PMC Basis Projekt mit CoDeSys

Produkt
Typ: PMC
Name: PMCprimo Drive
Hersteller: Pilz GmbH & Co. KG, Sichere Automation

Dokument
Version: 03
Ausgabestand: 21 März 2011

Application Note – Nr. 1001320_DE_03

the spirit of safety

Lebenslauf

Version	Datum	Änderung/Erweiterung	Kapitel
01	2008-08-14	Erstellung	alle
02	2010-07-19	Einrichten für Internet-Freigabe	nur logistische Daten
03	2011-03-21	Anpassungen für die Veröffentlichung	nur redaktionell

Haftungsausschluss

Wir haben unsere technische Application Note sehr sorgfältig zusammengestellt. Es enthält Informationen über unser Unternehmen sowie über unsere Produkte. Alle Angaben haben wir nach dem heutigen Stand der Technik und bestem Wissen und Gewissen gemacht. Dennoch können wir für die Richtigkeit und Vollständigkeit der Angaben, sofern uns nicht der Vorwurf grober Fahrlässigkeit trifft, keine Haftung übernehmen, da sich trotz aller Sorgfalt Fehler nicht vollständig vermeiden lassen. Insbesondere sind alle Angaben zu geltenden Normen, zu sicherheitstechnischen Einstufungen und zum Zeitverhalten vorläufig. Die Angaben in diesem Katalog haben nicht die rechtliche Qualität von Zusicherungen oder zugesicherten Eigenschaften. Für Hinweise auf Unstimmigkeiten sind wir dankbar.

März 2011

Support

Technische Hilfe rund um die Uhr!
Technische Unterstützung von Pilz erhalten Sie rund um die Uhr.
Diesen Service stellen wir Ihnen über unsere Geschäftszeiten hinaus kostenlos zur Verfügung

Amerika
- Brasilien
  +55 11 8245-8267
- Mexiko
  +52 55 5572 1300
- USA
  +1 877-PILZUSA (745-9872)

Asien
- China
  +86 21 62494658-216
- Japan
  +81 45 471-2281
- Korea
  +82 2 2263 9540

Australien
- Australien
  +61 3 95446300

Europa
- Belgien, Luxemburg
  +32 9 3217575
- Deutschland
  +49 711 3409-444
- England
  +44 1536 462203
- Frankreich
  +33 3 88104000
- Irland
  +353 21 4804983
- Italien
  +39 031 789511
- Niederlande
  +31 347 320477
- Österreich
  +43 1 7986263-0
- Schweiz
  +41 62 88979-30
- Skandinavien
  +45 74436332
- Spanien
  +34 938497433
- Türkei
  +90 216 5775552

Unsere internationale Hotline erreichen Sie unter:

+49 711 3409-444 oder **mailto:support@pilz.com**

Pilz GmbH & Co. KG
Sichere Automation
Felix-Wankel-Straße 2
73760 Ostfildern, Germany

Telefon: +49 711 3409-0
Telefax: +49 711 3409-133
E-Mail: pilz.gmbh@pilz.de
Internet: www.pilz.com

Inhalt

Abkürzungen

FB	Funktionsbaustein
PMC	Pilz Motion Control
UI	Datentyp UINT (Unsigned INTeger)
USI	Datentyp USINT (Unsigned Short INTeger)

1. Hilfreiche Dokumentation

Das Lesen der unten genannten Dokumentation ist für das Verstehen dieser Application Note wichtig.
Die Verfügbarkeit der verwendeten Software und der sichere Umgang damit werden ebenfalls vorausgesetzt.

1.1. Dokumentation von Pilz GmbH & Co. KG

Nr.	Description	Sachnr.
1	Pilz internationale Internetseite, Download bereich	www.pilz.com
2	PMCprimo Drive2, Installationshandbuch	21 485-DE-xx
3	PMCtendo DD5 / PMCprimo Drive3, Installationshandbuch	21 589-DE-xx
4	Motion Control PMC, Anwenderhandbuch zur Software PMCprimo SoftSPS	21 470-DE-xx
5	Motion Control PMC, Anwenderhandbuch zur Software PMotion	21 472-DE-xx
6	Motion Control PMC, Programmierhandbuch zur Software PMCprimo	21 506-DE-xx
7	Motion Control PMC, Dokumentation zum Datenaustausch über primoDLL	21 526-DE-xx
8	Motion Control PMC, Prospekt zur Bedienung, Steuerung und Bewegung hochdynamischer Antriebe von der Pilz GmbH & Co. KG	www.pilz.com

1.2. Dokumentation aus anderen Quellen

Nr.	Description	Kennung
1	CoDeSys, Programmiersoftware nach IEC 61131-3	www.3s-software.com
2		

HINWEIS:
Wichtige Dokumentationen sind als Dateien im PDF-Format verfügbar:

- auf der CD-ROM „Motion Control Tools“
  Konfigurationssoftware für Motion-Control-Geräte, Bestellnummer 1 802 959
  oder
- als Download im Internet unter: www.pilz.com.

Weiterhin gibt es die Software „CoDeSys Target“ unter der Bestellnummer 8 175 974.
Diese ist zur Freischaltung der CoDeSys-Funktionalität inkl. der Motion Control Tools notwendig.

2. Allgemeine Informationen über das Basis-Projekt

Das Basis-Projekt dient als Startprogramm unter der Programmieroberfläche CoDeSys für eine Motion Control Applikation.

Die Ergänzungen des Projektes auf die jeweiligen Anforderungen einer realen Maschine obliegen dem Anwender. Hierzu gehören unter anderem:

- Die Konfiguration und die Vervielfältigung der einzelnen Programmteile auf die tatsächlich vorhandene Achszahl
- Die Ergänzung der Funktionen zur Kommunikation mit der Maschine z.B. Feldbusanschluss, E/A-Ebene, Bediengeräte
- Bei Nutzung der Soft-SPS als Maschinensteuerung, alle relevanten Ablaufprogramme

Die einzelnen Schritte für die unterschiedlichen Funktionen sind in diesem Dokument beschrieben.
Alle Teile des Projekts sind kommentiert und änderbar für die jeweilige Applikation.

WICHTIG:
Im Basis-Projekt sind die Programmfunktionen zweisprachig kommentiert (DE, EN).

```
bDirection := TRUE ,          (* Vorgabe der Fahrrichtung für Fahren mit konstanter Geschwindigkeit. TRUE = positive Fa
udiPositionBound := 131072 ,  (* siehe SB-Befehl / see the SB command *)
udiAcceleration := 500000 ,   (* Beschleunigung in Inkrementen pro Sekundequadrat // Acceleration in increments per se
udiDeceleration := 500000 ,   (* Bremsrampe in Inkrementen pro Sekundoquadrat  // Deceleration in increments per secc
udiVelocity := 5000 ,         (* Geschwindigkeit in Inkrementen pro Sekunde // Velocity given in increments per second *
uiVelocityMul := 1 ,          (* Multiplikator für Skalierung Geschwindigkeit / velocity multiplied with VelocityMul *)
uiVelocityDiv := 1 ,          (* Divisor für Skallerung Geschwindigkeit / velocity divided with VelocityDiv *)
```

Abb. 1: Zweisprachige Kommentare im Basis-Projekt

WICHTIG:
Das Basis-Projekt erfordert Grundkenntnisse in der SPS-Programmierung und in den Funktionalitäten der PMCprimo.

Das Basis-Projekt wurde erstellt mit der CoDeSys Version 2.3.5 nach IEC 61131-3.
Die Funktionalität wurde getestet mit der PMC Firmware Version 2.009 zusammen mit der „primo_v2_009.lib“ auf den Geräten PMC 16+, PMCprimo Drive2 und PMCprimo Drive3.

3. Beschreibung der Applikation

Das Basis-Projekt beinhaltet für eine Achse 4 Grundfunktionen von PMCprimo
- Initialisierung,
- Konstante Geschwindigkeit (Endlospositionieren),
- Mapping und
- Positionieren.

Beachten:
Für alle nachfolgenden Funktionen gilt grundsätzlich, dass zuerst der gewünschte „OPmode“ (= OperationMode) gewählt werden muss, bevor ein Startsignal angelegt werden darf.

Das Umschalten zwischen „OPmodi“ ist nur bei Start = FALSE möglich!

Im Basis-Projekt sind bereits grundsätzliche Programmierarbeiten für eine einzelne Achse vorhanden. Es sind nicht alle für eine Applikation notwendigen Verknüpfungen integriert, denn viele Verknüpfungen sind von der jeweiligen Anwendung abhängig, z.B. muss nicht bei allen Achsen zunächst eine Referenzpunktfahrt vorgenommen werden um z.B. Mapping auszuführen.

3.1. Bestandteile des Basis-Projekts

In der Abb. 2 sind die Hauptbestandteile des Basis-Projekts dargestellt.

Der Programmbaustein „PLC_PRG“ organisiert die zyklische Programmbearbeitung. Der Baustein darf nicht umbenannt werden und muss ein Programmbaustein sein.

Besonderheit bei Achsen auf PMCprimo-Knoten:
Die Rückmeldung des Achsstatus von einer sich auf einem PMCprimo-Knoten befindlichen Achse wird automatisch gelesen, wenn nach dem Booten mindestens einmal an einer Achse im Array ein Wert in der Variable „*usiNode*“ größer Null gefunden wurde. Diese Rückmeldung liegt im Vergleich zu Achsen auf dem Host-System erst verzögert an.

