Akku Inhalt mit dem im MAX Befehl übergebenen Wert verglichen. Der größere von beiden Werten befindet sich nach dem Befehl im Akku. Beide Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 100 (* Lade 100 in den Akku *)
MAX 200 (* Vergleiche mit dem Wert 200 *)
ST iValue (* Speichere 200 in Value2 (weil größter Wert) *)
```

#### Beispiel in ST:

```
Ergebnis := Max(100, 200); (* Das Ergebnis ist 200 *)
```

### 3.4.1.6 MIN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Dieser Befehl ermittelt den minimalen Wert von zwei Variablen oder Konstanten. Dazu wird der aktuelle Akku Inhalt dem im MIN Befehl übergebenen Wert verglichen. Der kleinere von beiden Werten befindet sich nach dem Befehl im Akku. Beide Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 100 (* Lade 100 in den Akku *)
MIN 200 (* Vergleiche mit dem Wert 200 *)
ST Value2 (* Speichere 100 in Value2 (weil kleinerer Wert) *)
```

#### Beispiel in ST:

```
Ergebnis := Min(100, 200); (* Speichere 100 in Value2 (weil kleinerer Wert) *)
```

### 3.4.1.7 MOD und MOD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Der Akku wird durch eine oder mehrere Variablen oder Konstanten dividiert, der Rest der Division steht als Ergebnis im Akku. Bei der Klammer Modulo wird der Akku durch das Ergebnis des Klammersausdrucks dividiert und daraus der Modulo gebildet. Es sind bis zu 6 Klammerebenen möglich.

#### Beispiel in AWL:

```
LD 25 (* Lade den Dividend *)
MOD 20 (* Division 25/20 à Modulo = 5 *)
ST Var1 (* Speicher Ergebnis 5 in Var1 *)
LD 25 (* Lade den Dividend *)
MOD( Var1 (* Ergebnis = 25/(Var1 + 10) à Modulo in den Akku *)
ADD 10
)
ST Var3 (* Speicher Ergebnis 10 in Var3 *)
```

#### Beispiel in ST:

```
Ergebnis := 25 MOD 20; (* Speicher Ergebnis 5 in Var1 *)
Ergebnis := 25 MOD (Var1 + 10); (* Ergebnis = 25/(Var1 + 10) à Modulo in den Akku *)
```

### 3.4.1.8 MUL und MUL(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Multiplikation des Akkus mit einer oder mehreren Variablen oder Konstanten. Bei der Klammer Multiplikation wird der Akku mit dem Ergebnis des Klammersausdrucks multipliziert. Es sind bis zu 6 Klammerebenen möglich. Beide Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 25 (* Lade den Multiplikator *)
MUL Var1, Var2 (* 25 * Var1 * Var2 *)
ST Var2 (* Speicher Ergebnis *)
LD 25 (* Lade den Multiplikator *)
MUL( Var1 (* Ergebnis = 25*(Var1 + Var2) *)
ADD Var2
)
ST Var3 (* Speicher Ergebnis als Variable Var3 *)
```

#### Beispiel in ST:

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



### 3.4.1.9 MUX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Über einen Index, der sich vor dem Befehl im Akku befindet, können verschiedene Konstanten oder Variablen selektiert werden. Der erste Wert wird über den Index 0 angesprochen. Der ausgewählte Wert wird in den Akku geladen. Die Anzahl der Werte ist nur durch den Programmspeicher limitiert.

#### Beispiel in AWL:

```
LD 1 (* Wähle das gewünschte Element aus *)
MUX 10,20,30,40,Value1 (* MUX Befehl mit 4 Konstanten und einer Variable *)
ST Value (* Speichere Ergebnis = 20 *)
```

#### Beispiel in ST:

```
Ergebnis := Mux(1, 10, 20, 30, 40, Value1) (* Speichere Ergebnis = 20 *)
```

### 3.4.1.10 SUB und SUB(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Subtrahiert den Akku mit einer oder mehreren Variablen oder Konstanten. Bei der Klammer Subtraktion wird der Akku mit dem Ergebnis des Klammersausdrucks subtrahiert. Es sind bis zu 6 Klammerebenen möglich. Die zu subtrahierenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 10
SUB Var1 (* Ergebnis = 10 - Var1 *)
ST Ergebnis
LD 20
SUB Var1, Var2, 30 (* Ergebnis = 20 - Var1 - Var2 - 30 *)
ST Ergebnis
LD 20
SUB( 6 (* Subtrahiere 20 mit den Inhalt der Klammer *)
AND 2
) (* Ergebnis = 20 - (6 AND 2) *)
ST Ergebnis (* Ergebnis = 18 *)
```

#### Beispiel in ST:

```
Ergebnis := 10 - Value1;
```

### 3.4.2 Erweiterte mathematische Operatoren

#### **i** Information

Die hier aufgeführten Operatoren sind sehr rechenintensiv. Es kann zu deutlich längeren Laufzeiten des PLC Programmes kommen.

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

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Datentyp				X

Berechnung der jeweiligen mathematischen Funktion. Der zu berechnende Wert muss im Akku in Bogenminuten vorliegen. Die Skalierung entspricht 1 = 1000.

Umrechnung: Winkel in Bogenmaß = (Winkel in Grad \* PI / 180) \* 1000 z.B. ein Winkel von 90° wird wie folgt umgerechnet à 90° \* 3.14 / 180) \* 1000 = 1571

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

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.4.2.2 EXP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Datentyp				X

Bildet aus dem Akku die Exponentialfunktion zur Basis der Eulerschen Zahl (2,718). Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,002 muss als 1002 eingegeben werden.

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

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.4.2.3 LN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Datentyp				X

Logarithmus zur Basis e (2,718). Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

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

#### Beispiel in AWL:

```
LD 1234
LN
ST Ergebnis
```

#### Beispiel in ST:

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

### 3.4.2.4 LOG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Datentyp				X

Bildet aus dem Akku den Logarithmus zur Basis 10. Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

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

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.4.2.5 SQRT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Datentyp				X

Bildet aus dem Akku die Quadratwurzel. Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

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

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.4.3 Bit Operatoren

#### 3.4.3.1 AND und AND(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X	X	X	X

Bitweise UND Verknüpfung des AE/Akku mit einer oder zwei Variablen oder Konstanten. Bitweise UND(...) Verknüpfung mit dem AE/Akku und dem AE/Akku welches zuvor in der Klammer gebildet wurde. Es sind bis zu 6 Klammerebenen möglich. Alle Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 170
AND 204 (* AND Verknüpfung zwischen 2 Konstanten *)
(* Akku = 136 (Siehe Beispiel unter der Tabelle) *)

LD 170 (* Verknüpfung zwischen einer Konstanten und 2 Variablen.*)
AND Var1, Var2 (* Akku = 170dez AND Var1 AND Var2 *)

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

#### Beispiel in ST:

```
Ergebnis := 170 AND 204; (* Ergebnis = 136dez *)
```

Var2	Var1	Ergebnis
0	0	0
0	1	0
1	0	0
1	1	1

Beispiel: 170dez (1010 1010bin) AND 204dez (1100 1100bin) = (1000 1000bin) 136dez

### 3.4.3.2 ANDN und ANDN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X	X	X	X

Bitweise UND Verknüpfung des AE/Akkus mit einem negierten Operanden. Bitweise UND (...) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich. Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 2#0000_1111
ANDN 2#0011_1010 (* ANDN Verknüpfung zwischen 2 Konstanten *)
(* Akku = 2#1111_0101 *)

LD 170 (* Verknüpfung zwischen einer Konstanten und 2 Variablen. *)
ANDN Var1, Var2 (* Akku = 170d ANDN Var1 ANDN Var2 *)

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

Var2	Var1	Ergebnis
0	0	1
0	1	1
1	0	1
1	1	0

Beispiel: 170dez (1010 1010bin) AND 204dez (1100 1100bin) = (1000 1000bin) 136dez

**3.4.3.3 NOT**

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp	X	X	X	X

Bitweise Negation des Akkus.

