Automatisierungsbausteine

L_ICIA_CommunicationInterface Funk- _ _ _ _

Referenzhandbuch EN

Lenze

Dieses Handbuch gilt für die Automatisierungsbausteine „“ *L_ICIA_COMMUNICATIONINTERFACE Funktionsbausteine* .

Impressum

Lenze SE

Hans-Lenze-Straße 1, D-31855 Aerzen, Deutschland

Telefon: +49 (0)5154 / 82-0

Fax: +49 (0)5154 / 82-2111

E-Mail: Lenze@Lenze.de

Inhalt

1 Funktionsblöcke

1.1 Dokumenthistorie

Version			Beschreibung
0.1	24.11.2025	LSE	Erste Ausgabe
0.2	04.12.2025	LSE	Aktualisierung zur vereinfachten GSD-Identifizierungsmethode
1.0	03.02.2026	LSE	Version V1.0 veröffentlicht

1.2 Über die Automatisierungsbausteine „

Dieses Handbuch beschreibt eine Softwarelösung für eine Teilaufgabe.

Es liegt in der Verantwortung des Benutzers, zu überprüfen, ob die von der Software vorgeschlagene Lösung seinen Anforderungen entspricht. Falls erforderlich, muss die Lösung angepasst werden. Physikalische Aspekte wie die Konstruktion des Antriebs sind nicht Bestandteil dieses Handbuchs.

Hinweis

Die in diesem Handbuch enthaltenen Anschlusspläne zeigen die Verkabelung, die für den Betrieb der Software auf einem Beispiel-Demogerät erforderlich ist.

1.3 Verwendete Konventionen

In diesem Handbuch werden die folgenden Konventionen verwendet, um zwischen verschiedenen Arten von Informationen zu unterscheiden:

Art der Information	Hervorhebung	Beispiel/Anmerkungen
Schreibweise von Zahlen		
Dezimalzeichen	Punkt	Der Dezimalpunkt wird immer verwendet. Beispiel: 1234,56
Text		
Programmname	» «	»PLC Designer« ...
Variablennamen	*kursiv*	Durch Setzen von *xEnable* auf TRUE...
Funktionsblöcke	**fett**	Der Funktionsbaustein **L_MC1P_AxisBasicControl** ...
Funktionsbibliotheken		Die Funktionsbibliothek **L_TT1P_TechnologyModules** ...
Schaltflächen		... und bestätigen Sie mit **„Weiter**“.

Quellcode	Kurier	`...`
		`dwNumerator := 1;`
		`dwDenominator := 1;`
		`...`

Schlüsselwörter	**Courier fett**	...beginnt mit **`FUNCTION`** und endet mit **`END FUNCTION`**.
Tastaturbefehle	**<bold>**	Drücken Sie die Taste **<F2>**, um Hilfe zur Eingabe anzufordern
		Wenn für die Ausführung eines Befehls eine Tastenkombination erforderlich ist, werden die Befehle durch ein „+“ getrennt: Drücken Sie die Tasten **<Umschalt>+<ESC>**, um ...

Variablennamen

Die von Lenze verwendeten Konventionen für die Variablennamen von Lenze-Systemblöcken, Funktionsblöcken und Funktionen basieren auf der „ungarischen Notation“. Diese Notation ermöglicht es, die wichtigsten Eigenschaften (z. B. den Datentyp) der entsprechenden Variable anhand ihres Namens zu identifizieren, z. B. *xAxisEnabled*.

1.4 Systemanforderungen

Software

Produkt	Typ	Version
PLC Designer		4.1 oder höher

Hardware

Produkt	Typ	Hardware-Version	Firmware-Version
i950	I95AExxxF1AV10Z02R	nicht relevant	1.14 oder höher
PROFIBUS-Steckplatzmodul	I9MAFP0000000S		

2 Funktionsblöcke

Die Funktionsblöcke **L_ICIA_PROFIBUS_Base**, **L_ICIA_PROFIBUS_In** und **L_ICIA_PROFIBUS_Out** sind für Ersatzszenarien der Servoumrichter-Serie 9300 durch die neuesten CbM/DbM-Systeme von Lenze wie beispielsweise den i950 vorgesehen.

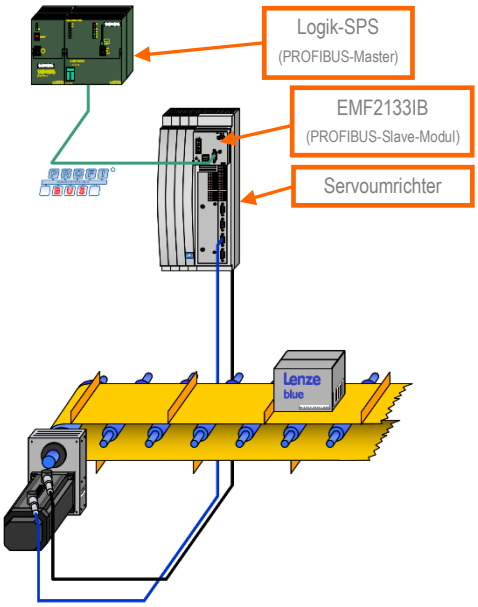


Abbildung1 : 9300 Servoumrichter mit PROFIBUS-Modul[1]