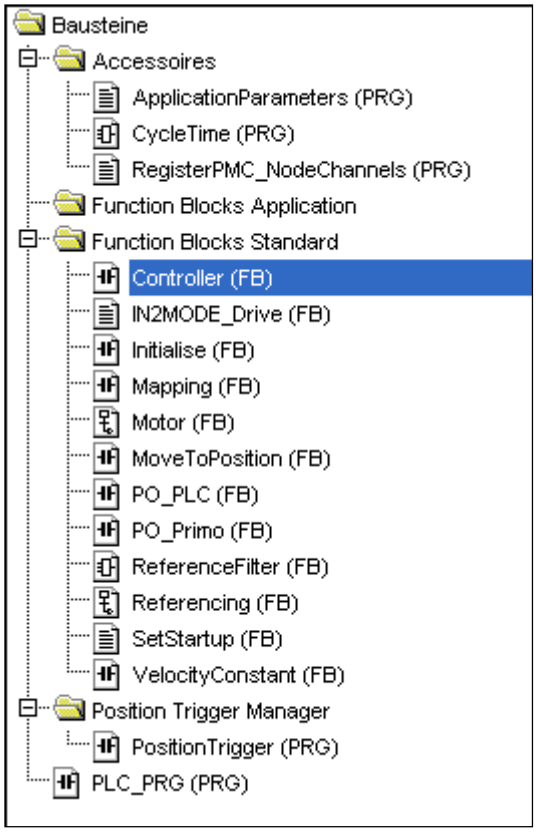


Abb. 2: Übersicht zu den Bestandteilen des Basis-Projekts

3.2. Grundfunktionen des Programms

Wichtigster Teil des Basis-Projekts sind die 5 Arrays in der globalen Variablenliste „Variablen_ChannelParameter“:

- *g_ControllingStatusParameter*: ARRAY [1..20] OF ControllingStatusStruct;
- *g_SetChannelParameter*: ARRAY [1..20] OF PowerOnStruct :=
- *g_AllOpmodeParameters*: ARRAY [1..20] OF OpmodeStruct :=
- *g_ReferencingParameters*: ARRAY [1..20] OF ReferencingStruct :=
- *g_MappingParameters*: ARRAY [1..20] OF MappingStruct :=

Im folgenden Bild sind beispielhaft für mehrere Achsen Namen vergeben. In der Praxis empfiehlt sich eine etwas andere Namensgebung, wie *„g_usiFolientransport“*. Der Präfix „g_usi“ steht für eine globale Variable im Format „USINT“ (Unsigned Short INTeger).
Diese Art der Notation ist eine Empfehlung. Der zugewiesene Variablenwert bestimmt die Array-Adresse der für die Achse bestimmten Parameterwerte.

```
VAR_GLOBAL

(*  die Motornummer bestimmt die Adresse der Variablen in den folgenden ARRAYS
    The motornumber defines the adress of the variables  in the following ARRAYS *)
    g_usiMotorNumber1:      USINT := 1;
    g_usiMotorNumber2:      USINT := 2;
    g_usiMotorNumber3:      USINT := 3;
    g_usiMotorNumber4:      USINT := 4;
    g_usiMotorNumber5:      USINT := 5;
    g_usiMotorNumber6:      USINT := 6;
    g_usiMotorNumber7:      USINT := 7;
    g_usiMotorNumber8:      USINT := 8;
    g_usiMotorNumber9:      USINT := 9;
    g_usiMotorNumber10:     USINT := 10;
    g_usiMotorNumber11:     USINT := 11;
    g_usiMotorNumber12:     USINT := 12;
    g_usiMotorNumber13:     USINT := 13;
    g_usiMotorNumber14:     USINT := 14;
    g_usiMotorNumber15:     USINT := 15;
    g_usiMotorNumber16:     USINT := 16;
    g_usiMotorNumber17:     USINT := 17;
    g_usiMotorNumber18:     USINT := 18;
    g_usiMotorNumber19:     USINT := 19;
    g_usiMotorNumber20:     USINT := 20;

(*  Bei mehr als 20 Achsen sind die folgenden Arrays zu erweitern / If more than 20 primo channels are
```

Abb. 3: Globale Variablenliste, Beispiel für mehrere Achsen

- Im Array *„g_ControllingStatusParameter“* werden für jede Achse folgende Rückmeldungen bereitgestellt:

```
⊟ g_ControllingStatusParameter
  ⊟ g_ControllingStatusParameter[1]
      .bStart = FALSE
      .bPositionStop = FALSE
      .byOperationMode = 0
      .bDone = FALSE
      .bParametersDone = FALSE
      .bMotorOff = FALSE
      .bPositionControlActive = FALSE
      .bInitialiseActive = FALSE
      .bInitialized = FALSE
      .bVelocityConstActive = FALSE
      .bClutchIn = FALSE
      .bSynchronized = FALSE
      .bClutchOut = FALSE
      .bMoveActive = FALSE
      .bPositionReached = FALSE
      .bError = FALSE
      .wErrorNumber = 0
      .usiTestNumber = 0
```

Abb. 4: Parameter für die Überwachung einer Achse

3.3. Operationsmodi des FB „Motor“

Für jede Achse gibt es die nachfolgenden 4 Operationsmodi (= OPmode):

- *OPmode_IN* (OPmode = 1)
  Initialisierung der Achse (= Referenzpunkt suchen)
- *OPmode_VC* (OPmode = 2)
  Fahren der Achse mit konstanter Geschwindigkeit (=Endlospositionierung)
- *OPmode_MAP* (OPmode = 3)
  Achse synchronisieren (= Mapping aktivieren / deaktivieren)
- *OPmode_MOVE* (OPmode = 4)
  Achse positionieren (= Anfahren von absoluten und relativen Zielpositionen)

HINWEISE (siehe auch Kap. 4.3, Seite 23):

- Zuerst muss der gewünschte „OPmode“ (= OperationMode) gewählt, anschließend die Fahrrichtung, Zielposition, Mapname etc. und danach der Start ausgeführt werden.
- Ein Startsignal muss solange anstehen, bis die Fertig- oder die Fehlermeldung kommt.
- Es ist nicht möglich, mehr als einen „OPmode“ zur selben Zeit zu wählen.
- Ein „OPmode“ wird nicht verlassen, solange noch Start anliegt!
- Bei Verlassen eines „OPmode“ schaltet die Reglerfreigabe aus.
- Die Überwachung des „OPmode“, der Rückmeldungen usw. erfolgt im Schritt *„CONTROLLING“*, der zyklisch bearbeitet wird.

3.4. Deklarationsteil des ChannelManagers

- Nach dem BOOTEN der Soft-SPS werden folgende Parameter für die einzelnen Achsen einmalig beschrieben:

```
    g_SetChannelParameter: ARRAY [1..20] OF PowerOnStruct :=

(* g_usiMotorNumber1 =1 *)
        (usiNode := 0,                          (* Angabe der Knotennummer / set node number to local variable 'Node' *)
         usiChannel := 5,                       (* Angabe der Achsnummer auf dem Knoten / set channel number at the node
         usiCanNetwork := 0,                    (* Angabe des CAN-Netzwerks für die Ansteuerung eines tendo DD4/5 / Numb
         usiCanNode_ADDR := 5,                  (* Angabe einer CAN-Adresse 'ADDR' für die Ansteuerung eines tendo DD4/5
         uiProportionalGain := 100,             (* KP-Faktor des Lageregelkreises / see the KP command *)
         uiFeedForwardGain := 3750,             (* KF-Faktor des Lageregelkreises / see the KF command *)
         uiWindow := 100,                       (* siehe SW-Befehl / see the SW command *)
         uiMaxPositionError := 800,             (* siehe SE-Befehl / see the SE command *)
         uiTimeOutEncoder := 500,               (* siehe TO-Befehl / see the TO command *)
         usiFeedbackEncoder := 19,              (* siehe FS-Befehl / see the FS command *)
         usiNumberOfBits := 24,                 (* siehe NB-Befehl / see the NB command *)
         uiVelocityAveragingTime_VT :=10,       (* siehe VT-Befehl / see the VT command *)
         usiEncoderScaling_MS := 0,             (* siehe MS-Befehl / see the MS command *)
         uiEncoderFilterTime_PT := 0,           (* siehe PT-Befehl / see the PT command *)
         diMapBaseAdvance_BA := 0,              (* siehe BA-Befehl / see the BA command *)
         uiMapBaseAdvanceTime_BT:=10,           (* siehe BT-Befehl / see the BT command *)
         usiControlWord := 2#00010000),         (* siehe CW-Befehl / see the CW command *)
```

Abb. 5: Parameter der Achse, die nur einmalig beim Start gesetzt werden

Der Schritt *„SetChannelParameter“* kann nochmals durchlaufen werden, wenn im Zustand Motor Off (Antrieb softwaremäßig disabled) die Variable *„ParameterGesetzt“* einmalig auf FALSE gesetzt wird.

- Diese Parameter werden aus dem „*g_SetChannelParameter*“ entnommen. Die Änderungen dieser Werte während des Betriebs sind nicht wirksam.

Variablenname im Basis-Project	Defaultwert	Befehl / Bedeutung in PMCprimo
usiNode	0	Angabe der Knotennummer
usiChannel	1	Angabe der Achsnummer
usiCanNetwork	0	CAN-Netzwerk, siehe PD-Befehl
usiCanNode_ADDR	0	CAN-Adresse ADDR, siehe PD-Befehl, Param. ADDR von DD4
uiProportionalGain	100	KP-Faktor des Lageregelkreises
uiFeedForwardGain	3750	KF-Faktor des Lageregelkreises
uiWindow	100	siehe Befehl „SW“
uiMaxPositionError	800	siehe Befehl „SE“
uiTimeOutEncoder	500	siehe Befehl „TO“
usiFeedbackEncoder	0	siehe Befehl „FS“
usiNumberOfBits	24	siehe Befehl „NB“
uiVelocityAveragingTime_VT	10	siehe Befehl „VT“
usiEncoderScaling_MS	0	siehe Befehl „MS“
uiEncoderFilterTime_PT	0	siehe Befehl „PT“
diMapBaseAdvance_BA	0	siehe Befehl „BA“
uiMapBaseAdvanceTime_BT	10	siehe Befehl „BT“
usiControlWord	2#00010000	siehe Befehl „CW“, Einstellung „*ControlWord*“

Table 1: Parameter der Achse, die nur einmalig beim Start gesetzt werden

3.5. Grundsätzlicher Programmablauf des FB „Motor“

- Für jede Achse muss der Funktionsbaustein „Motor“ zyklisch aufgerufen werden.
- Als Instanzenname empfiehlt sich der Name der Achse in der Applikation.