**Beispiel in AWL:**

```
LD BYTE#10 (* Lade In den AKKU den Wert 10dez im Format Byte *)
NOT (* Der Wert wird auf Bit - Ebene aufgelöst (0000 1010), *)
(* bitweise negiert (1111 0101) und wieder in einen Dezimalwert *)
(* gewandelt, Ergebnis = 245dez *)
ST Var3 (* Speicher Ergebnis als Variable Var3 *)
```

**Beispiel in ST:**

```
Ergebnis := not BYTE#10; (* Ergebnis = 245dez *)
```

### 3.4.3.4 OR und OR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X	X	X	X

Bitweise ODER Verknüpfung des AE/Akku mit einer oder zwei Variablen oder Konstanten. Bitweise ODER(...) Verknüpfung mit dem AE/Akku und dem AE/Akku welches zuvor in der Klammer gebildet wurde. Es sind bis zu 6 Klammerebenen möglich. Alle Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 170
OR 204 (* OR Verknüpfung zwischen 2 Konstanten *)

LD 170 (* Verknüpfung zwischen einer Konstanten und 2 Variablen. *)
OR Var1, Var2 (* Akku = 170d OR Var1OR Var2 *)

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

#### Beispiel in ST:

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

Var2	Var1	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	1



3.4.3.5 ORN undORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp	X	X	X	X

Bitweise ODER Verknüpfung des AE/Akkus mit einem negierten Operanden. Bitweise ODER (...) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich. Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

**Beispiel in AWL:**

```
LD 2#0000_1111
ORN 2#0011_1010 (* ORN Verknüpfung zwischen 2 Konstanten *)
(* Akku = 2#1100_0000 *)

LD 170 (* Verknüpfung zwischen einer Konstanten und 2 Variablen. *)
ORN Var1, Var2 (* Akku = 170d ORN Var1 ORN Var2 *)

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

**Beispiel in ST:**

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

Var2	Var1	Ergebnis
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	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Bitweise Linksrotation des Akkus. Dabei wird der Inhalt des Akkus um n mal nach links verschoben, wobei das links Bit wieder rechts reingeschoben wird.

#### Beispiel in AWL:

```
LD 175      (* Lädt den Wert 1010_1111*)
ROL 2       (* Akku Inhalt wird 2 mal nach links rotiert *)
ST Value1  (* Speichert den Wert 1011_1110 ab *)
```

#### Beispiel in ST:

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

### 3.4.3.7 ROR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Bitweise Rechtsrotation des Akkus. Dabei wird der Inhalt des Akkus um n mal nach rechts verschoben, wobei das rechte Bit wieder links reingeschoben wird.

#### Beispiel in AWL:

```
LD 175      (* Lädt den Wert 1010_1111*)
ROR 2       (* Akku Inhalt wird 2 mal nach rechts rotiert *)
ST Value1  (* Speichert den Wert 1110_1011 ab *)
```

#### Beispiel in ST:

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

### 3.4.3.8 S und R

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp	X			

Setzen und Rücksetzen einer booleschen Variable, wenn das vorherige Verknüpfungsergebnis (das AE) TRUE war.

#### Beispiel in AWL:

```
LD TRUE (* Lädt das AE mit TRUE *)
S Var1 (* VAR1 wird TRUE gesetzt *)
R Var1 (* VAR1 wird FALSE gestzt *)
```

#### Beispiel in ST:

```
Ergebnis := TRUE;
Ergebnis := FALSE;
```

### 3.4.3.9 SHL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Bitweises Linksschieben des Akkus. Dabei wird der Inhalt des Akku um n mal nach links verschoben, die rausgeschobenen Bits sind verloren.

#### Beispiel in AWL:

```
LD 175 (* Lädt den Wert 1010_1111 *)
SHL 2 (* Akku Inhalt wird 2 mal nach links verschoben *)
ST Value1 (* Speichert den Wert 1011_1100 ab *)
```

#### Beispiel in ST:

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

### 3.4.3.10 SHR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Bitweises Rechtsschieben des Akkus. Dabei wird der Inhalt des Akkus um n mal nach rechts verschoben, die rausgeschobenen Bits sind verloren.

#### Beispiel in AWL:

```
LD 175      (* Lädt den Wert 1010_1111 *)
SHR 2      (* Akku Inhalt wird 2 mal nach rechts verschoben *)
ST Value1  (* Speichert den Wert 0010_1011 ab *)
```

#### Beispiel in ST:

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

### 3.4.3.11 XOR und XOR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp	X			

Bitweises „Exklusiv Oder“ Verknüpfung zwischen dem AE/Akku und ein bis zwei Variablen oder Konstanten. Der erste Wert befindet sich im AE/Akku der zweite wird mit dem Befehl geladen oder er befindet sich innerhalb der Klammer. Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 2#0000_1111
XOR 2#0011_1010 (* XOR Verknüpfung zwischen 2 Konstanten *)
                (* Akku = 2#0011_0101 *)

LD 170          (* Verknüpfung zwischen einer Konstanten und 2 Variablen. *)
XOR Var1, Var2 (* Akku = 170d XOR Var1 XOR Var2 *)

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

#### Beispiel in ST:

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

Var2	Var1	Ergebnis
0	0	0
0	1	1
1	0	1
1	1	0

### 3.4.3.12 XORN und XORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X			

Bitweise Exklusiv ODER Verknüpfung des AE/Akkus mit einem negierten Operanden. Bitweise Exklusiv ODER (...) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich. Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

#### Beispiel in AWL:

```
LD 2#0000_1111
XORN 2#0011_1010 (* XORN Verknüpfung zwischen 2 Konstanten *)
(* Akku = 2#1100_1010 *)

LD 170 (* Verknüpfung zwischen einer Konstanten und 2 Variablen. *)
XORN Var1, Var2 (* Akku = 170d XORN Var1 XORN Var2 *)

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

#### Beispiel in ST:

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

Var2	Var1	Ergebnis
0	0	1
0	1	0
1	0	0
1	1	1

### 3.4.4 Lade- und Speicheroperatoren

#### 3.4.4.1 LD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X	X	X	X

Lädt eine Konstante oder eine Variable in den AE bzw. in den Akku.

#### Beispiel in AWL:

```
LD 10 (* Lädt die 10 als BYTE *)
LD -1000 (* Lädt die -1000 als INT *)
LD Value1 (* Lädt die Variable Value1 *)
```

#### 3.4.4.2 LDN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X			

Lädt eine boolesche Variablen negiert in den AE.

#### Beispiel in AWL:

```
LDN Value1 (* Value1 = TRUE à AE = FALSE *)
ST Value2 (* Speicher auf Value2 = FALSE *)
```

### 3.4.4.3 ST

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X	X	X	X

Speichert den Inhalt des AE/Akku auf eine Variable ab. Die abzuspeichernde Variable muss zu dem vorher geladenen und verarbeiteten Datentyp passen.

#### Beispiel in AWL:

```
LD 100 (* Lädt den Wert 1010_1111 *)
ST Value1 (* Akku Inhalt 100 wird in Value1 abgespeichert *)
```

### 3.4.4.4 STN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>	X			

Speichert den Inhalt des AE auf eine Variable ab und negiert ihn. Die abzuspeichernde Variable muss zu dem vorher geladenen und verarbeiteten Datentyp passen.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = TRUE à AE = TRUE *)
STN Value2 (* Speicher auf Value2 = FALSE *)
```



### 3.4.5 Vergleichs Operatoren

#### 3.4.5.1 EQ

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Sind die Werte gleich, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 5 *)
EQ 10 (* AE = Ist 5 gleich 10 ? *)
JMPC NextStep (* AE = FALSE à Programm springt nicht *)
ADD 1
NextStep:
ST Value1
```

#### Beispiel in ST:

```
(* Ist Value = 10 *)
if Value = 10 then
  Value2 := 5;
end_if;
```

#### 3.4.5.2 GE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Ist der Wert im Akku größer oder gleich der Variabel oder Konstante, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 5 *)
GE 10 (* Ist 5 größer oder gleich 10? *)
JMPC NextStep (* AE = FALSE à Programm springt nicht *)
ADD 1

NextStep:
ST Value1
```

#### Beispiel in ST:

```
(* Ist 5 größer oder gleich 10? *)
if Value >= 10 then
  Value := Value - 1
end_if;
```

### 3.4.5.3 GT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Ist der Wert im Akku größer als die Variabel oder Konstante, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 12 *)
GT 8 (* Ist 12 größer als 8? *)
JMPC NextStep (* AE = TRUE - Programm springt *)
ADD 1
NextStep:
ST Value1
```

#### Beispiel in ST:

```
(* Ist 12 größer als 8? *)
if Value > 8 then
    Value := 0;
end_if;
```

### 3.4.5.4 LE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Ist der Wert im Akku kleiner oder gleich der Variablen oder Konstante, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 5 *)
LE 10 (* Ist 5 kleiner oder gleich 10? *)
JMPC NextStep:
ST Value1
```

#### Beispiel in ST:

```
(* Ist Value kleiner oder gleich 10?*)
if Value <= 10 then
    Value := 11;
end_if;
```

### 3.4.5.5 LT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Ist der Wert im Akku kleiner als die Variablen oder Konstante, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 12 *)
LT 8 (* Ist 12 kleiner 8 ? *)
JMPC NextStep (* AE = FALSE à Programm springt nicht *)
ADD 1
NextStep:
ST Value1
```

#### Beispiel in ST:

```
(* Ist Value kleiner als 0? *)
if Value < 0 then
  Value := 0;
end_if;
```

### 3.4.5.6 NE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X	X	X

Vergleicht den Inhalt vom Akku mit einer Variabel oder Konstanten. Ist der Wert im Akku ungleich der Variablen oder Konstante, dann wird das AE auf TRUE gesetzt.