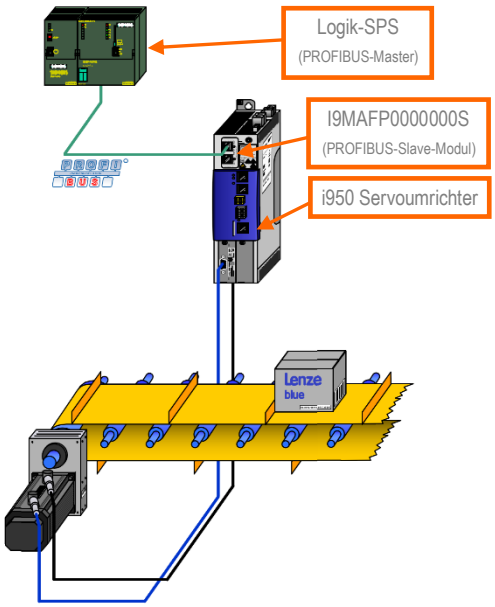


Abbildung2 : i950 Servoumrichter mit PROFIBUS-Steckplatzmodul

Hinweis:

In vielen Fällen ist möglicherweise ein Eins-zu-Eins-Austausch erforderlich, ohne das Programm der Logik-SPS zu verändern.

[1] Für Lenze-Geräte wie 9300 standen zwei AIF-Module für PROFIBUS zur Verfügung:
1 = EMF2133IB mit erweitertem Umfang an GSD/GSE-Konfigurationen (siehe Kapitel „4.1 ")
2 = EMF2131IB mit einem grundlegenden Umfang an GSD/GSE-Konfigurationen (nur Drivecom-Antriebsprofil mit 1 … 4 Prozessdaten)

Es gibt drei Funktionsblöcke, die die AIF-Feldbuskommunikation in drei Unterfunktionen aufteilen:

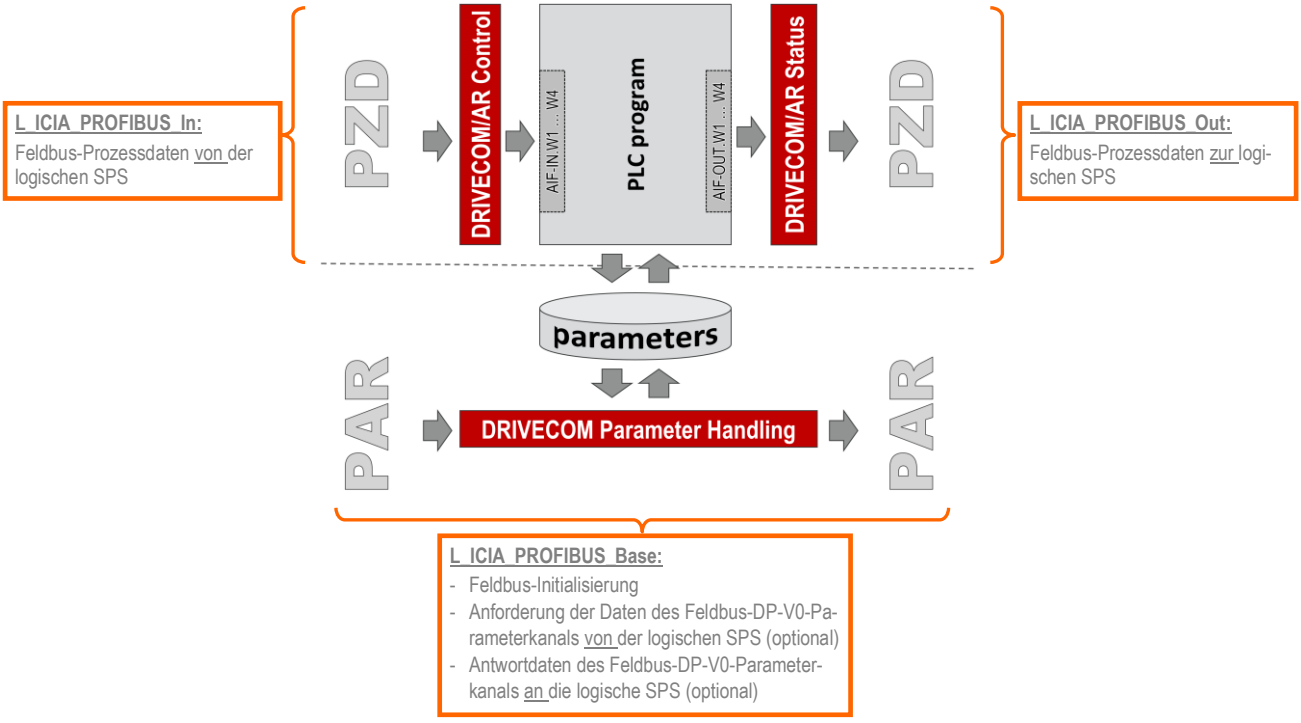


Abbildung3 : Übersicht über die Prozessdatenkommunikation (PDO) und die Parameterkanalkommunikation (SDO)