Der Funktionsbaustein „Motor“ besteht aus einer Ablaufstruktur für unterschiedliche Betriebszustände:

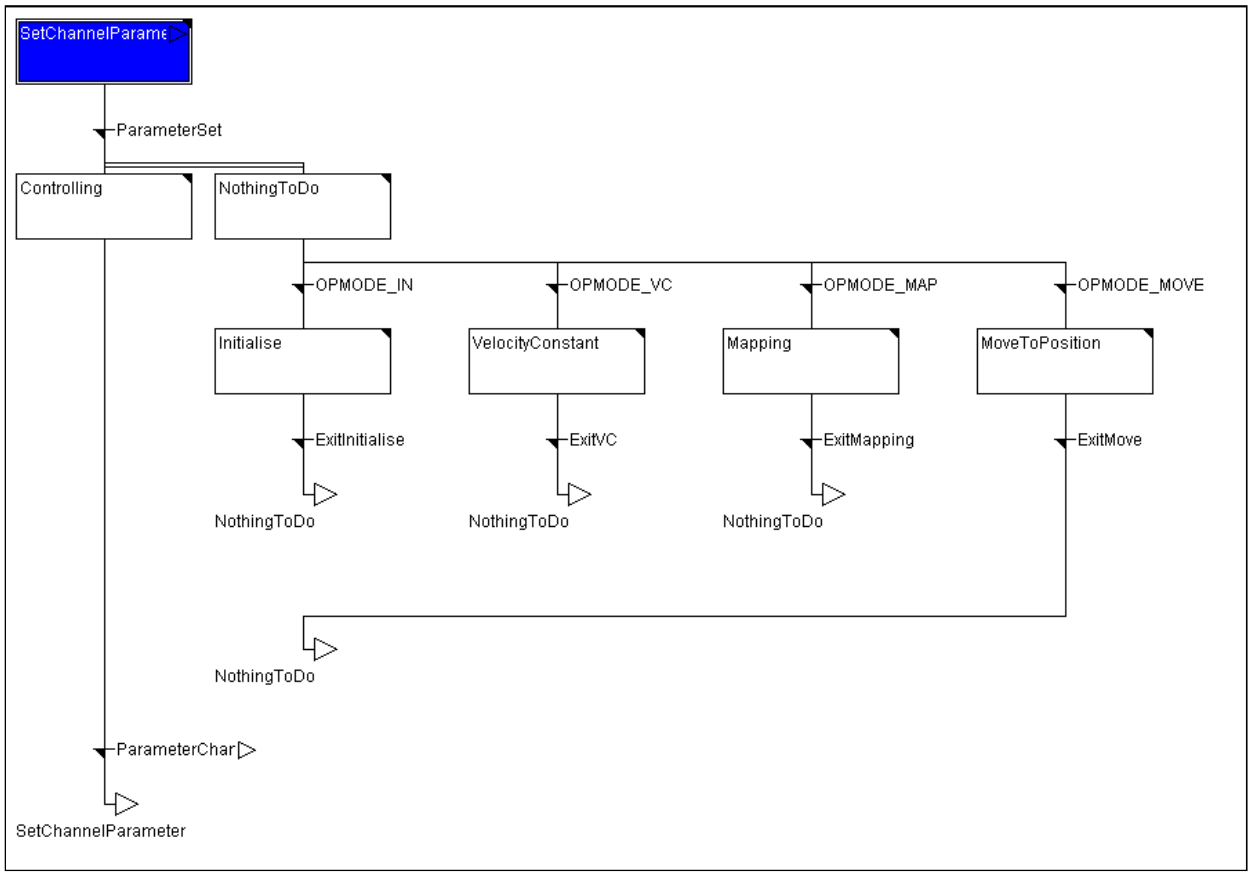


Abb. 6: FB „Motor“: Ablauf 01

- Bei erfolgreicher Übergabe der Parameterwerte wird:
  - die Variable *„g_ControllingStatusParameter [Achsnummer].bParametersDone“* TRUE gesetzt und
  - der Schritt *„SetChannelParameter“* verlassen.

- Nach erfolgreicher Parametervorgabe wechselt die Struktur ihren Status (Instanz des FB) auf folgende Ansicht:

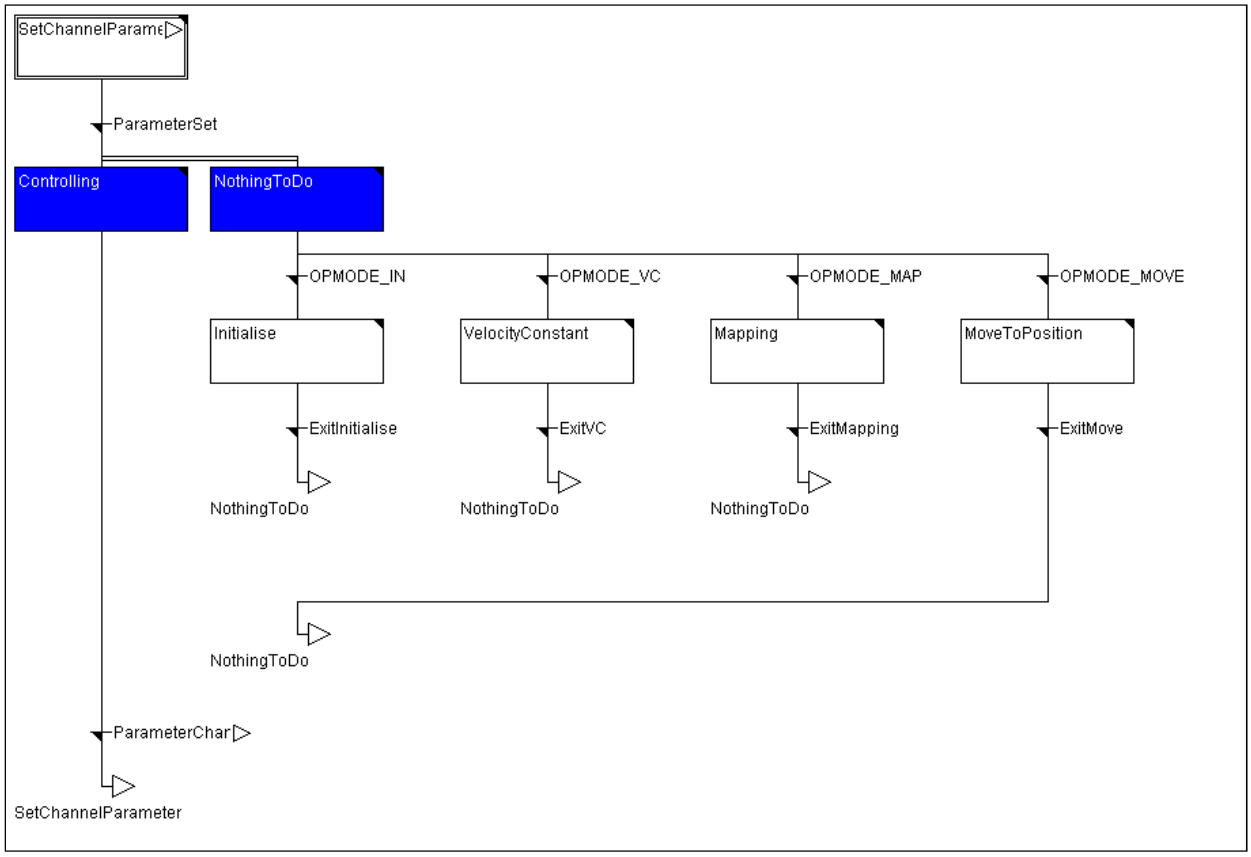


Abb. 7: FB „Motor“: Ablauf 02

- In diesem Status der Achse wartet die Achse darauf, welche Betriebsart umgesetzt werden soll.

3.6. Beispiel für die Achsprogrammierung mittels des FB „Motor“

3.6.1. Allgemeine Beschreibung des FB "Motor"

Für jede Achse muss zyklisch der Funktionsbaustein „Motor“ aufgerufen werden.