#### Beispiel in AWL:

```
LD Value1 (* Value1 = 5 *)
NE 10 (*Ist 5 ungleich 10 ?*)
JMPC NextStep (* AE = TRUE à Programm springt *)
ADD 1
NextStep:
ST Value1
```

#### Beispiel in ST:

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

### 3.5 Prozesswerte

Alle analogen und digitalen Ein- und Ausgänge bzw. Bussoll- und Istwert können durch die PLC gelesen und weiterverarbeitet bzw. durch die PLC gesetzt (wenn Ausgangswert) werden. Der Zugriff auf die einzelnen Werte erfolgt über die hier nachfolgend aufgeführten Prozesswerte. Für alle Ausgangswerte muss der Ausgang (z.B. Digitalausgänge oder PLC Sollwert) so programmiert werden, dass als Ereignisquelle die PLC vorgesehen ist. Alle Prozessdaten werden von der PLC bei jedem neuen zyklischen Durchlauf am Anfang vom Gerät eingelesen und erst am Ende des PLC Programms in das Gerät geschrieben! In den nachfolgenden Tabellen sind alle Werte dargestellt, auf welche die PLC – Funktion direkt zugreifen kann. Auf alle anderen Prozesswerte muss über die Funktionsblöcke MC\_ReadParameter oder MC\_WriteParameter zugegriffen werden.

#### 3.5.1 Ein- und Ausgänge

Hier sind alle Prozesswerte zusammengefasst, die das I/O- Interface des Gerätes beschreiben.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_0_Set_digital_output	Setzen digitaler Ausgänge	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: dig. Fkt. AOUT Bit 9: frei Bit 10: BusIO Bit0 Bit 11: BusIO Bit1 Bit 12: BusIO Bit2 Bit 13: BusIO Bit3 Bit 14: BusIO Bit4 Bit 15: BusIO Bit5	UINT	R/W	SK 5xxP On/On+
_0_Set_digital_output	Setzen digitaler Ausgänge	Bit 0: Mfr1 Bit 1: Mfr2 Bit 2: DOUT1 Bit 3: DOUT2 Bit 4: dig. Fkt. AOUT Bit 5: DOUT3 (Din7) Bit 6: Statuswort Bit 10 Bit 7: Statuswort Bit 13 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7	UINT	R/W	SK 54xE
_0_Set_digital_output	Setzen digitaler Ausgänge	Bit 0: Mfr1 Bit 1: Mfr2	UINT	R/W	SK 52xE SK 53xE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
		Bit 2: DOUT1 Bit 3: DOUT2 Bit 4: dig. Fkt. AOUT Bit 5: frei Bit 6: Statuswort Bit 10 Bit 7: Statuswort Bit 13 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7			
_0_Set_digital_output	Setzen digitaler Ausgänge	Bit 0: DOUT1 Bit 1: BusIO Bit0 Bit 2: BusIO Bit1 Bit 3: BusIO Bit2 Bit 4: BusIO Bit3 Bit 5: BusIO Bit4 Bit 6: BusIO Bit5 Bit 7: BusIO Bit6 Bit 8: BusIO 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	Setzen digitaler Ausgänge	Bit 0: DOUT1 Bit 1: DOUT2 Bit 2: BusIO Bit0 Bit 3: BusIO Bit1 Bit 4: BusIO Bit2 Bit 5: BusIO Bit3 Bit 6: BusIO Bit4 Bit 7: BusIO Bit5 Bit 8: BusIO Bit6 Bit 9: BusIO Bit7 Bit 10: Bus PZD Bit 10 Bit 11: Bus PZD Bit 13	UINT	R/W	SK 180E SK 190E
_0_Set_digital_output	Setzen digitaler Ausgänge	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	Setzen analoger Ausgang FU	10,0V = 100	BYTE	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE On/On+
_2_Set_external_	Setzen analoger	10,0V = 100	BYTE	R/W	SK 5xxP

Name	Funktion	Normierung	Typ	Zugriff	Gerät
analog_out1	Ausgang 1. IOE				SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_3_Set_external_analog_out2	Setzen analoger Ausgang 2. IOE	10,0V = 100	BYTE	R/W	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_4_State_digital_output	Zustand digitale Ausgänge	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: dig. Fkt. AOUT Bit 9: frei Bit 10: DOUT1 IOE1 Bit 11: DOUT2 IOE1 Bit 12: DOUT1 IOE2 Bit 13: DOUT2 IOE2 Bit 14: frei Bit 15: frei	INT	R	SK 5xxP On/On+
_4_State_digital_output	Zustand digitale Ausgänge	Bit 0: Mfr1 Bit 1: Mfr2 Bit 2: DOUT1 Bit 3: DOUT2 Bit 4: dig. Fkt. AOUT Bit 5: DOUT3 (Din7) Bit 6: Statuswort Bit 8 Bit 7: Statuswort Bit 9 Bit 8: BusIO Bit0 Bit 9: BusIO Bit1 Bit 10: BusIO Bit2 Bit 11: BusIO Bit3 Bit 12: BusIO Bit4 Bit 13: BusIO Bit5 Bit 14: BusIO Bit6 Bit 15: BusIO Bit7	INT	R	SK 54xE
_4_State_digital_output	Zustand digitale Ausgänge	P711	BYTE	R	SK 52xE SK 53xE SK 2xxE SK 2xxE-FDS SK 180E

Name	Funktion	Normierung	Typ	Zugriff	Gerät
					SK 190E
_4_State_digital_output	Zustand digitale Ausgänge	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	Zustand digitale Eingänge	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 frei Bit 11 frei Bit 12: Digitalfunktion AIN1 Bit 8: Digitalfunktion AIN2	INT	R	SK 5xxP On/On+
_5_State_Digital_input	Zustand digitale Eingänge	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7 Bit 7: Digitalfunktion AIN1 Bit 8: Digitalfunktion AIN2	INT	R	SK 54xE
_5_State_Digital_input	Zustand digitale Eingänge	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	Zustand digitale Eingänge	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: frei Bit 5: Kaltleiter Bit 6: frei Bit 7: frei Bit 8: DIN1 IOE 1	INT	R	SK 2xxE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
		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			
_5_State_Digital_input	Zustand digitale Eingänge	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: AIN1 Bit 4: AIN2 Bit 5: Kaltleiter Bit 6: frei Bit 7: frei 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	Zustand digitale Eingänge	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: TF (Kaltleiter) Bit 4: DIN-BUS1 (ASi1) Bit 5: DIN-BUS2 (ASi2) Bit 6: DIN-BUS3 (ASi3) Bit 7: DIN-BUS4 (ASi4) Bit 8: BDDI1 (ASIO3) Bit 9: BDDI2 (ASIO4) Bit 10: STO	INT	R	SK 155E-FDS SK 175E-FDS
_5_State_Digital_input	Zustand digitale Eingänge	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: Kaltleiter 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



Name	Funktion	Normierung	Typ	Zugriff	Gerät
_6_Delay_digital_inputs	Zustand digitale Eingänge nach P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: DIN4 Bit 4: DIN5 Bit 5: DIN6 Bit 6: DIN7 Bit 7: Digitalfunktion AIN1 Bit 8: Digitalfunktion AIN2	INT	R	SK 5xxP SK 54xE On/On+
_6_Delay_digital_inputs	Zustand digitale Eingänge nach 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	Zustand digitale Eingänge nach P475	Bit 0: DIN1 Bit 1: DIN2 Bit 2: DIN3 Bit 3: AIN1 Bit 4: AIN2 Bit 5: Kaltleiter Bit 6: free Bit 7: free 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	Zustand digitale Eingänge nach 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: Kaltleiter 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

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_7_Analog_input1	Wert Analogeingang 1 (AIN1)	10,00V = 1000	INT	R	alle
_8_Analog_input2	Wert Analogeingang 2 (AIN2)	10,00V = 1000	INT	R	alle
_9_Analog_input3	Wert Analogfunktion DIN2	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_10_Analog_input4	Wert Analogfunktion DIN3	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_11_External_analog_input1	Wert analoger Eingang 1 (1.IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_12_External_analog_input2	Wert analoger Eingang 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	Wert analoger Eingang 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	Wert analoger Eingang 2 (2.IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_15_State_analog_output	Zustand analoger Ausgang	10,0V = 100	BYTE	R	SK 5xxP SK 54xE
_16_State_ext_analog_out1	Zustand Analogausgang (1. IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_17_State_ext_analog_out2	Zustand Analogausgang (2. IOE)	10,00V = 1000	INT	R	SK 5xxP SK 54xE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
					SK 2xxE SK 180E SK 190E
_18_Dip_switch_state	Zustand der DIP Schalter	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
_19_State_digital_input _IOE	Zustand digitale Eingänge (IOE)	Bit 0: DIN1 IOE 2 Bit 1: DIN2 IOE 2 Bit 2: DIN3 IOE 2 Bit 3: DIN4 IOE 2 Bit 4: DIN1 IOE 1 Bit 5: DIN2 IOE 1 Bit 6: DIN3 IOE 1 Bit 7: DIN4 IOE 1	INT	R	SK 5xxP On/On+

### 3.5.2 PLC Soll- und Istwerte

Die hier aufgeführten Prozesswerte bilden die Schnittstelle der PLC zum Gerät. Die Funktion der PLC Sollwerte wird im (P553) festgelegt.