1. Ein Funktionsblock **L_ICIA_PROFIBUS_In** zum Extrahieren der Prozessdaten-Steuerungsinformationen aus den auf PROFIBUS empfangenen Rohdaten. Die Methode liefert die Prozessdaten im Format **AIF_IN** des Servoumrichters 9300 (siehe Kapitel „2.2 ” und „3.2 ”).
2. Ein Funktionsblock **L_ICIA_PROFIBUS_Out** zum Zusammenstellen des vollständigen Feldbus-Telegramms, das an die Logik-SPS übertragen werden soll. Die Methode liest die Prozessdaten-Steuerungsinformationen im Format **AIF_Out** des Servoumrichters 9300 (siehe Kapitel2.3 und3.2).
3. Der Funktionsbaustein **L_ICIA_PROFIBUS_Base**, der den Kommunikationsaufbau und den DP-V0-Parameterkanal verarbeitet (siehe Kapitel2.1).

Hinweis:

Deklarieren und rufen Sie die PROFIBUS-Funktionsbausteine immer in der folgenden Reihenfolge auf:

- **L_ICIA_PROFIBUS_Base**
- **L_ICIA_PROFIBUS_In**
- **L_ICIA_PROFIBUS_Out**

2.1 Funktionsbaustein *L_ICIA_PROFIBUS_Base*

Der Funktionsbaustein **L_ICIA_PROFIBUS_Base** muss zur korrekten Initialisierung der GSD/GSE-Konfiguration immer in das SPS-Projekt integriert werden. Bei Konfiguration verarbeitet der Baustein zusätzlich den DP V0-Parameterkanal, sofern dieser ausgewählt ist.

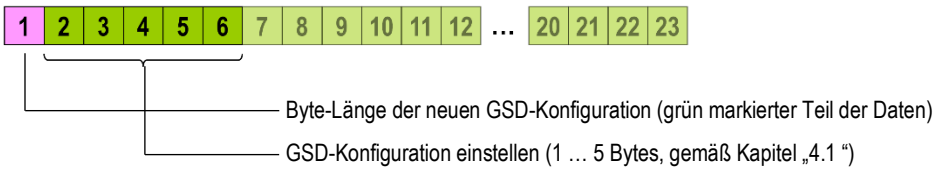
2.1.1 Auswahl des Konfigurationsmodus (GSD/GSE-Konfiguration)

Die Konfiguration der PROFIBUS-Kommunikation für jeden Slave wird mit Hilfe der GSD/GSE-Datei definiert. In der Regel wird der Umfang der PROFIBUS-Tx/Rx-Daten während der Programmierung der Logik-SPS vordefiniert. Eine aus einer Reihe möglicher Konfigurationen (siehe Kapitel „4.1 ”) bestimmt die Struktur des PROFIBUS-Telegramms an ein Slave-Gerät.

Während der Initialisierung der PROFIBUS-Kommunikation identifiziert der Funktionsbaustein **L_ICIA_PROFIBUS_Base** die ausgewählte GSD/GSE-Konfiguration, wie im Anhang, Kapitel „4.1 ” aufgeführt. Zu diesem Zweck liest ein interner Service[2] die empfangene GSD-Konfiguration.

GSD/GSE-Konfiguration einstellen (BYTE ARRAY[23])

Die angeforderte GSD-Konfiguration wird vom Funktionsbaustein **L_ICIA_PROFIBUS_Base** kontinuierlich abgefragt, um nach einer neuen GSD/GSE-Konfiguration zu suchen:



Die aktive GSD-Konfiguration wird im Index 0x2348:003 angezeigt.

[2]

2.1.2 Parameterhandhabung

Die Parameterkommunikation muss über die GSD/GSE-Konfiguration aus der logischen SPS ausgewählt werden und ist in den vollständigen Rx/Tx-PROFIBUS-Daten enthalten. Beim Empfang einer gültigen GSD/GSE-Konfiguration (*xInit* = FALSE) wird ein optionaler Parameterkanal (DRIVECOM DP V0) initialisiert, sofern dieser in der GSD/GSE-Konfiguration enthalten ist.