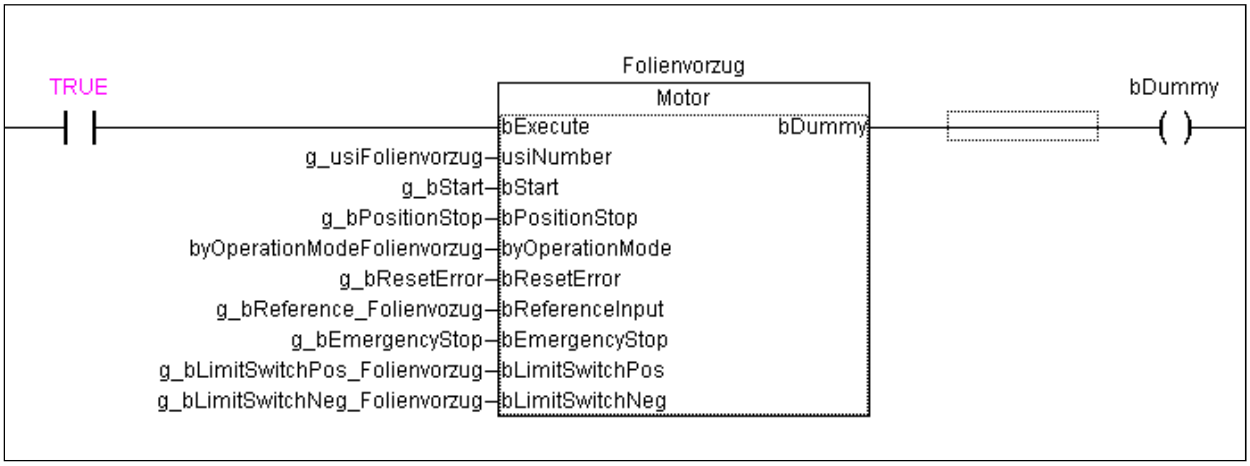


Abb. 8: FB „Motor“

Die Deklaration für digitale Ein- und Ausgänge sollte in der globalen Variablenliste „Input_Output“ erfolgen. Die Rückmeldungen der Achsfehler und des Achsstatus erfolgt im ARRAY „*g_ControllingStatusParameter*“.

Die Eingangsvariablen aus der Abb. 8: FB „Motor“ sind im einzelnen:

- **usiNumber**:
  Beinhaltet die Nummer der Achse in den globalen Arrays der „*zChannelParameters*“. Idealerweise ist dieser Wert identisch mit der Achsnummer in der PMC.
- **bStart**:
  Start der Achse. Der Start wird nur akzeptiert, wenn:
  - keine Fehlernummer anliegt und
  - „*bResetError*“ FALSE ist.
  - Ebenso muss *bEmergencyStop* = TRUE sein und
  - die Endschalter müssen je nach „*OPmode*“ und Fahrrichtung ebenfalls TRUE sein.
  - Es muss „OPmode“ 1 bis 4 angewählt sein.

  Wechselt „*bStart*“ auf FALSE, wird die Achse gestoppt
- **bPositionStop**:
  Die fallende Flanke von „*g_bPositionStop*“ und *g_bStart* = TRUE aktiviert im „*OPmode 2*“ und „*OPmode* 3“ die Funktion Auskuppeln auf Zielposition. Die Art des Auskuppelns und die Zielposition werden im „*g_AllOpmodeParameters*“ ausgewählt bzw. vorgegeben.
- **byOperationMode**:
  Die Auswahl des „OPmode“ muss vor dem Start erfolgen:
  - 1 = Initialisierungsfahrt
  - 2 = Endlospositionierung
  - 3 = Mapping
  - 4 = Positionieren.

- **bResetError**:
  Die steigende Flanke löscht einen eventuell anstehenden Fehler von der PMCprimo (sendet auch SDO zu einem über CAN angeschlossenen Drive), wenn *g_bStar*t = FALSE.
- **bReferenceInput**:
  Referenzsignal, falls vorhanden, muss dem entsprechenden Eingang zugeordnet sein (in globaler Variablenliste z.B. mit AT%IX0.3).
- **bEmergencyStop**:
  Muss immer TRUE sein, damit die Achse aktiv sein kann.
  Bei FALSE wird die Achse in den „*OPmode 0*“ und damit Motor Off geschaltet.
  Die Fehlernummer 266 wird ausgegeben und muss vor dem nächsten Start quittiert/gelöscht werden.
- **bLimitSwitchPos**:
  Endschalter in positiver Fahrrichtung.
  Ohne Endschalter muss dieser Eingang immer TRUE sein.
- **bLimitSwitchNeg**:
  Endschalter in negativer Fahrrichtung.
  Ohne Endschalter muss dieser Eingang immer TRUE sein.

- Fehlernummern, die von der Soft-SPS und nicht von PMCprimo generiert werden:
  - 263: beide Endschalter gleichzeitig FALSE
  - 264: Endschalter in Plus-Richtung angefahren
  - 265: Endschalter in Negativ-Richtung angefahren
  - 266: Not-Aus erkannt

3.6.2. Hinweise zur Anwendung des FB "Motor"

- **udiPositionBound**
  beinhaltet die Zyklusgrenze der Achse.
  Ändert sich der Wert der Variable „*udiPositionBound*“, so wird der neue Wert sofort an die Achse übergeben. Diese Veränderung an einer Masterachse kann an einer sich in „Mapping“ befindenden Slave-Achse zu einer nicht gewollten Bewegung oder zu einem Motorfehler führen!

- **sMapName**
  Ändert sich der Name der Map in „*sMapName*“, dann wechselt die Achse auch bei aktivem „Mapping“ die Tabelle.
  Dies kann zu einer ungewollten Bewegung oder zu einem Motorfehler führen!

Mit Anwahl eines „OPmode“ erfolgt das Schließen des Lageregelkreises (PC), um mit dem Startsignal sofort eine erforderliche Bewegung ausführen zu können.
Das Starten einer Bewegung ist nur möglich, wenn kein Fehler ansteht. Das Quittieren einer Störung ist nur möglich, wenn kein Startsignal anliegt (*bStart* = FALSE)!

- **OPmode 4**:
  - bei absoluter Positionierung wird bei *bStart* = TRUE eine neue Positionierung gestartet, wenn eine neue Zielposition („*diTargetPosition*“) vorgegeben wird.
  - eine relative Positionierung wird nur ausgeführt, wenn ein Flankenwechsel von „*bStart*“ von FALSE auf TRUE erkannt wird.

- Schnelle Wertänderungen (**Fast Trigger**):
  Jede Änderung einer Variablen (außer bei den Variablen von „*g_SetChannelParameters*“) wird, wenn möglich, an die Achse weitergegeben.
  Beispiel:
  Ändert sich z.B. der Wert „*udiAcceleration*“ für die Beschleunigung in jedem SPS-Zyklus, so wird nur jeder 3. Wert übertragen. Endet die Wertänderung, so endet die Übertragung des Parameters erst, wenn der am Schluss aktuelle Wert übertragen ist.
  Unzulässige Änderungen, wie z.B. der Wechsel der Master-Achse während aktiven „Mappings“, werden nicht durchgeführt.

3.7. Hinweise zum FB „PositionTrigger“ (Zusatzfunktion)

Diese Zusatzfunktionalität ist nur eine mögliche Vereinfachung für die Anwendung von dem „Positions-Nocken“ (siehe auch „PO-Befehl“).
Es steht dem Anwender frei, andere Lösungen als diese zu verwenden.

Die PMCprimo bietet die Möglichkeit, einen digitalen Ausgang abhängig von einer Achsposition den logischen Zustand TRUE oder FALSE zuzuweisen. Dies kann bei Verwendung der Soft-SPS auf zweierlei Art ausgeführt werden:

- als „Positionsabfrage“ nur durch die Soft-SPS
  - Vorteil: Innerhalb der Logik problemlos weiterverarbeitbar und bei interner Verwendung in der Anzahl nicht limitiert.
  - Nachteil: Genauigkeit abhängig von der Zykluszeit der Soft-SPS

- als „PositionTrigger“ über die PMCprimo-Funktionalität (PO-Befehl), der von der Soft-SPS nur definiert wird. Es sind 16 virtuelle Nocken pro PMCprimo-Knoten möglich, die über virtuelle Eingänge in der Soft-SPS weiter verarbeitet werden können.
  - Vorteil: Genauigkeit von 1ms unabhängig von der Zykluszeit der Soft-SPS
  - Nachteil: Zahl der realen Nocken begrenzt auf die Zahl der auf dem Gerät vorhandenen digitalen Ausgängen. Zusätzlich können auf dem Host-System bis zu 48 virtuelle, intern verwendbare, Nocken definiert werden.

Beide Möglichkeiten sind als Beispiel im Baustein „PositionTrigger“ ausgeführt.
Die Funktion ist auch durch eine Visualisierung ergänzt. Der Anwender muss in der realen Applikation die erforderlichen Entscheidungen und Belegungen der möglichen Adressen selbst festlegen. Hier ist ein Beispiel für die Visualisierung „PositionTrigger“:

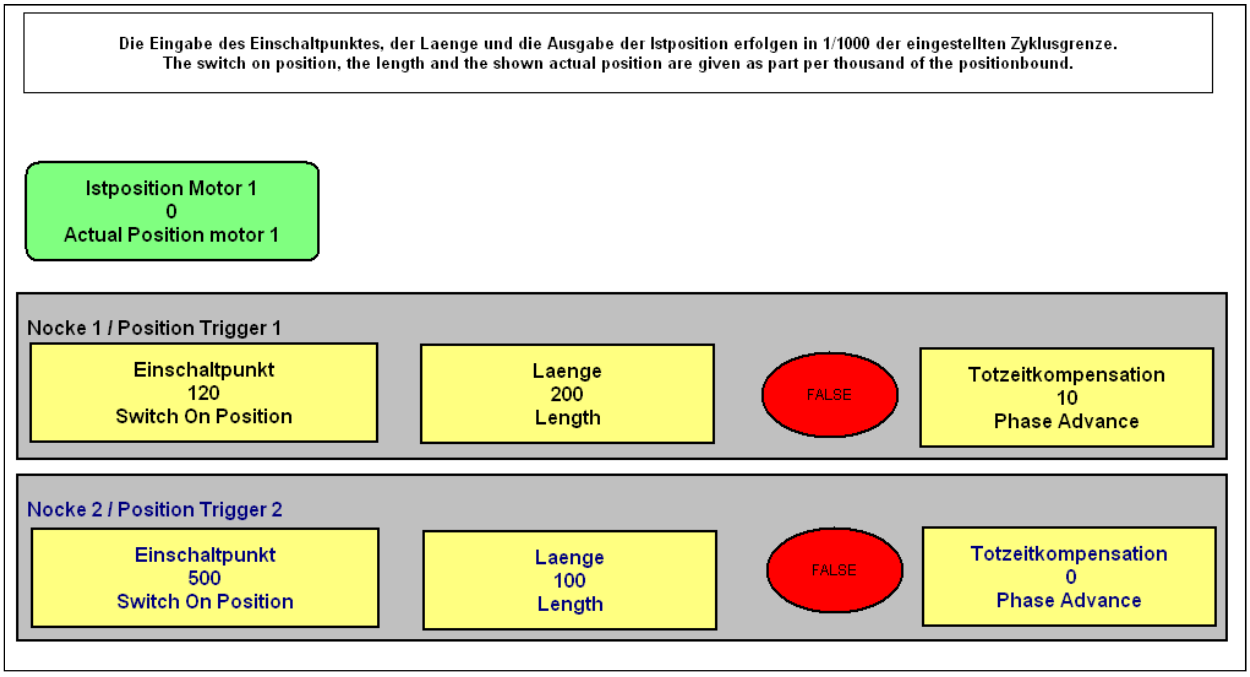


Abb. 9: FB „PositionTrigger“, Visualisierungsbeispiel

3.8. Hinweise zum FB „PO_PLC“

Der Funktionsbaustein „PO_PLC“ erzeugt Nocken, die aus der Abfrage der Ist-Position durch die Soft-SPS selbst gebildet werden.