#### **i** Information

Der Prozesswert PLC\_control\_word überschreibt den Funktionsblock MC\_Power. Die PLC Sollwerte überschreiben die Funktionsblöcke MC\_Move.... und MC\_Home.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_20_PLC_control_word	PLC Steuerwort	Entspricht USS Profil	INT	R/W	alle
_21_PLC_set_val1	PLC Sollwert 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_22_PLC_set_val2	PLC Sollwert 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_23_PLC_set_val3	PLC Sollwert 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_24_PLC_set_val4	PLC Sollwert 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS On/On+
_25_PLC_set_val5	PLC Sollwert 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
					SK 2xxE-FDS On/On+
_26_PLC_additional_control_word1	PLC Zusatzsteuerwort 1	Entspricht USS Profil	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_27_PLC_additional_control_word2	PLC Zusatzsteuerwort 2	Entspricht USS Profil	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_28_PLC_status_word	PLC Statuswort	Entspricht USS Profil	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_29_PLC_act_val1	PLC Istwert 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_30_PLC_act_val2	PLC Istwert 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_31_PLC_act_val3	PLC Istwert 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
					SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_32_PLC_act_val4	PLC Istwert 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS On/On+
_33_PLC_act_val5	PLC Istwert 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS On/On+
_34_PLC_Busmaster_Control_word	Steuerwort der Leitfunktion (Busmasterfunktion) über PLC	Entspricht USS Profil	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_35_PLC_32Bit_set_val1	32Bit PLC Sollwert - P553[1] = Low Part des 32Bit Wert - P553[2] = High Part des 32Bit Wert	–	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_36_PLC_32Bit_act_val1	32Bit PLC Istwert - PLC Istwert 1 = Low Part des 32Bit Wert - PLC Istwert 2 = High Part des 32Bit Wert	–	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_37_PLC_status_bits	Virtuelle Status-	Bit 0: PLC-DOU1	INT	R/W	SK 155E-FDS

Name	Funktion	Normierung	Typ	Zugriff	Gerät
	Ausgänge der PLC	Bit 1: PLC-DOOUT2			SK 175E-FDS
_38_PLC_control_bits	Virtuelle Steuer- Ausgänge der 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
_39_PLC_set_digital_ output_bus	Ausgehende PLC BusI/O Daten	Bit 0: BusIO Bit0 Bit 1: BusIO Bit1 Bit 2: BusIO Bit2 Bit 3: BusIO Bit3 Bit 4: BusIO Bit4 Bit 5: BusIO Bit5 Bit 6: BusIO Bit6 Bit 7: BusIO Bit7 Bit 8: Merker 1 Bit 9: Merker 2 Bit 10: Statuswort Bit 11 Bit 11: Statuswort Bit 12	INT	R/W	SK 5xxP On/On+

### 3.5.3 Bus Soll- und Istwerte

Diese Prozesswerte spiegeln alle Soll- und Istwerte wieder, die über die verschiedenen Bussysteme in das Gerät gelangen.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_40_Inverter_status	FU Statuswort	Entspricht USS Profil	INT	R	alle
_41_Inverter_act_val1	FU Istwert 1	100% = 4000h	INT	R	alle
_42_Inverter_act_val2	FU Istwert 2	100% = 4000h	INT	R	alle
_43_Inverter_act_val3	FU Istwert 3	100% = 4000h	INT	R	alle
_44_Inverter_act_val4	FU Istwert 4	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_45_Inverter_act_val5	FU Istwert 5	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_46_Inverter_lead_val1	Broadcast Master Funktion: Leitwert 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_47_Inverter_lead_val2	Broadcast Master Funktion: Leitwert 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_48_Inverter_lead_val3	Broadcast Master Funktion: Leitwert 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_49_Inverter_lead_val4	Broadcast Master Funktion: Leitwert 4	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_50_Inverter_lead_val5	Broadcast Master	100% = 4000h	INT	R	SK 5xxP



Name	Funktion	Normierung	Typ	Zugriff	Gerät
	Funktion: Leitwert 5				SK 54xE On/On+
_51_Inverter_control_word	Resultierendes Steuerwort Bus	Entspricht USS Profil	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_52_Inverter_set_val1	Resultierender Hauptsollwert 1 Bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_53_Inverter_set_val2	Resultierender Hauptsollwert 2 Bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_54_Inverter_set_val3	Resultierender Hauptsollwert 3 Bus	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_55_Inverter_set_val4	Resultierender Hauptsollwert 4 Bus	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_56_Inverter_set_val5	Resultierender Hauptsollwert 5 Bus	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_57_Broadcast_set_val1	Broadcast Slave: Nebensollwert 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
					SK 2xxE-FDS SK 180E SK 190E On/On+
_58_Broadcast_set_val 2	Broadcast Slave: Nebensollwert 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_59_Broadcast_set_val 3	Broadcast Slave: Nebensollwert 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_60_Broadcast_set_val 4	Broadcast Slave: Nebensollwert 4	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_61_Broadcast_set_val 5	Broadcast Slave: Nebensollwert 5	100% = 4000h	INT	R	SK 5xxP SK 54xE On/On+
_62_Inverter_32Bit_set _val1	Resultierender 32Bit Hauptsollwert 1 Bus	- Low Part in P546[1] - High Part in P546[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_63_Inverter_32Bit_act _val1	FU 32Bit Istwert 1	- Low Part in P543[1] - High Part in P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_64_Inverter_32Bit_lea d_val1	32Bit Leitwert 1	- Low Part in P502[1] - High Part in P502[2]	LONG	R	SK 5xxP SK 54xE SK 2xxE SK 180E SK 190E On/On+

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_65_Broadcast_32Bit_set_val1	32Bit Broadcast Slave Nebensollwert 1	- Low Part in P543[1] - High Part in P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_66_BusIO_input_bits	Eingehende BusI/O Daten	- Bit0 – 7 = Bus I/O In Bit 0 – 7 - Bit 8 = Merker 1 - Bit 9 = Merker 2 - Bit 10 = Bit8 vom Bus Steuerwort - Bit 11 = Bit9 vom Bus Steuerwort	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_67_BusIO_output_bits	Ausgehende BusI/O Daten	Bit0 = Bus / AS-i Dig Out1 Bit1 = Bus / AS-i Dig Out2 Bit2 = Bus / AS-i Dig Out3 Bit3 = Bus / AS-i Dig Out4 Bit4 = Bus / 1.IOE Dig Out1 Bit5 = Bus / 1.IOE Dig Out2 Bit6 = Bus / 2.IOE Dig Out1 Bit7 = Bus / 2.IOE Dig Out2 Bit8 = Bit 10 Bus Statuswort Bit9 = Bit 11 Bus Statuswort	INT	R	SK 5xxP SK 54xE On/On+
_67_BusIO_output_bits	Ausgehende BusI/O Daten	Bit0 = Bus / AS-i Dig Out1 Bit1 = Bus / AS-i Dig Out2 Bit2 = Bus / AS-i Dig Out3 Bit3 = Bus / AS-i Dig Out4 Bit4 = AS-i Aktor 1 Bit5 = AS-i Aktor 2 Bit6 = Merker 1 Bit7 = Merker 2 Bit8 = Bit 10 Bus Statuswort Bit9 = Bit 11 Bus	INT	R	SK 53xE SK 52xE

Name	Funktion	Normierung	Typ	Zugriff	Gerät
		Statuswort			
_67_BusIO_output_bits	Ausgehende Bus/O Daten	Bit0 = Bus / AS-i Dig Out1 Bit1 = Bus / AS-i Dig Out2 Bit2 = Bus / AS-i Dig Out3 Bit3 = Bus / AS-i Dig Out4 Bit4 = Bus / IOE Dig Out1 Bit5 = Bus / IOE Dig Out2 Bit6 = Bus / 2nd IOE Dig Out1 Bit7 = Bus / 2nd IOE Dig Out2 Bit8 = Bit 10 Bus Statuswort Bit9 = Bit 11 Bus Statuswort	INT	R	SK 2xxE
_67_BusIO_output_bits	Ausgehende Bus/O Daten	Bit0 = Bus / AS-i Dig Out1 Bit1 = Bus / AS-i Dig Out2 Bit2 = Bus / AS-i Dig Out3 Bit3 = Bus / AS-i Dig Out4 Bit4 = Bus / AS-i Dig Out5 Bit5 = Bus / AS-i Dig Out6 Bit6 = Bus / 2nd IOE Dig Out1 Bit7 = Bus / 2nd IOE Dig Out2 Bit8 = Bit 10 Bus Statuswort Bit9 = Bit 11 Bus Statuswort	INT	R	SK 2xxE-FDS

### 3.5.4 ControlBox und ParameterBox

Über die hier aufgeführten Prozesswerte kann auf die Bedienboxen zugegriffen werden. Damit ist die Realisierung einfacher HMI Anwendungen möglich.