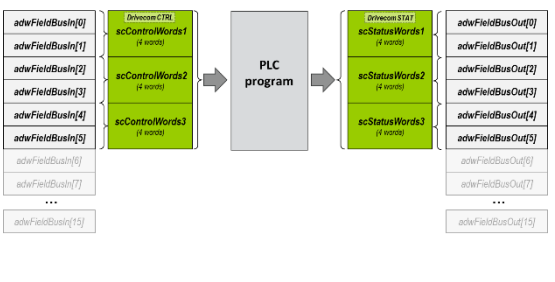


Abbildung4 : kein Parameterkanal (nur PZD)

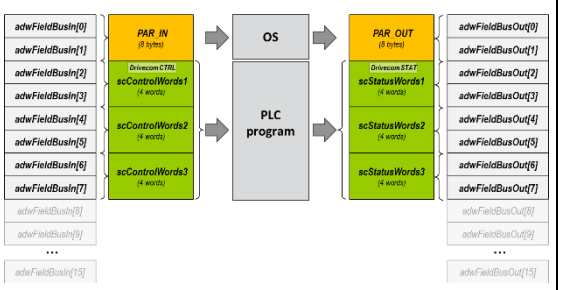


Abbildung5 : mit Parameterkanal (PZD + PAR)

Bei einem konfigurierten Parameterkanal werden die niedrigsten 8 Bytes der auf *L_ICIA_PROFIBUS_In.adwFieldBusIn* empfangenen/auf *L_ICIA_PROFIBUS_Out.adwFieldBusOut* gesendeten Rohdaten wie im Anhang, Kapitel „4.6 ” und „4.7 ” dargestellt interpretiert.

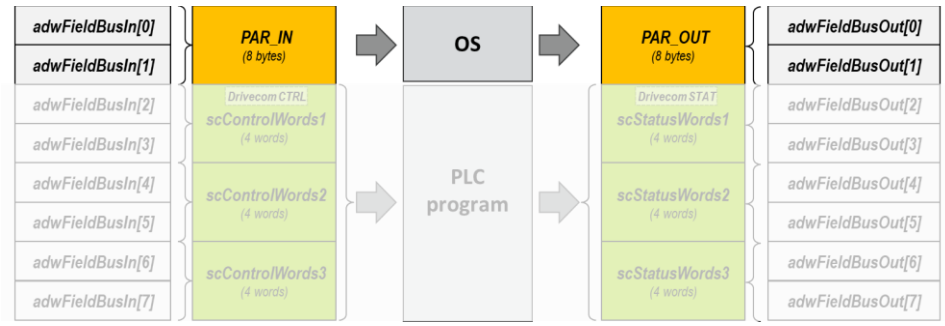
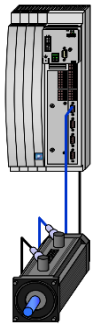


Abbildung6 : DP V0-Parameterkanaldaten als Teil des vollständigen PROFIBUS-Telegramms

In Migrationsszenarien kann es vorkommen, dass die überlagerte Logik-SPS Adresscodes/Subcodes der Lenze GDC-[3] -Geräte 8200/9300 (Legacy-Geräte) adressiert, die auf dem Servoumrichter i950 nicht verfügbar sind.

[3] GDC = Global Drive Control, eine der erfolgreichsten Frequenzumrichter-/Servoumrichter-Serien von Lenze

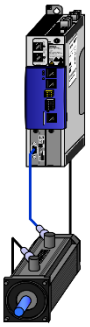
Beispiel:



tatsächliche Motordrehzahl:

Codenummer:	51
Untercodenummer:	0
Zugriff:	schreibgeschützt
Einheit:	[rpm]
Größe:	4 Byte
Skalierungsfaktor:	10000 : 1 (FIX32[4])

Abbildung7 : 9300 Servoumrichter (Auslaufmodell)



tatsächliche Motordrehzahl:

Indexnummer:	0x606C
Unterindexnummer:	0
Zugriff:	schreibgeschützt
Einheit:	[rpm]
Größe:	4 Byte
Skalierungsfaktor:	2^{31} : 480000 („_s"[5])

Abbildung8 : i950-Servoumrichter (aktuelles Produkt)

Aufgrund der Nichtübereinstimmung zwischen der Codeliste des alten GDC-Geräts und der aktuellen Indexliste des i950 ist eine interne Neuzuordnung der Codes/Subcodes zu den aktuellen (Benutzer-)Indizes erforderlich. Die Entsprechung zwischen den alten GDC-Codes und den i950-Indizes wird über eine Zuordnungstabelle auf dem Ein-/Ausgang *ascParReference* definiert.

Die Erstellung der Zuordnungstabelle muss vom Benutzer auf folgende Weise vorgenommen werden:

start

Schritt 1

Legen Sie die Anzahl der Objekte in Ihrer Zuordnungsliste fest: Die Anzahl der Objekte bestimmt, auf wie viele verschiedene Codes/Indizes Sie zugreifen möchten.

Beispiel: Die Logik-SPS verwendet den DP V0-Parameterkanal, um die folgenden Codes von einem 9300-Servoumrichter zu lesen:

- C0051/000: tatsächliche Motordrehzahl, skaliert in [U/min]
- C0053/000: tatsächliche Gleichstrom-Busspannung, skaliert in [V]
- C0063/000: tatsächliche Motortemperatur, skaliert in [°C]

In diesem Beispiel erfordert die Zuordnungsliste drei Einträge.