- Diese Nocken:
  - besitzen keine Totzeitkompensation und
  - sind mit einer Schalthysterese von 10 Inkrementen versehen.
- Der Baustein:
  - liefert eine boolsche Variable und
  - an den Schaltpunkten wird zusätzlich noch für einen SPS-Zyklus eine Flanke ausgegeben.

Belegung der Ein- und Ausgabevariable:

```
0001 FUNCTION_BLOCK PO_PLC
0002 (*
0003 General descriptions:
0004 PositionOutput depending on the cyclerate of the PLC
0005 *)
0006
0007 VAR_INPUT
0008     bExecute : BOOL;              (* not used *)
0009     diActualPosition   :   DINT;      (*  position of the channel.  the value is given in increments *)
0010     diSwitchOnPosition   :   DINT;  (* startposition of the position trigger. the value is given in steps by 0,1 % *)
0011     diLengthPositionTrigger  :   DINT;  (* length of the position trigger. the value is given in steps by 0,1 % *)
0012     diSetBound    :   DINT;       (* setbound value of the channel. the value is given in increments *)
0013 END_VAR
0014 VAR_OUTPUT
0015     bDone : BOOL;          (* copy of bExecute *)
0016     diSwitchOffPosition   :   DINT;       (* value is given in steps by 0,1 % *)
0017     bOutput    :   BOOL;          (* Position trigger output of the function block *)
0018     bOutputRisingEdge   :   BOOL;      (* Rising trigger position output, is TRUE for one PLC-cycle *)
0019     bOutputFallingEdge   :   BOOL;     (* Falling trigger position output, is TRUE for one PLC-cycle *)
0020 END_VAR
```

Abb. 10: FB „PO_PLC“, Variablendeklaration

4. Anhang

4.1. Übersicht: Was ist neu im Basis-Projekt?

Nr.	Änderung / Ergänzung
1	Eingangsdefinitionen dürfen durch Anwahl von „*OPmode 1*“ nicht gelöscht werden (betraf immer Eingang 1)
2	Der Parameter SB (Set Position Bound) muss jederzeit und unabhängig vom OPmode änderbar sein
3	Die Referenz-Parameter müssen jederzeit und „OPmode“ unabhängig änderbar sein.
4	Fast Trigger: Ändert sich ein Parameter über n PLC-Zyklen zyklisch, so muss spätestens im n+1 PLC-Zyklus der aktuelle Wert an PMCprimo übergeben sein.
5	Der Parameter CI hinzugefügt
6	Alle Achsparameter in einer globalen Variablenliste -> g_Array of Structs angelegt für 20 Achsen
7	Pro Achse genügt der Aufruf eines Funktionsbausteins (Schalen-FB für alle intern verwendeten), damit in zukünftig erzeugten Projekten die Standard-FB’s einfach hineinkopiert werden können.
8	Kein Eingriff in die Achsstruktur mehr zur applikationsspezifischen Anpassung notwendig (davor hatten immer viele Angst)
9	Eine erweiterte Steuerungskonfiguration integriert
10	Separate globale Variablenliste für Ein- und Ausgänge
11	CAN-Schieberegister für Achsen auf PMCprimo-Knoten automatisiert im Hintergrund, falls mindestens eine Achse auf einem PMCprimo-Knoten vorhanden.
12	Einführung der ungarischen Notation für die Benennung der Variablen
13	Variablennamen in englisch ohne Verwendung von Kürzeln/Abkürzungen etc.
14	Bei Not-Aus wird die Achse abgeschaltet.
15	Bei Anwahl eines OPmode <> 0 wird die Achse sofort freigegeben, damit kann ein Startsignal schneller in Bewegung umgesetzt werden.
16	Fehler quittieren nur möglich, wenn Start = FALSE
17	Start ist nicht möglich, solange ein Fehler ansteht
18	Die Funktionsbaustein für Initialisierung übernimmt auch die Funktion für das Initialisieren mit Drehgebernullspur.
19	Wird die Option Referenzsignal am Regler ausgewählt, so schreibt die PLC am Regler, falls erforderlich, den IN2MOE auf 26 und führt auch Save&Coldstart aus.
20	Die Vorgabe einer neuen Zielposition führt zur Ausführung der Positionierung, wenn Start = TRUE, bzw. die Achse ändert die anzufahrende Zielposition auf den neuen Wert.
21	Die Anwendung des Basis-Projekts soll für den Applikateur einfacher und sicherer werden.
22	Separates ARRAY of Struct für Rückmeldung von Achsfehler und Achszustand.
23	Funktionsbausteine der PMCprimo Bibliothek erhalten als Instanzenname den 2-Buchstabenbefehl plus den underscore und evtl eine Zusatzinfo

Table 2: Übersicht: Was ist neu im Basis-Projekt?

Beachten:
Das Basis-Projekt wurde neu erstellt und ist mit der alten Version (Beschreibung in alter Pilz-Dokumentation 21 702-xx „PMCprimo Basisprojekt unter CoDeSys“) nicht kompatibel.

4.2. Parametertabellen

4.2.1. Parameter, die keiner speziellen Betriebsart zugeordnet sind

Parameter, die keiner speziellen Betriebsart (unabhängig vom wählbaren OPmode 0 bis 4) zugeordnet sind:

```
    g_AllOpmodeParameters: ARRAY [1..20] OF OpmodeStruct :=
(*
Die folgenden Parameter werden bei Wertänderung an die Achse übergeben. Diese Parameter werden in mehreren Opmodi verwendet.
The following parameters are given to the channel in case of value change. These parameters are used in different opmodes.
*)
(* g_usiMotorNumber1 = 1 *)
        (bRealChannel := TRUE ,              (* TRUE = VM0: Reale Achse / real channel, FALSE = VM1: Virtuelle Achse / virtual channel *)
        bDirection := TRUE ,                 (* Vorgabe der Fahrrichtung für Fahren mit konstanter Geschwindigkeit. TRUE = positive Fahr
        udiPositionBound := 131072 ,         (* siehe SB-Befehl / see the SB command *)
        udiAcceleration := 500000 ,          (* Beschleunigung in Inkrementen pro Sekundequadrat // Acceleration in increments per sec
        udiDeceleration := 500000 ,          (* Bremsrampe in Inkrementen pro Sekundequadrat  // Deceleration in increments per secon
        udiVelocity := 5000 ,                (* Geschwindigkeit in Inkrementen pro Sekunde // Velocity given in increments per second *)
        uiVelocityMul := 1 ,                 (* Multiplikator für Skalierung Geschwindigkeit / velocity multiplied with VelocityMul *)
        uiVelocityDiv := 1 ,                 (* Divisor für Skalierung Geschwindigkeit / velocity divided with VelocityDiv *)
        diTargetPosition := 0 ,              (* Vorgabe der Zielposition skaliert / target position scaled *)
        diTargetStopPosition := 0,           (* Diese Position wird angefahren, wenn PositionStop auf FALSE wechselt. Start muss auf TR
                                             This position will be reached if PositionStop changes from TRUE to FALSE. START has to be
                                             Die Zielposition wird mit TargetPosMul und TargetPosDiv skaliert / The TargetStopPosition is
        uiTargetPosMul := 1 ,                (* Multiplikator für Skalierung Zielposition / target position multiplied with TargetPosMul *)
        uiTargetPosDiv := 1 ,                (* Divisor für Skalierung Zielposition / target position divided with TargetPosDiv *)
        usiPositionWord := 2#00000000 ,      (* siehe ZW-Befehl im Programmierhandbuch // see ZW-command in the reference manual.
        bClutchOutWithST := FALSE,           (* TRUE: Stop auf Position mit ST; FALSE: Stop auf Position mit MA // TRUE: ClutchOut with S
        bRelativeMove := FALSE),             (* TRUE: Positionierung relativ / move relative // FALSE: Positionierung absolut / move absolu
```