#### **i** Information

Damit die „key\_states“ in der PLC angezeigt werden, müssen sich die Control- und die ParameterBox im PLC-Anzeige-Modus befinden. Anderenfalls wird nur ein Wert „0“ dargestellt.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_70_Set_controlbox_show_val	Anzeigewert für die ControlBox	Anzeigewert = Bit 29 – Bit 0 Kommastelle = Bit 31 – Bit30	DINT	R/W	alle
_71_Controlbox_key_state	Tastaturzustand der ControlBox	Bit 0: ON Bit 1: OFF Bit 2: DIR Bit 3: UP Bit 4: DOWN Bit 5: Enter	BYTE	R	alle
_72_Parameterbox_key_state	Tastaturzustand der 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	alle

### 3.5.5 Infoparameter

Hier sind die wichtigsten Istwerte des Gerätes aufgeführt.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_80_Current_fault	aktuelle Störungsnummer	Fehler 10.0 = 100	BYTE	R	alle
_81_Current_warning	aktuelle Warnung	Warnung 10.0 = 100	BYTE	R	alle
_82_Current_reason_FI_blocked	aktuelle Ursache für den Zustand Einschaltsperr	Problem 10.0 = 100	BYTE	R	alle
_83_Input_voltage	aktuelle Netzspannung	100 V = 100	INT	R	alle
_84_Current_frequenz	aktuelle Frequenz	10Hz = 100	INT	R	alle
_85_Current_set_point_frequency1	aktuelle Sollfrequenz von der Sollwertquelle	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_86_Current_set_point_frequency2	aktuelle Sollfrequenz Umrichter	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_87_Current_set_point_frequency3	aktuelle Sollfrequenz nach Rampe	10Hz = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_88_Current_Speed	aktuelle berechnete Drehzahl	100rpm = 100	INT	R	alle
_89_Actual_current	aktueller Ausgangsstrom	10.0A = 100	INT	R	alle
_90_Actual_torque_current	aktueller Momentstrom	10.0A = 100	INT	R	alle
_91_Current_voltage	aktuelle Spannung	100V = 100	INT	R	alle

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_92_Dc_link_voltage	aktuelle Zwischenkreisspannung	100V = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_93_Actual_field_current	aktueller Feldstrom	10.0A = 100	INT	R	alle
_94_Voltage_d	aktuelle Spannungskomponente d-Achse	100V = 100	INT	R	alle
_95_Voltage_q	aktuelle Spannungskomponente q-Achse	100V = 100	INT	R	alle
_96_Current_cos_phi	aktueller Cos(phi)	0.80 = 80	BYTE	R	alle
_97_Torque	aktuelles Drehmoment	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_98_Field	aktuelles Feld	100% = 100	BYTE	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_99_Apparent_power	aktuelle Scheinleistung	1,00KW = 100	INT	R	alle
_100_Mechanical_power	aktuelle mechanische Leistung	1,00KW = 100	INT	R	alle
_101_Speed_encoder	aktuelle gemessene Drehzahl	100rpm = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE On/On+
_102_Usage_rate_motor	aktuelle Auslastung Motor (Momentanw.)	100% = 100	INT	R	alle

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_103_Usage_rate_motor_I2t	aktuelle Auslastung Motor I2t	100% = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_104_Usage_rate_brake_resistor	aktuelle Auslastung Bremswiderstand	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_105_Head_sink_temp	aktuelle Kühlkörpertemperatur	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_106_Inside_temp	aktuelle Innenraumtemperatur	100°C = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_107_Motor_temp	aktuelle Motortemperatur	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E On/On+
_108_Actual_net_frequency	aktuelle Netzfrequenz	10Hz = 100	INT	R	SK 155E-FDS SK 175E-FDS
_109_Mains_phase_sequence	aktuelle Netz-Phasenfolge	0=CW, 1=CCW	BYTE	R	SK 155E-FDS SK 175E-FDS



Name	Funktion	Normierung	Typ	Zugriff	Gerät
_141_Pos_Sensor_Inc	Position des Inkrementalgebers	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_142_Pos_Sensor_Abs	Position des Absolutwertgebers	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_143_Pos_Sensor_Uni	Position des Universalgebers	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE On/On+
_144_Pos_Sensor_HTL	Position des HTL-Gebers	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE On/On+
_145_Actual_pos	Istposition	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_146_Actual_ref_pos	Aktuelle Sollposition	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_147_Actual_pos_diff	Positionsdifferenz zwischen Soll- und Istwert	0.001 Umdrehung	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E On/On+
_150_Direct_dc_link_voltage	aktuelle Zwischenkreisspannung (ungefiltert)	100V = 100	INT	R	SK 5xxP On/On+

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_151_Direct_torque_current	aktueller Momentstrom (ungefiltert)	ST: Wert := INT_TO_DINT(_151_Direct_torque_current) * INT_TO_DINT(_153_Factor_InFu_B) / DINT#819  1A = 100	INT	R	SK 5xxP On/On+
_152_Direct_field_current	aktueller Feldstrom (ungefiltert)	AWL: LD _153_Factor_InFu_B INT_TO_DINT ST Num_InFu  LD _151_Direct_torque_current INT_TO_DINT MUL Num_InFu DIV DINT#819 ST Wert  1A = 100	INT	R	SK 5xxP On/On+
_153_Factor_InFu_B	Faktor für die Berechnung des aktuellen Moment- oder Feldstroms		INT	R	SK 5xxP On/On+

### 3.5.6 PLC Fehler

Über die User Error Flags können aus dem PLC Programm heraus die Gerätefehler E23.0 bis E24.7 gesetzt werden.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_110_ErrorFlags	Erzeugt Benutzerfehler im Gerät	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	alle
_111_ErrorFlags_ext	Erzeugt Benutzerfehler im Gerät	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	alle

### 3.5.7 PLC Parameter

Über diese Gruppen von Prozessdaten kann direkt auf die PLC Parameter P355, P356 und P360 zugegriffen werden.

Name	Funktion	Normierung	Typ	Zugriff	Gerät
_115_PLC_P355_1	PLC INT Parameter P355 [-01]	-	INT	R	alle
_116_PLC_P355_2	PLC INT Parameter P355 [-02]	-	INT	R	alle
_117_PLC_P355_3	PLC INT Parameter P355 [-03]	-	INT	R	alle
_118_PLC_P355_4	PLC INT Parameter P355 [-04]	-	INT	R	alle
_119_PLC_P355_5	PLC INT Parameter P355 [-05]	-	INT	R	alle
_120_PLC_P355_6	PLC INT Parameter P355 [-06]	-	INT	R	alle
_121_PLC_P355_7	PLC INT Parameter P355 [-07]	-	INT	R	alle
_122_PLC_P355_8	PLC INT Parameter P355 [-08]	-	INT	R	alle
_123_PLC_P355_9	PLC INT Parameter P355 [-09]	-	INT	R	alle
_124_PLC_P355_10	PLC INT Parameter P355 [-10]	-	INT	R	alle
_125_PLC_P356_1	PLC LONG Parameter P356 [-01]	-	DINT	R	alle
_126_PLC_P356_2	PLC LONG Parameter P356 [-02]	-	DINT	R	alle
_127_PLC_P356_3	PLC LONG Parameter P356 [-03]	-	DINT	R	alle
_128_PLC_P356_4	PLC LONG Parameter P356 [-04]	-	DINT	R	alle
_129_PLC_P356_5	PLC LONG Parameter P356 [-05]	-	DINT	R	alle
_130_PLC_P360_1	PLC Anzeige Parameter P360[-01]	-	DINT	R/W	alle
_131_PLC_P360_2	PLC Anzeige Parameter P360[-02]	-	DINT	R/W	alle
_132_PLC_P360_3	PLC Anzeige Parameter P360[-03]	-	DINT	R/W	alle
_133_PLC_P360_4	PLC Anzeige Parameter P360[-04]	-	DINT	R/W	alle
_134_PLC_P360_5	PLC Anzeige Parameter	-	DINT	R/W	alle

Name	Funktion	Normierung	Typ	Zugriff	Gerät
	P360[-05]				
_135_PLC_Scope_Int_1	PLC Scope Anzeigewert 1	-	INT	R/W	alle
_136_PLC_Scope_Int_2	PLC Scope Anzeigewert 2	-	INT	R/W	alle
_137_PLC_Scope_Int_3	PLC Scope Anzeigewert 3	-	INT	R/W	alle
_138_PLC_Scope_Int_4	PLC Scope Anzeigewert 4	-	INT	R/W	alle
_139_PLC_Scope_Bool_1	PLC Scope Anzeigewert 5	-	INT	R/W	alle
_140_PLC_Scope_Bool_2	PLC Scope Anzeigewert 6	-	INT	R/W	alle

## 3.6 Sprachen

### 3.6.1 Anweisungsliste (AWL / IL)

#### 3.6.1.1 Allgemein

##### Datentypen

Die PLC unterstützt die nachfolgend aufgeführten Datentypen.

Name	Benötigter Speicherplatz	Wertebereich
BOOL	1 Bit	0 bis 1
BYTE	1 Byte	0 bis 255
INT	2 Byte	-32768 bis 32767
DINT	4 Byte	-2.147.483.648 bis 2.147.483.647
LABEL_ADDRESSES	2 Byte	Sprungmarke

##### Literale

Zur besseren Übersicht ist es möglich Konstanten aller Datentypen in verschiedenen Darstellungsformen einzugeben. In nachfolgender Tabelle ist eine Übersicht über alle möglichen Varianten enthalten.