Schritt 2

Deklarieren Sie im SPS-Programm des i950 ein Datenarray mit einer entsprechenden Anzahl von Einträgen vom Typ *L_ICIA_sc93ParReference* wie folgt:

```
ascParReference:        ARRAY[1..3] OF L_ICIA_sc93ParReference;              // Zuordnungstabelle
```

[4] Das FIX32-Format verwendet eine Datengröße von 4 Byte und einen Skalierungsfaktor von 10000, d. h. ein Wert von 10000 entspricht einem physikalischen Wert von 1,0000 [U/min]. Dieses Format wird in der Lenze-Terminologie auch als „_e4"-Format bezeichnet.
[5] Die Skalierung „_s" wurde mit der Lenze 9400-Serie eingeführt und skaliert die Motordrehzahl als 32-Bit-Wert. Ein Rohwert von 2^{31} entspricht einem physikalischen Wert von 480000[U/min].

Schritt 3

Erweitern Sie die Deklaration aus Schritt 2, indem Sie dem Datenarray der Zuordnungstabelle Initialisierungswerte zuweisen:

```
ascParReference:        ARRAY[1..4] OF L_ICIA_sc93ParReference   := [                    // Zuord-
nungstabelle
 (wCode:=11, wSubCode:=0, wIndex:=16#5500, bySubIndex:=1, bySize:=8, diNum:=10000,     diDen:=1),
 (wCode:=51, wSubCode:=0, wIndex:=16#606C, bySubIndex:=0, bySize:=4, diNum:=1171875, diDen:=524288),
 (wCode:=53, wSubCode:=0, wIndex:=16#6079, bySubIndex:=0, bySize:=4, diNum:=10,       diDen:=1),
 (wCode:=63, wSubCode:=0, wIndex:=16#2D49, bySubIndex:=5, bySize:=2, diNum:=1000,     diDen:=1)];
```

Wie findet man die Werte für das Verhältnis von Zähler und Nenner?

Der Zähler/Nenner skaliert einen Rohwert des physischen Lenze-Geräts auf den Rohwert des älteren Lenze-Geräts.

Beispiel: Der Zähler-/Nennerwert für die tatsächliche Motordrehzahl ergibt sich aus dem Skalierungsverhältnis des Servoumrichters i950 (0x606C:000) und des älteren Lenze-Produkts (C0051/000):

Skalierungsverhältnis des physischen Lenze-Geräts (i950)
Die Index-Skalierungsfaktoren können Sie in

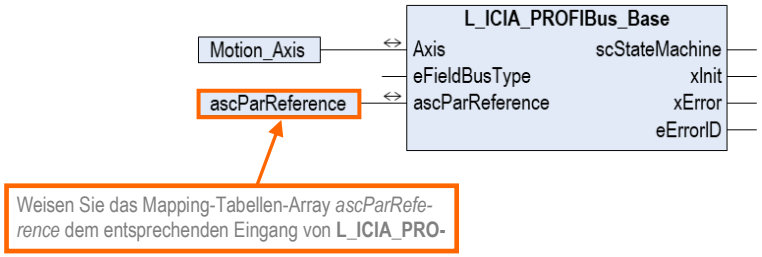
$$\frac{diNum}{diDen} = \frac{480000[rpm]}{2^{31}} \cdot \frac{10000}{1} = \frac{4800000000}{2147483648} = \frac{1171875}{524288}$$

Skalierungsverhältnis des Legacy-Lenze-Geräts (9300)
Die Code-Skalierungsfaktoren finden Sie im Re-

*) Verwenden Sie Berechnungen mit dem größten gemeinsamen Nenner, um die Zahlen für Zähler/Nenner zu verkürzen.

Schritt 4

Weisen Sie das Datenarray der Zuordnungstabelle dem Funktionsblock **L_ICIA_PROFIBUS_Base** zu:



Tipp:

Rufen Sie den Funktionsbaustein **L_ICIA_PROFIBUS_Base** in einer Freilauf-Task mit niedriger Priorität auf, um die Motion-Task mit hoher Priorität zu entlasten.

Die Funktionsbausteine **L_ICIA_PROFIBUS_In** und **L_ICIA_PROFIBUS_Out** für Prozessdaten können jedoch weiterhin in der Bewegungsaufgabe mit hoher Priorität aufgerufen werden.