Abb. 11: Parameter, die keiner speziellen Betriebsart zugeordnet sind

Variablenname im Basis-Projekt	Defaultwert	Befehl / Bedeutung in PMCprimo
bRealChannel	TRUE	VM
bDirection	TRUE	Fahr- und Suchrichtung bei VC und IN
udiPositionBound	131072	SB
udiAcceleration	500000	SA
udiDeceleration	500000	DC
udiVelocity	5000	SV (bei OPmode 3 auch SS)
bDirection	TRUE	Fahr- / Suchrichtung bei VC oder IN
bRealChannel	TRUE	VM
uiVelocityMul	1	Skalierungsfaktor Geschwindigkeit
uiVelocityDiv	1	Skalierungsfaktor Geschwindigkeit
uiTargetPosMul	1	Skalierungsfaktor Position
uiTargetPosDiv	1	Skalierungsfaktor Position
diTargetPosition	0	Zielposition MA / MR
diTargetStopPosition	0	Zielposition bei Auskuppeln auf Position im OPmode 2 oder 3
bClutchOutWithST	FALSE	Auskuppeln mit ST oder MA bei OPmode 2 oder 3
usiPositionWord	2#00000000	ZW
bRelativeMove	FALSE	MA oder MR

Table 3: Übersicht der Parameter, die keiner speziellen Betriebsart zugeordnet sind

4.2.2. Parameter, die für zyklisches Referieren zugeordnet sind

Parameter, die für zyklisches Referieren und damit ebenfalls keiner speziellen Betriebsart (unabhängig vom wählbaren OPmode 0 bis 4) zugeordnet sind (jede Änderung einer Variablen wird an die Achse weitergegeben.):

```
    g_ReferencingParameters: ARRAY [1..20] OF ReferencingStruct :=
(*  Parameter für einmaliges und zyklisches Referieren / parameters for referencing *)

(*  g_usiMotorNumber1 = 1 *)
        (usiReferenceInputAddress := 0 ,          (* Adresse des Referenzsignals (bei PMCprimo ist Eingang 1 bis 4 von Byte 1 möglich; Ad
                                                  Set Bit address of the reference signal (at PMCprimo only input 1 to 4 of byte 1 is available
        bPolarityInput := TRUE ,                  (* Polarität des Referenzeingangs TRUE => steigende Flanke = Referenz // polarity of refe
        udiReferenceHoldOffTime := 20 ,           (* see RH command *)
        bReferenceModeON := FALSE ,               (* see RM command *)
        bZeroMarkerReference := FALSE ,           (* see DZ command *)
        usiReferenceWord := 2#00000000 ,          (* see RW command *)
        udiReferenceCorrectionVelocity := 25 ,    (* see RV command *)
        udiReferenceAcceleration := 250000 ,      (* see RC command *)
        diReferenceOffset:= 0 ,                   (* see RF command *)
        diReferencePosition_RJ := 0 ,             (* see RJ command *)
        diReferenceLength_RL := 0 ,               (* see RL command *)
        uiMaxReferenceCorrection_SR := 0 ,        (* see SR command *)
        uiReferenceFilter_FR := 2#0000000 ,       (* see FW command *)
        uiReferenceErrorLimit_LR := 0 ,           (* see LR command *)
        usiReferenceFilterOptionWord_FW := 0 ,    (* see FW command *)
        udiReferenceTrueHighLimit_ZH := 0 ,       (* see ZH command *)
        udiReferenceTrueLowLimit_ZL := 0 ,        (* see ZL command *)
        udiReferenceFalseHighLimit_FH := 0 ,      (* see FH command *)
        udiReferenceFalseLowLimit_FL := 0),       (* see FL command *)
```

Abb. 12: Parameter, die für zyklisches Referieren zugeordnet sind

Variablenname im Basis-Projekt	Defaultwert	Befehl / Bedeutung in PMCprimo
usiReferenceInputAdress	0	DR (bei FS21,22,24,25 auch IN2mode 26, falls Variable = 6)
bPolarityInput	TRUE	Flankenrichtung von DR
udiReferenceHoldOffTime	20	RH
bReferenceModeON	FALSE	RM
bZeroMarkerReference	FALSE	DZ
usiReferenceWord	2#00000000	RW
udiReferenceCorrectionVelocity	25	RV
udiReferenceAcceleration	250000	RC
diReferenceOffset	0	RF
diReferencePosition_RJ	0	RJ
diReferenceLength_RL	0	RL
uiMaxReferenceCorrection_SR	0	SR
uiReferenceFilter_FR	0	FR
uiReferenceErrorLimit_LR	0	LR
usiReferenceFilterOptionWord_FW	2#00000000	FW
udiReferenceTrueHighLimit_ZH	0	ZH
udiReferenceTrueLowLimit_ZL	0	ZL
udiReferenceFalseHighLimit_FH	0	FH
udiReferenceFalseLowLimit_FL	0	FL

Table 4: Übersicht der Parameter, die für zyklisches Referieren zugeordnet sind

4.2.3. Parameter, die nur in der Betriebsart „Mapping“ verwendet werden

Parameter, die nur in der Betriebsart „Mapping“ (OPmode = 3) verwendet werden (jede Änderung einer Variablen wird an die Achse weitergegeben):

```
    g_MappingParameters: ARRAY [1..20] OF MappingStruct :=

(* g_usiMotorNumber1 = 1 *)
        (usiNodeMaster := 0 ,              (* Knotennummer der Master-Achse / node number of the master channel *)
        usiChannelMaster := 2 ,            (* Achsnummer des Masters, muss unterschiedlich zur Angabe  der Variablen "Channel" sein // c
        usiMapWord := 2#00000000 ,         (* siehe Befehl MW im Handbuch / see command MW in the Reference Manual *)
        sMapName := 'LINEAR' ,             (* Name der gewünschten Kurve / name of the map *)
        udiClutchTimeLength := 1000 ,      (* siehe Befehle CT und CL im Handbuch // Bit 5 von MapWord entscheidet, ob CL oder CT // see
        udiVelocityAlignment := 2000 ,     (* Geschwindigkeit der Achse während Ausgleichsfahrt (skaliert). // Velocity of the alignment move
        usiAdjustmentVelocity := 25 ,      (* Geschwindigkeit während Abgleich von MB, MF oder SM- Änderung, siehe auch AV-Befehl // velo
        udiSM_Counter := 1 ,               (* Multiplikator vom SM-Befehl / multiplier of SM command *)
        udiSM_Divider := 1 ,               (* Divisor vom SM-Befehl, muss >0 sein / divider of SM command, it has to be >0 *)
        diMapBase := 0 ,                   (* Siehe Befehl MB / see MB command *)
        diMapOffset := 0 ,                 (* Siehe Befehl MF / see MF command *)
        uiClutchWindow := 0),              (* Siehe Befehl CI / see CI command *)
```

Abb. 13: Parameter, die nur in der Betriebsart "Mapping" verwendet werden

Variablenname im Basis-Projekt	Defaultwert	Befehl / Bedeutung in PMCprimo
usiNodeMaster	0	ML
usiChannelMaster	2	ML
usiMapWord	2#00000000	MW
sMapName	LINEAR	XM
udiClutchTimeLength	1000	CT / CL
udiVelocityAlignment	2000	SV / SS
usiAdjustmentVelocity	25	AV
udiSM_Counter	1	SM
udiSM_Divider	1	SM
diMapBase	0	MB
diMapOffset	0	MF
uiClutchWindow	0	CI
AA wird auf den Wert von SA (*udiAcceleration*) gesetzt, wenn OPmode = 3		

Table 5: Übersicht der Parameter, die nur in der Betriebsart "Mapping" verwendet werden

4.3. Flussdiagramme

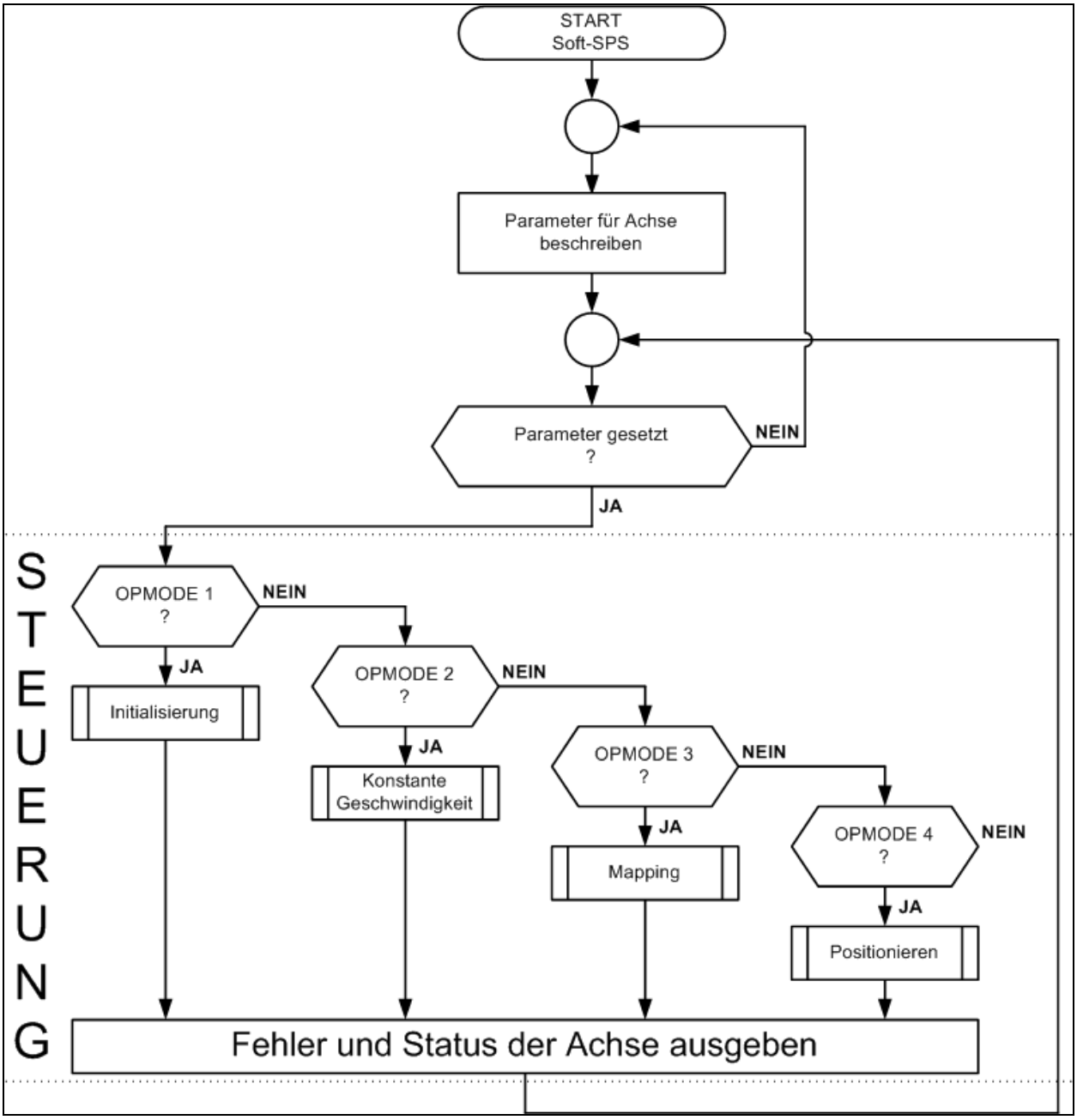


Abb. 14: Flussdiagramm „Ablauf“ (4 Grundfunktionen pro Achse)