Literal	Beispiel	Zahl in dezimaler Darstellung
<b>Bool</b>	FALSE	0
	TRUE	1
	BOOL#0	0
	BOOL#1	1
<b>Dual (Basis 2)</b>	2#01011111	95
	2#0011_0011	51
	BYTE#2#00001111	15
	BYTE#2#0001_1111	31
<b>Oktal (Basis 8)</b>	8#0571	377
	8#05_71	377
	BYTE#8#10	8
	BYTE#8#111	73
	BYTE#8#1_11	73
<b>Hexadezimal (Basis 16)</b>	16#FFFF	-1
	16#0001_FFFF	131071
	INT#16#1000	4096
	DINT#16#0010_2030	1056816
<b>Ganzzahlige (Basis 10)</b>	10	10
	-10	-10
	10_000	10000
	INT#12	12
	DINT#-100000	-100000
<b>Zeit</b>	TIME#10s50ms	10,050 Sekunden
	T#5s500ms	5,5 Sekunden
	TIME#5.2s	5,2 Sekunden
	TIME#5D10H15M	5Tage+10Stunden+15Minuten
	T#1D2H30M20S	1Tag+2Stunden+30Minuten+20Sekunden

## Kommentare

Für die spätere Lesbarkeit des PLC – Programmes ist es empfehlenswert Programmabschnitte mit Erklärungen zu versehen. Diese Kommentare werden im Anwenderprogramm beginnend durch die Zeichenfolge „(\*“ und abschließend durch „\*)“ gemäß nachfolgenden Beispielen gekennzeichnet.

```
(* Kommentar über einem Programmblock *)
LD 100 (* Kommentar hinter einem Befehl *)
ADD 20
```

## Sprungmarke

Mit Hilfe der Operatoren JMP, JMPC oder JMPCN können ganze Programmteile übersprungen werden. Als Zieladresse wird eine Sprungmarke angegeben. Sie kann mit Ausnahme von Umlauten und „ß“ alle Buchstaben, die Zahlen 0 bis 9 und Unterstriche enthalten, andere Zeichen sind nicht zulässig. Über einen Doppelpunkt wird die Sprungmarke abgeschlossen. Sie kann für sich alleine stehen. Es kann sich in derselben Zeile, hinter der Sprungmarke, auch noch ein weiterer Befehl befinden.

Mögliche Varianten könnten wie folgt aussehen:

### Beispiel:

```
Sprungmarke:
LD 20

Das_Ist_eine_Sprungmarke:
ADD 10

MainLoop: LD 1000
```

Eine weitere Variant ist die Übergabe einer Sprungmarke als Variable. Dies Variable muss in der Variablen-tabelle als Typ LABEL\_ADDRESS definiert werden, dann können in diese Variable Sprungmarken geladen werden. Hierüber lassen sich sehr einfach Zustandsmaschinen erzeugen, siehe unten

### Beispiel:

```
LD FirstTime
JMPC AfterFirstTime
(* Die Labeladresse muss zu Beginn initialisiert werden. *)
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
Ende:
```



### Funktionsaufrufe

Der Editor unterstützt eine Form von Funktionsaufrufen. In den nachfolgenden Varianten wird die Funktion CTD über die Instanz I\_CTD aufgerufen. Die Ergebnisse werden in Variablen gespeichert. Die Bedeutung der im Folgenden verwendeten Funktionen ist weiter hinten im Handbuch erläutert.

#### Beispiel:

```
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
```

### Bitweiser Zugriff auf Variablen

Für den Zugriff auf ein Bit aus einer Variablen oder Prozessvariablen, ist eine vereinfachte Schreibweise möglich.

Befehl	Bedeutung
LD Var1.0	lädt das Bit 0 von Var1 ins AE
ST Var1.7	speichert den AE auf das Bit 7 von Var1
EQ Var1.4	vergleicht das AE mit dem Bit4 von Var1

### 3.6.2 Strukturierter Text (ST)

Der Strukturierte Text besteht aus einer Reihe von Anweisungen, die wie in Hochsprachen bedingt ("IF..THEN..ELSE) oder in Schleifen (WHILE..DO) ausgeführt werden können.

**Beispiel:**

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

#### 3.6.2.1 Allgemein

##### Datentypen in ST

Die PLC unterstützt die nachfolgend aufgeführten Datentypen.

Name	Benötigter Speicherplatz	Wertebereich
BOOL	1 Bit	0 bis 1
BYTE	1 Byte	0 bis 255
INT	2 Byte	-32768 bis 32767
DINT	4 Byte	-2.147.483.648 bis 2.147.483.647

#### Information

Bei Zahlen ist es sinnvoll den Datentyp mit anzugeben, um ein effizientes PLC Programm zu erzeugen z.B.: VarInt := INT#-32768, VarDINT := DINT#-2147483648.

##### Zuweisungsoperator

Auf der linken Seite einer Zuweisung steht ein Operand (Variable, Adresse), dem der Wert des Ausdrucks auf der rechten Seite zugewiesen wird mit dem Zuweisungsoperator ":=".

**Beispiel:**

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

Nach Ausführung dieser Zeile hat Var1 den zehnfachen Wert von Var2.

### Aufruf von Funktionsblöcken in ST

Ein Funktionsblock in ST wird aufgerufen, indem man den Namen der Instanz des Funktionsblocks schreibt und anschließend in Klammer die gewünschten Werte den Parametern zuweist. Im folgenden Beispiel wird ein Timer aufgerufen mit Zuweisungen für dessen Parameter IN und PT. Anschließend wird die Ergebnisvariable Q an die Variable A zugewiesen.

Die Ergebnisvariable wird wie in AWL mit dem Namen des Funktionsblocks, einem anschließenden Punkt und dem Namen der Variablen angesprochen.

#### Beispiel:

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

### Auswertung von Ausdrücken

Die Auswertung eines Ausdrucks erfolgt durch Abarbeitung der Operatoren nach bestimmten Bindungsregeln. Der Operator mit der stärksten Bindung wird zuerst abgearbeitet, dann der Operator mit der nächststärkeren Bindung, usw., bis alle Operatoren abgearbeitet sind. Operatoren mit gleicher Bindungsstärke werden von links nach rechts abgearbeitet.

Nachfolgend finden Sie eine Tabelle der ST-Operatoren in der Ordnung ihrer Bindungsstärke:

Operation	Symbol	Bindungsstärke
Einklammern	(Ausdruck)	Stärkste Bindung
Funktionsaufruf	Funktionsname (Parameterliste)	
Negieren Komplementbildung	NOT	
Multiplizieren Dividieren Modulo AND	* / MOD AND	
Addieren Subtrahieren OR XOR	+ - OR XOR	
Vergleiche Gleichheit Ungleichheit	<, >, <=, >= = <>	Schwächste Bindung

### 3.6.2.2 Anweisungen

#### Return

Die RETURN-Anweisung kann man verwenden, um an des Ende des Programms zu springen, beispielsweise abhängig von einer Bedingung.

#### IF

Mit der IF-Anweisung kann man eine Bedingung prüfen und abhängig von dieser Bedingung Anweisungen ausführen.

#### Syntax:

```
IF <Boolscher_Ausdruck1> THEN
  <IF_Anweisungen>
ELSIF <Boolscher_Ausdruck2> THEN
  <ELSIF_Anweisungen1>
ELSIF <Boolscher_Ausdruck n> THEN
  <ELSIF_Anweisungen n-1>
ELSE
  <ELSE_Anweisungen>}
END_IF;
```

Der Teil in geschweiften Klammern {} ist optional.

Wenn <Boolscher\_Ausdruck1> TRUE ergibt, dann werden nur die <IF\_Anweisungen> ausgeführt und keine der weiteren Anweisungen. Andernfalls werden die Boolschen Ausdrücke, beginnend mit <Boolscher\_Ausdruck2> der Reihe nach ausgewertet, bis einer der Ausdrücke TRUE ergibt. Dann werden nur die Anweisungen nach diesem Boolschen Ausdruck und vor dem nächsten ELSE oder ELSIF ausgewertet. Wenn keine der Boolschen Ausdrücke TRUE ergibt, dann werden ausschließlich die <ELSE\_Anweisungen> ausgewertet.

#### Beispiel:

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

## CASE

Mit der CASE-Anweisung kann man mehrere bedingte Anweisungen mit derselben Bedingungsvariablen in ein Konstrukt zusammenfassen.

### Syntax:

```
CASE <Var1> OF
  <Wert 1>: <Anweisung 1>
  <Wert 2>: <Anweisung 2>
  <Wert3, Wert4, Wert5: <Anweisung 3>
  <Wert6 .. Wert10 : <Anweisung 4>
  ...
  <Wert n>: <Anweisung n>
ELSE <ELSE-Anweisung>
END_CASE;
```

Eine CASE-Anweisung wird nach folgendem Schema abgearbeitet:

- Wenn die Variable in <Var1> den Wert <Wert i> hat, dann wird die Anweisung <Anweisung i> ausgeführt
- Hat <Var 1> keinen der angegebenen Werte, dann wird die <ELSE-Anweisung> ausgeführt.
- Wenn für mehrere Werte der Variablen, dieselbe Anweisung auszuführen ist, dann kann man diese Werte mit Kommatas getrennt hintereinander schreiben, und damit die gemeinsame Anweisung bedingen.
- Wenn für einen Wertebereich der Variablen, dieselbe Anweisung auszuführen ist, dann kann man den Anfangs- und Endwert getrennt durch zwei Punkte hintereinanderschreiben, und damit die gemeinsame Anweisung bedingen.