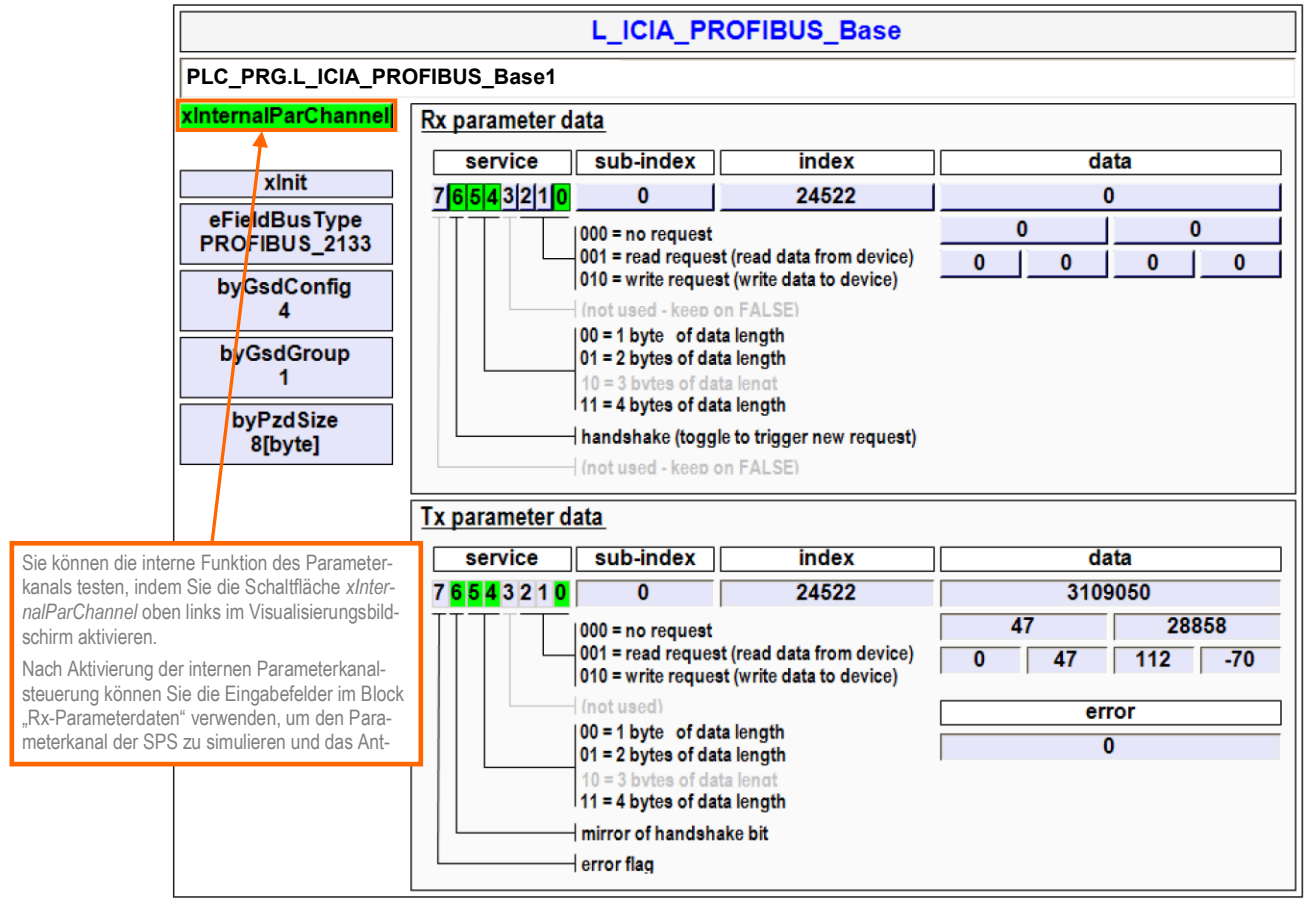
Ende

Hinweis:

Im Gegensatz zur Serie 9300 erlaubt die Serie i950 Parameter mit einem Fließkomma-Datentyp (*LREAL*) mit einer Datengröße von 8 Byte. Auch dieser Parametertyp der Serie i950 kann vom Funktionsbaustein **L_ICIA_PROFIBUS_Base** verarbeitet werden.

Um einen Datenwert mit vier Dezimalstellen auf Ihrer Maschinen-SPS zu empfangen, wenden Sie eine Zähler-/Nenner-Skalierung von *diNum* = 10000 und *diDen* = 1 in der Parameterreferenzliste an.

Zur Überwachung des Parameterkanals enthält der Funktionsblock **L_ICIA_PROFIBUS_Base** einen integrierten Visualisierungsbildschirm:



Sie können die interne Funktion des Parameterkanals testen, indem Sie die Schaltfläche *xInternalParChannel* oben links im Visualisierungsbildschirm aktivieren.

Nach Aktivierung der internen Parameterkanalsteuerung können Sie die Eingabefelder im Block „Rx-Parameterdaten“ verwenden, um den Parameterkanal der SPS zu simulieren und das Ant-

So finden Sie heraus, auf welche Parameter/Indizes die logische SPS zugreift

Manchmal ist das Programm der logischen SPS nicht verfügbar. In diesem Fall sind die von den logischen SPSen aufgerufenen Parameter nicht im Voraus bekannt und können in der Referenzliste nicht berücksichtigt werden.