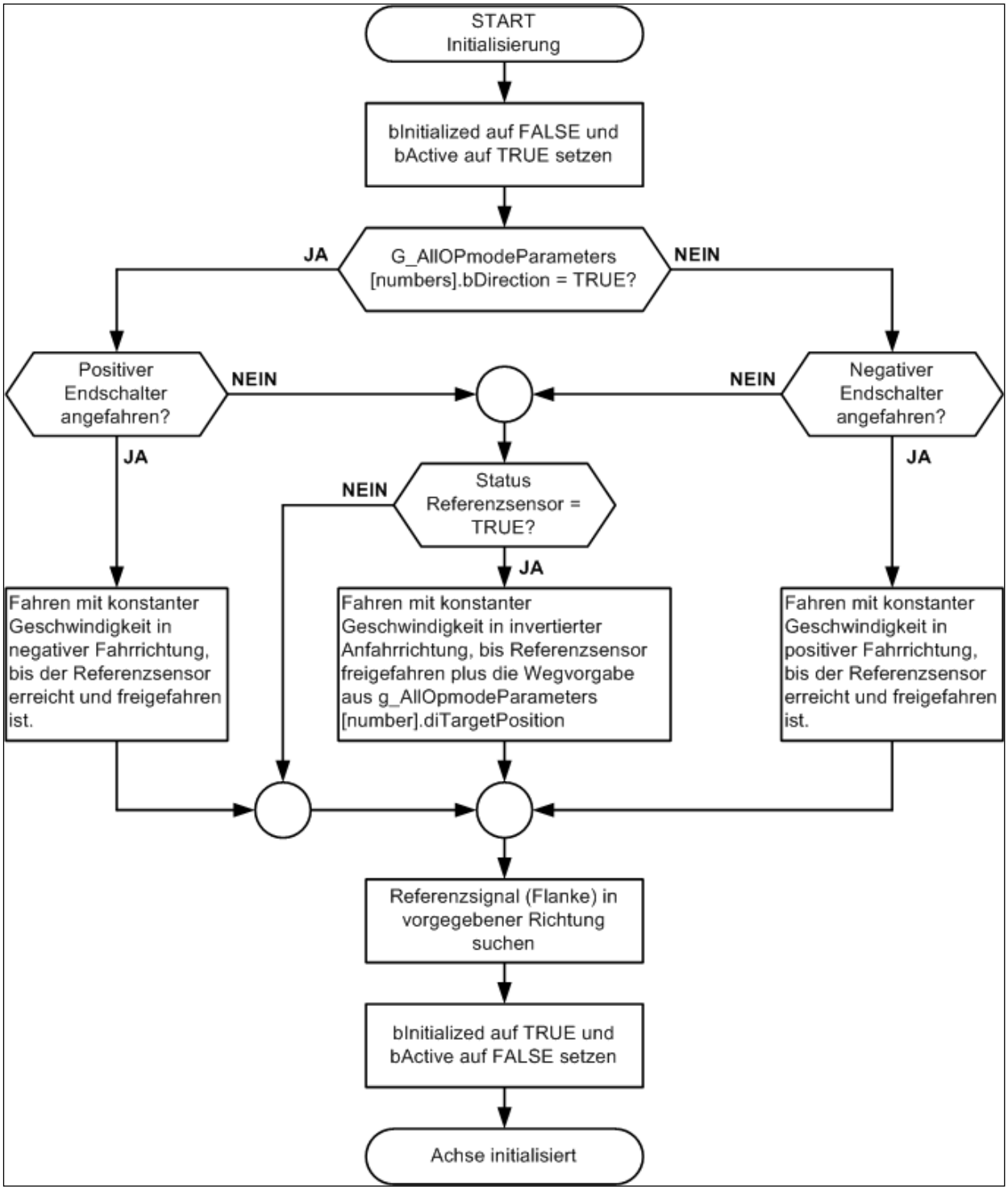


Abb. 15: Flussdiagramm „Initialisierung“

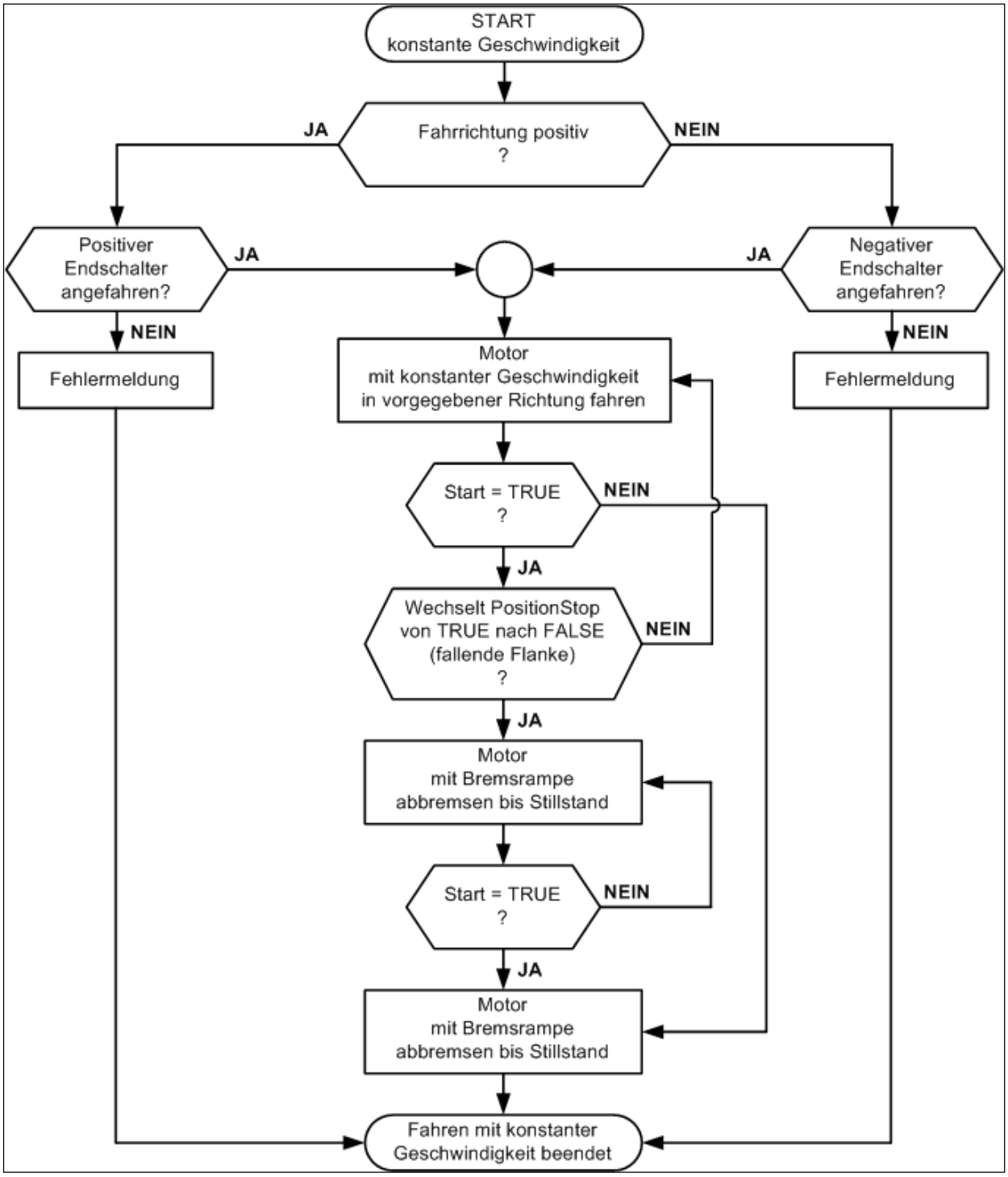


Abb. 16: Flussdiagramm „Konstante Geschwindigkeit“

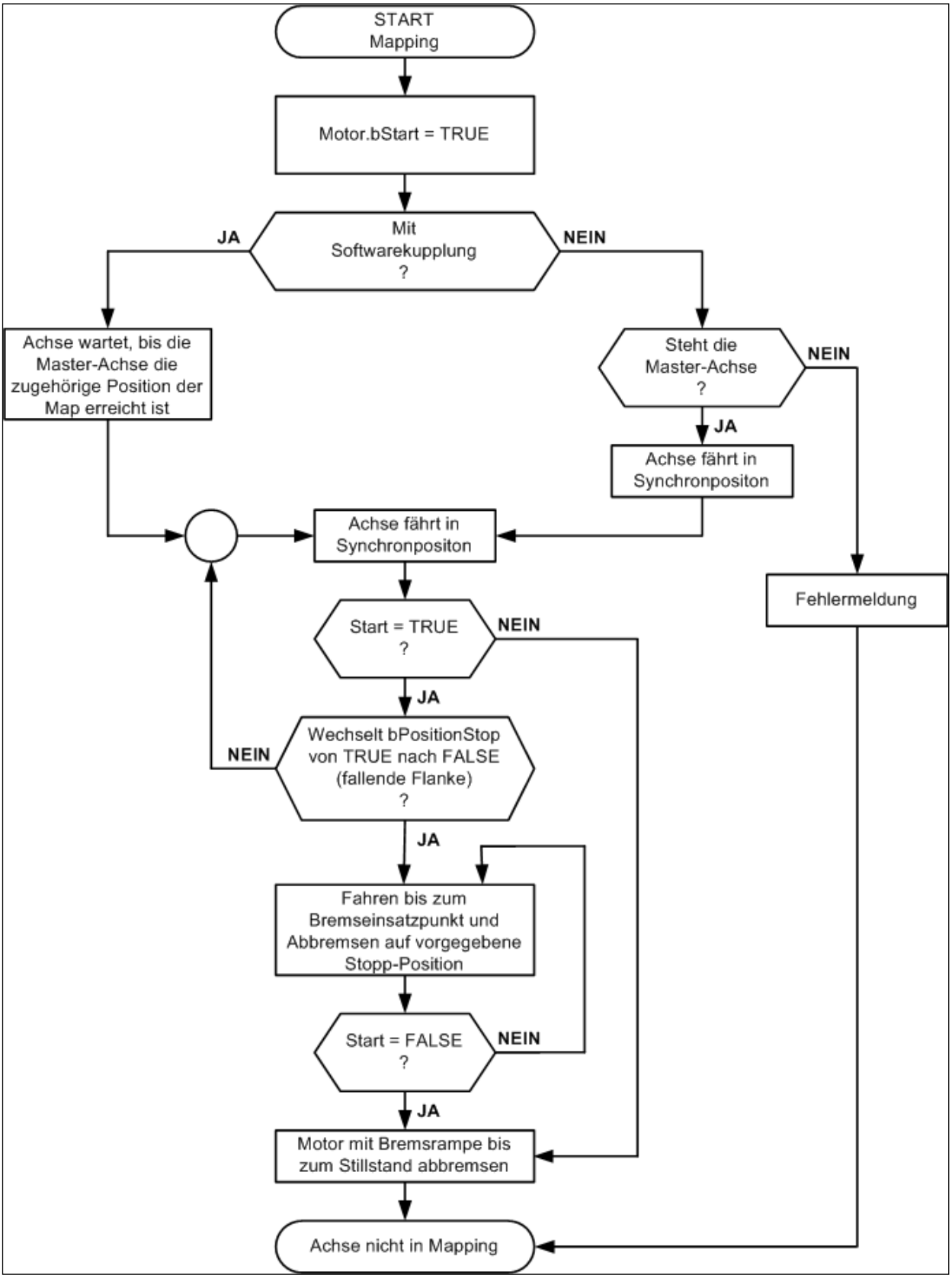


Abb. 17: Flussdiagramm „Mapping“

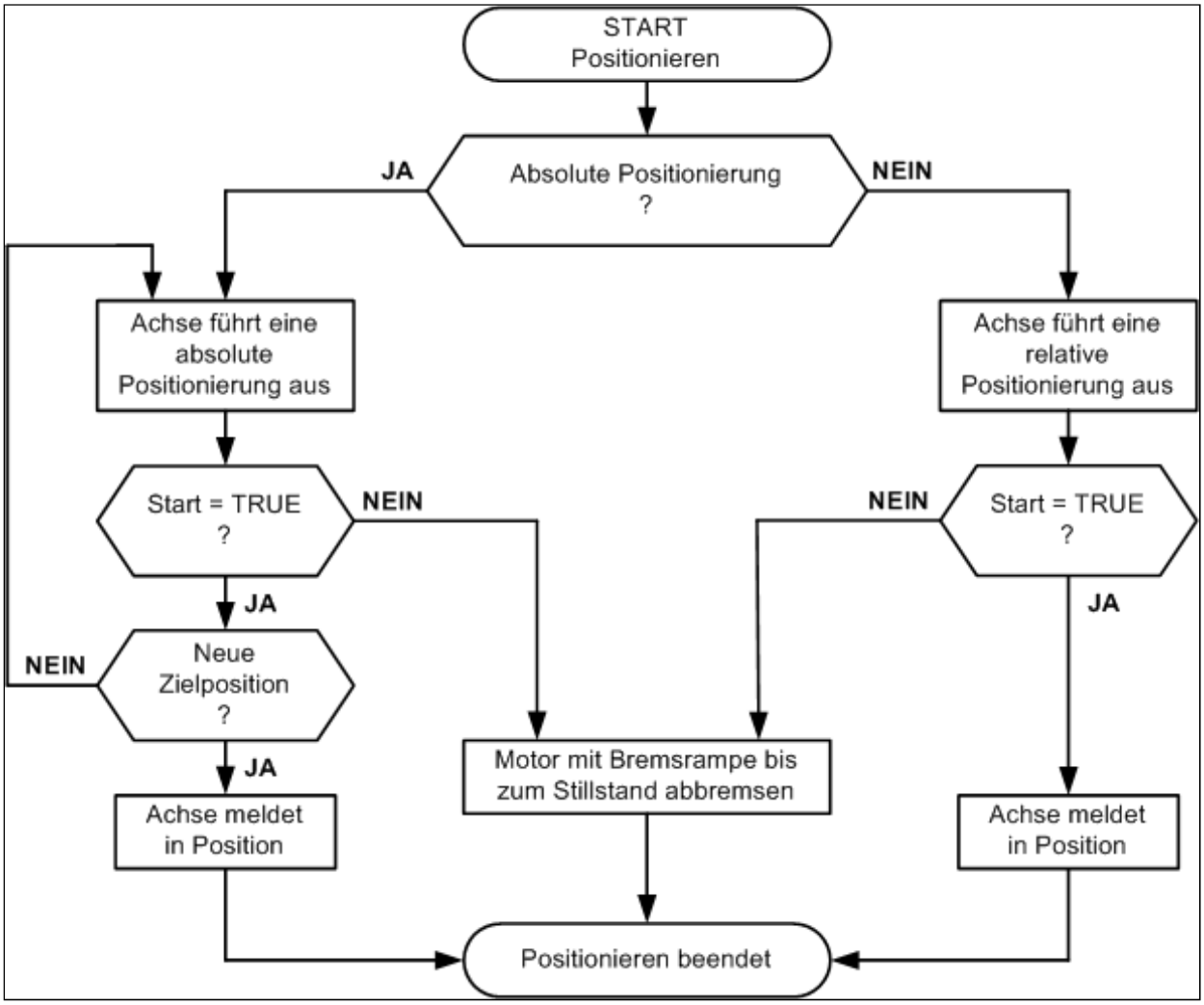
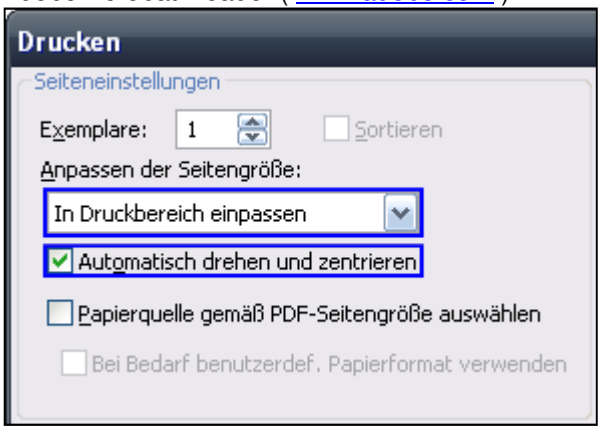


Abb. 18: Flussdiagramm „Positionieren“

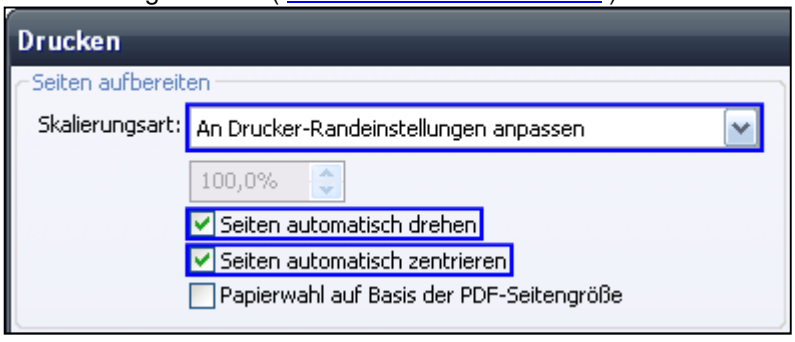
5. Abbildungsverzeichnis

Empfohlene Druckereinstellungen

Adobe Acrobat Reader (www.adobe.com)



PDF-XChange Viewer (www.tracker-software.com)



Technical support
+49 711 3409-444
support@pilz.com

...
In many countries we are represented by our subsidiaries and sales partners.

Please refer to our homepage for further details or contact our headquarters.

Pilz GmbH & Co. KG
Felix-Wankel-Straße 2
73760 Ostfildern, Germany
Telephone: +49 711 3409-0
Telefax: +49 711 3409-133
E-Mail: pilz.gmbh@pilz.de
Internet: www.pilz.com