### Beispiel:

```
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;
```

## FOR- Schleife

Mit der FOR-Schleife kann man wiederholte Vorgänge programmieren.

### Syntax:

```
FOR <INT_Var> := <INIT_WERT> TO <END_WERT> {BY <Schrittgröße>} DO
  <Anweisungen>
END_FOR;
```

Der Teil in geschweiften Klammern {} ist optional. Die <Anweisungen> werden solange ausgeführt, solange der Zähler <INT\_Var> nicht größer als der <END\_WERT> ist. Dies wird vor der Ausführung der <Anweisungen> überprüft, so dass die <Anweisungen> niemals ausgeführt werden, wenn <INIT\_WERT> größer als <END\_WERT> ist. Immer, wenn <Anweisungen> ausgeführt worden ist, wird <INT\_Var> um <Schrittgröße> erhöht. Die Schrittgröße kann jeden Integerwert haben. Fehlt sie wird diese auf 1 gesetzt. Die Schleife muss also terminieren, da <INT\_Var> nur größer wird.

### Beispiel:

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

## REPEAT- Schleife

Die REPEAT-Schleife unterscheidet sich von den WHILE-Schleifen dadurch, dass die Abbruchbedingung erst nach dem Ausführen der Schleife überprüft wird. Das hat zur Folge, dass die Schleife mindestens einmal durchlaufen wird, egal wie die Abbruchbedingung lautet.

### Syntax:

```
REPEAT
  <Anweisungen>
UNTIL <Boolescher Ausdruck>
END_REPEAT;
```

Die <Anweisungen> werden solange ausgeführt, bis <Boolescher Ausdruck> TRUE ergibt. Wenn <Boolescher Ausdruck> bereits bei der ersten Auswertung TRUE ergibt, dann werden <Anweisungen> genau einmal ausgeführt. Wenn <Boolescher\_Ausdruck> niemals den Wert TRUE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

---

## Information

Der Programmierer muss selbst dafür sorgen, dass keine Endlosschleife entsteht, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler hoch- oder runterzählt.

---

### Beispiel:

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

## WHILE- Schleife

Die WHILE-Schleife kann benutzt werden wie die FOR-Schleife, mit dem Unterschied, dass die Abbruchbedingung ein beliebiger boolescher Ausdruck sein kann. Das heißt, man gibt eine Bedingung an, die, wenn sie zutrifft, die Ausführung der Schleife zur Folge hat.

### Syntax:

```
WHILE <Boolescher Ausdruck> DO
  <Anweisungen>
END_WHILE;
```

Die <Anweisungen> werden solange ausgeführt, bis <Boolescher Ausdruck> FALSE ergibt. Wenn <Boolescher Ausdruck> bereits während der ersten Ausführung FALSE ergibt, dann werden <Anweisungen> genau einmal ausgeführt. Wenn <Boolescher\_Ausdruck> niemals den Wert FALSE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

---

## Information

Der Programmierer muss selbst dafür sorgen, dass keine Endlosschleife entsteht, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler hoch- oder runterzählt.

---

### Beispiel:

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

### Exit

Wenn die EXIT-Anweisung in einer FOR-, WHILE- oder REPEAT-Schleife vorkommt, dann wird die innerste Schleife beendet, ungeachtet der Abbruchbedingung.

## 3.7 Sprünge

### 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
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X

Unbedingter Sprung zu einer Sprungmarke.

#### Beispiel in AWL:

```
JMP NextStep (* Unbedingter Sprung zu 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
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X

Bedingter Sprung (Jump Conditional) zu einer Sprungmarke. Ist das AE = TRUE dann springt die Anweisung JMPC zur angegebenen Sprungmarke.

#### Beispiel in AWL:

```
LD 10
JMPC NextStep (* AE = TRUE à Programm springt *)
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
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X

Bedingter Sprung (Jump Conditional) zu einer Sprungmarke. JMPCN springt, wenn das AE Register = FALSE ist. Ansonsten wird das Programm mit der nachfolgenden Anweisung fortgesetzt.

#### Beispiel in AWL:

```
LD 10
JMPCN NextStep (* AE = TRUE à Programm springt nicht *)
ADD 1

NextStep:
ST Value1
```



## 3.8 Typkonvertierung

### 3.8.1 BOOL\_TO\_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp	X			

Konvertiert den Datentyp AE von BOOL zu BYTE. Ist das AE gleich FALSE, dann wird der Akku auf 0 konvertiert. Ist das AE gleich TRUE, dann wird der Akku auf 1 konvertiert.

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.8.2 BYTE\_TO\_BOOL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp		X		

Konvertiert den Datentyp von BYTE zu BOOL. Solange das BYTE ungleich Null ist, gibt es immer ein TRUE als Konvertierungsergebnis.

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.8.3 BYTE\_TO\_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>		X		

Konvertiert den Datentyp von BYTE zu INT. Das BYTE wird in den Low Teil des INT hineinkopiert und der High Teil vom INT wird 0 gesetzt.

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.8.4 DINT\_TO\_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
<b>Verfügbarkeit</b>	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
<b>Datentyp</b>				X

Konvertiert den Datentyp von DINT zu INT. Dabei wird der High Teil vom DINT Wert nicht mit übernommen.

#### Beispiel in AWL:

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

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

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

#### Beispiel in ST:

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

### 3.8.5 INT\_TO\_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp			X	

Konvertiert den Datentyp von INT zu BYTE. Dabei wird der High Teil vom INT Wert nicht mit übernommen. Vorzeichen gehen verloren, da der Typ BYTE vorzeichenlos ist.

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.8.6 INT\_TO\_DINT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	On/On+	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Verfügbarkeit	X	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Datentyp			X	

Konvertiert den Datentyp von INT zu DINT. Das INT wird in den Low Teil des DINT hineinkopiert und der High Teil vom DINT wird 0 gesetzt.

#### Beispiel in AWL:

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

#### Beispiel in ST:

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

### 3.9 PLC Störmeldungen

Störmeldungen führen zum Abschalten des Gerätes, um einen Gerätedefekt zu verhindern. Bei PLC Störmeldungen wird die Abarbeitung der PLC gestoppt und die PLC geht in den Zustand „PLC-Error“. Bei anderen Störmeldungen läuft die PLC weiter. Nach einer Quittierung des Fehlers startet die PLC wieder automatisch.

**Beim PLC User Fault 23.X und 24.X läuft die PLC weiter!**

Anzeige in der SimpleBox		Störung Text in der ParameterBox	Ursache Abhilfe
Gruppe	Detail in P700[-01]/ P701		
E022	22.0	Kein PLC-Programm	Die PLC wurde gestartet es befindet sich jedoch kein PLC-Programm im FU - PLC-Programm in das Gerät laden
	22.1	PLC-Programm ist fehlerhaft	Die Checksummen Prüfung über das PLC-Programm ergab einen Fehler. - Gerät neu starten (Power ON) und wieder versuchen - Alternative, PLC-Programm neu laden
	22.2	Falsche Sprungadresse	Programmfehler, Verhalten wie im Fehler 22.1
	22.3	Stack Überlauf	Es wurden in der Laufzeit des Programm mehr als 6 Klammerebenen geöffnet - Programm auf Laufzeitfehler überprüfen
	22.4	Max. PLC-Zyklen überschritten	Die angegebene max. Zykluszeit des PLC-Programmes wurde überschritten - Zykluszeit anpassen oder Programm überprüfen
	22.5	Unbekannter Befehlscode	Ein im Programm vorhandener Befehlscode kann nicht ausgeführt werden, da er unbekannt ist - Programmfehler, Verhalten wie im Fehler 22.1 - Version der PLC und die Version von NORDCON passen nicht zusammen
	22.6	PLC-Schreibzugriff	Während eines laufenden PLC-Programmes wurde der Programminhalt verändert
	22.9	PLC-Sammelfehler	Die Fehlerursache kann nicht genau aufgelöst werden - Verhalten wie im Fehler 22.1
E023/ E024	23.0 bis 23.7	PLC User Fault 1 bis 8	Dieser Fehler kann durch das PLC-Programm ausgelöst werden, um Probleme im Ablauf des PLC-Programm nach außen zu signalisieren. Die Auslösung erfolgt über das Beschreiben der Prozessvariable „ErrorFlags“.
	24.0 bis 24.7	PLC User Fault 9 bis 16	

## 4 Parameter

Die für die PLC-Funktionalität relevanten Geräteparameter sind ausführlich im Handbuch des betreffenden Frequenzumrichters bzw. Motorstarters beschrieben.