Eine einfache Möglichkeit, sich einen Überblick zu verschaffen, besteht darin, die Parameteranforderungstelegramme des DP-V0-Parameterkanals zu verfolgen. Gehen Sie dazu wie folgt vor:

- Fügen Sie im SPS-Projekt des i950-Antriebs eine neue Ablaufverfolgung in „PLC Designer“ ein.
- Fügen Sie der Ablaufverfolgung die folgenden Variablen der Parameterkanal-Rx/Tx-Telegramme hinzu:
  - Rx-Index (*L_ICIA_PROFIBUS_Base1.RxParData.wIndex*)
  - Rx-Subindex (*L_ICIA_PROFIBUS_Base1.RxParData.bySubIndex*)
- Starten Sie die Ablaufverfolgung, während die logische SPS versucht, über den DP V0-Parameterkanal auf die Antriebsparameter zuzugreifen.
- Aktivieren Sie einen Messcursor in der Ablaufverfolgung: Die am Rx-Index und Rx-Subindex gemessenen Werte geben die Antriebsparameter an, auf die die Logik-SPS zuzugreifen versucht.

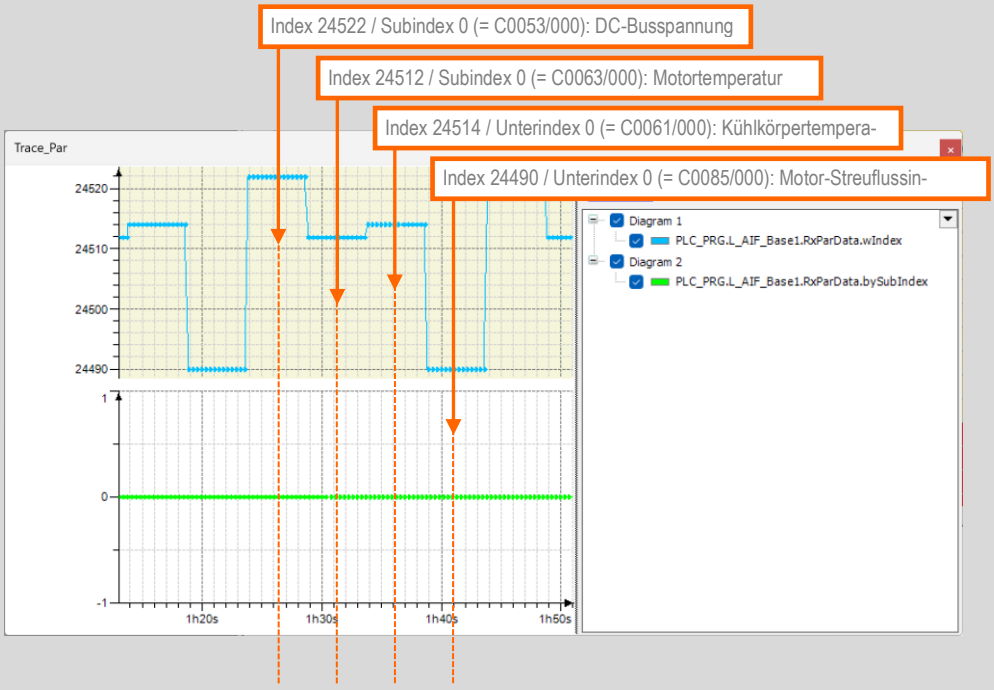


Abbildung10 : Beispiel für eine Trace-Überwachung des Parameterzugriffs der SPS (SDO)

2.1.3 Inkompatibilitätsliste

Die folgenden Funktionen sind im Funktionsbaustein **L_ICIA_PROFIBUS_Base** nicht implementiert:

- Es wird kein PROFIsafe-Protokoll auf i950 PROFIBUS unterstützt.
- Die Parameter-/Indexnummern zwischen 9300 und i950 stimmen nicht überein. Verwenden Sie eine Zuordnungsliste als Referenz zwischen den Parametern eines älteren Lenze-Geräts und einem i950-Antriebsregler, wie im vorherigen Kapitel „2.1.2 ” gezeigt.

2.1.4 Schnittstelle

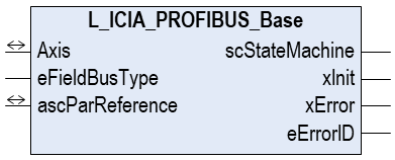


Abbildung11 : Schnittstelle des Funktionsblocks **L_ICIA_PROFIBUS_Base**

2.1.5 Aufgabeninformationen

Aufruf möglich von:	☒ Freilauf-Task	☒ zeitgesteuerter Aufgabe (INTERVAL)	☐ ereignisgesteuerte Aufgabe (EVENT)	☐ Unterbrechungsaufgabe

Hinweis:
Stellen Sie sicher, dass Sie die CAA-Speicherbibliothek in Ihr SPS-Projekt aufgenommen haben, um eine fehlerfreie Erstellung Ihres Codes zu gewährleisten.

2.1.6 Ein- und Ausgänge

Bezeichner **Datentyp**	**Beschreibung**
Achse *AXIS_REF*	Bezug zur angeschlossenen Antriebsachse Bei einer i950-Anwendung weisen Sie diesem Signal immer die *Motion_Axis* zu.
ascParReference *ARRAY [*] OF L_ICIA_sc93ParReference*	Parameter-Entsprechungsliste Diese Liste definiert die Entsprechung zwischen 9300-Codes und i950-Indizes. Da Parameterwerte in Indizes gespeichert sind, die bei 9300 und i950 unterschiedliche Nummern haben, ermöglicht die Liste … • … einen 9300-Code mit einem i950-Index zu verknüpfen • … einen Skalierungsfaktor für Zähler/Nenner zwischen dem 9300-Parameterwert und dem i950-Indexwert zu berücksichtigen Eine detaillierte Übersicht über die Struktur *ascParReference* finden Sie im Kapitel „2.1.7 ” (nächste Seite).