## 5 Anhang

### 5.1 Service- und Inbetriebnahmehinweise

Bei Problemen, z. B. während der Inbetriebnahme, nehmen Sie Kontakt mit unserem Service auf:

☎ +49 4532 289-2125

Unser Service steht Ihnen rund um die Uhr (24 h/7 Tage) zur Verfügung und kann Ihnen am besten helfen, wenn Sie folgende Informationen vom Gerät und dessen Zubehör bereithalten:

- Typenbezeichnung,
- Seriennummer,
- Firmwareversion.

### 5.2 Dokumente und Software

Dokumente und Software können Sie von unserer Internetseite [www.nord.com](http://www.nord.com) herunterladen.

#### Mitgeltende und weiterführende Dokumente

Dokumentation	Inhalt
<a href="#">BU 0155</a>	Handbuch für Feldverteiler Motorstarter NORDAC <i>LINK SK 180E / SK 190E</i>
<a href="#">BU 0180</a>	Handbuch für Frequenzumrichter NORDAC <i>BASE SK 180E / SK 190E</i>
<a href="#">BU 0200</a>	Handbuch für Frequenzumrichter NORDAC <i>FLEX SK 200E .. SK 235E</i>
<a href="#">BU 0250</a>	Handbuch für Feldverteiler Frequenzumrichter NORDAC <i>LINK SK 250E-FDS .. SK 280E-FDS</i>
<a href="#">BU 0500</a>	Handbuch für Frequenzumrichter NORDAC <i>PRO SK 500E .. SK 535E</i>
<a href="#">BU 0505</a>	Handbuch für Frequenzumrichter NORDAC <i>PRO SK 540E .. SK 545E</i>
<a href="#">BU 0600</a>	Handbuch für Frequenzumrichter NORDAC <i>PRO SK 500P .. SK 550P</i>
<a href="#">BU 0800</a>	Handbuch für Frequenzumrichter NORDAC <i>ON/ON+ SK 300P</i>
<a href="#">BU 0000</a>	Handbuch zum Umgang mit der NORDCON-Software
<a href="#">BU 0040</a>	Handbuch zum Umgang mit den NORD-Parametrierboxen

#### Software

Software	Beschreibung
<a href="#">NORDCON</a>	Parametrier- und Diagnosesoftware

### 5.3 Abkürzungen

- **AE** Aktuelles Ergebnis
- **AIN** Analogeingang
- **AOUT** Analogausgang
- **AWL** Anwendungsliste (auch IL)
- **COB-ID** Communication Objekt Identifier
- **DI / DIN** Digitaleingang
- **DO / DOUT** Digitalausgang
- **E/A bzw. I/O** Ein- / Ausgang
- **EEPROM** Nicht flüchtiger Speicher
- **EMV** Elektromagnetische Verträglichkeit
- **FB** Funktionsblock
- **FU** Frequenzumrichter
- **HSW** Hauptsollwert
- **IL** Instruction List (siehe auch AWL)
- **ISD** Feldstrom (Stromvektorregelung)
- **LED** Leuchtdiode
- **MC** Motion Control
- **NSW** Nebensollwert
- **P** Parametersatzabhängiger Parameter, d.h. ein Parameter, dem in jedem der 4 Parametersätze des Gerätes unterschiedliche Funktionen bzw. Werte zugewiesen werden können.
- **P-BOX** ParameterBox
- **PDO** Prozess Daten Objekt
- **PLC** SPS (Speicher Programmierbare Steuerung)
- **S** Supervisor Parameter, d.h. Ein Parameter der nur sichtbar wird, wenn der korrekte Supervisor Code in Parameter **P003** eingetragen ist
- **SW** Softwareversion (Siehe Parameter **P707**)
- **STW** Steuerwort
- **ZSW** Zustandswort (Statuswort)

## Stichwortverzeichnis

<b>B</b>		CTUD.....	63
Bestimmungsgemäße Verwendung .....	9	Datentypen .....	142
<b>D</b>		Datentypen in ST.....	146
Dokumente		Datenverarbeitung über Akku .....	14
mitgeltend .....	158	Debugging .....	23
<b>E</b>		DINT_TO_INT .....	154
Elektrofachkraft.....	10	DIV.....	94
<b>P</b>		DIV( .....	94
PLC.....	11	Editor .....	18
ABS .....	92	Ein- und Ausgänge.....	116
ACOS .....	98	Eingabefenster .....	20
ADD.....	93	Einzelschritt .....	24
ADD( .....	93	Elektronisches Getriebe mit Fliegender Säge .....	15, 35
AND.....	101	EQ .....	113
AND( .....	101	Erweiterte mathematische Operatoren .....	98
ANDN .....	102	Exit.....	151
ANDN(.....	102	EXP .....	99
Anweisungsliste (AWL / IL).....	142	F_TRIG.....	65
Arithmetische Operatoren .....	92	FB_FunctionCurve .....	86
ASIN.....	98	FB_PIDT1.....	87
ATAN.....	98	FB_ResetPostion .....	89
Aufruf von Funktionsblöcken in ST .....	147	FB_Capture .....	81
Auswertung von Ausdrücken .....	147	FB_DinCounter.....	84
Beobachtungspunkte .....	23	FB_DINTToPBOX .....	76
Bit Operatoren.....	101	FB_FlyingSaw .....	36
Bitweiser Zugriff auf Variablen.....	145	FB_Gearing .....	38
BOOL_TO_BYTE.....	153	FB_NMT .....	27
Bus Soll- und Istwerte .....	128	FB_PDConfig.....	28
BYTE_TO_BOOL.....	153	FB_PDOReceive .....	31
BYTE_TO_INT .....	154	FB_PDOSend.....	33
CANopen Kommunikation.....	16	FB_ReadTrace .....	71
CASE .....	149	FB_STRINGToPBOX.....	79
ControlBox .....	15	FB_Weigh.....	90
ControlBox und ParameterBox .....	133	FB_WriteTrace .....	73
COS .....	98	Fehler .....	139
CTD.....	61	FOR- Schleife .....	150
CTU.....	62	Funktionsaufrufe.....	145



Funktionsblöcke .....	26	Meldungsfenster .....	21
Funktionsumfang .....	15	MIN .....	95
GE .....	113	MOD .....	96
GT .....	114	MOD( .....	96
Haltepunkte .....	23	Motion Control Lib .....	15
IF .....	148	MUL .....	96
Infoparameter .....	134	MUL( .....	96
INT_TO_BYTE .....	155	MUX .....	97
INT_TO_DINT .....	155	NE .....	115
JMP .....	152	NOT .....	103
JMPC .....	152	Operatoren .....	92
JMPCN .....	152	OR .....	104
Kommentare .....	144	OR( .....	104
Konfiguration .....	25	ORN .....	105
Lade- und Speicheroperatoren .....	111	ORN( .....	105
Laden, Speichern & Drucken .....	17	Parameter .....	140
LD .....	111	ParameterBox .....	15
LDN .....	111	Programm Task .....	14
LE .....	114	Programm zum Gerät übertragen .....	22
LIMIT .....	94	Prozessabbild .....	13
Literale .....	142	Prozessregler .....	16
LN .....	99	Prozesswerte .....	116
LOG .....	100	R .....	107
LT .....	115	R_TRIG .....	65
MAX .....	95	REPEAT- Schleife .....	150
MC_MoveAbsolute .....	47	Return .....	148
MC_WriteParameter_16 .....	60	ROL .....	106
MC_WriteParameter_32 .....	60	ROR .....	106
MC_Control .....	41	RS Flip Flop .....	66
MC_Control_MS .....	43	S .....	107
MC_Home .....	44	SHL .....	107
SK5xxP .....	45	SHR .....	108
MC_MoveAdditive .....	49	SIN .....	98
MC_MoveRelative .....	50	Soll- und Istwerte .....	124
MC_MoveVelocity .....	51	Sollwert Verarbeitung .....	14
MC_Power .....	53	Speicher .....	13
MC_ReadActualPos .....	55	Spezifikation .....	12
MC_ReadParameter .....	56	Sprachen .....	142
MC_ReadStatus .....	57	Sprünge .....	152
MC_Reset .....	58	Sprungmarke .....	144
MC_Stop .....	59	SQRT .....	100

SR Flip Flop .....	67	Variablen und FB Deklaration .....	19
ST .....	112	Vergleichs Operatoren .....	113
Standard Funktionsblöcke .....	61	Visualisierung .....	15
STN .....	112	Visualisierung ParameterBox .....	75
Störmeldungen .....	156	Watch- & Breakpoint Anzeigefenster .....	21
Strukturierter Text (ST) .....	146	WHILE- Schleife .....	151
SUB .....	97	XOR .....	109
SUB( .....	97	XOR( .....	109
TAN .....	98	XORN .....	110
TOF .....	68	XORN( .....	110
TON .....	69	Zuweisungsoperator .....	146
TP .....	70	<b>S</b>	
Typkonvertierung .....	153	Sicherheitshinweise .....	10
Überblick Visualisierung .....	75	Software .....	158



**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](http://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](mailto:info@nord.com), [www.nord.com](http://www.nord.com)

**Member of the NORD DRIVESYSTEMS Group**