2.1.7 Eingaben „ “

Bezeichner **Datentyp**	**Beschreibung**
eFieldBusType *L_ICIA_eFieldBusType*	Typ des Feldbusses Standardmäßig ist dieses Signal auf 1 („PROFIBUS_2133“) gesetzt. Bislang wird diese Variable im Funktionsbaustein **L_ICIA_PROFIBUS_Base** nicht verwendet, da die PROFIBUS-Funktionsbausteine nur die PROFIBUS-Kommunikation unterstützen. **Hinweis:** In Zukunft könnte der Eingang die Unterstützung verschiedener Feldbussysteme ermöglichen, die Lenze für die Serie 9300 anbietet, wie z. B. CAN, INTERBUS, …

Benutzerdefinierte Variablenstruktur *L_ICIA_sc93ParReference*

Die Struktur dient zur Definition der Parameterzuordnung auf der PROFIBUS-Ebene und der i950-Ebene. Folgende Elemente sind Teil dieser Variablenstruktur:

Bezeichner Datentyp	Beschreibung
wCode *WORD*	Codenummer des 9300-Servoantriebs **Hinweis:** Die 9300-Codenummer ergibt sich aus der Subtraktion des Indexwerts (Byte 3 und 4 des Parameterkanals) von einem festen Wert von 24575 (0x5FFF).
bySubCode *BYTE*	Subcode-Nummer des Servoantriebs 9300
wIndex *WORD*	entsprechende Indexnummer des i950-Servoantriebs
bySubIndex *BYTE*	Unterindexnummer des i950-Servoantriebs
bySize *BYTE*	Datengröße des Indexwerts des i950 **Hinweis:** Diese Information ist erforderlich, da die Datengröße des Indexwerts des i950 nicht unbedingt mit der Datengröße des 9300-Codewerts übereinstimmt.
diNum *diDen* *BYTE*	Skalierungsfaktor zwischen dem 9300-Codewert und dem Indexwert des i950 (aufgeteilt in Zähler-/Nennerwerte) Die Werte für den Skalierungszähler/Nenner können wie auf der nächsten Seite gezeigt ermittelt werden. Weitere Details finden Sie im Kapitel „2.1.2 “.

Hinweis:

Ein Anwendungsbeispiel finden Sie im Anhang im Kapitel „2.1.2 ”.

Beispiel: Berechnung der Skalierungswerte für Zähler/Nenner:

Die tatsächliche Motordrehzahl 0x606C:000 des i950-Servoantriebs sollte über den Parameterkanal gelesen und im 9300-Format des Codes C0051/000 an die SPS zurückgegeben werden:

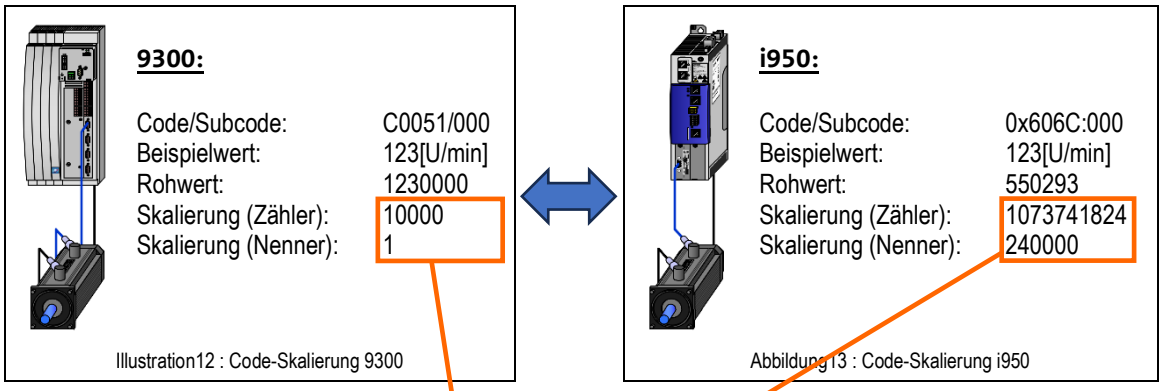


Illustration12 : Code-Skalierung 9300

Abbildung13 : Code-Skalierung i950

Aus dem obigen Beispiel lassen sich die Gesamtwerte für Zähler/Nenner *diNum* und *diDen* berechnen:

$$\frac{diNum}{diDen} = \frac{\frac{10000}{1}}{\frac{1073741824}{240000}} = \frac{2400000000}{1073741824} = \frac{1171875}{524288}$$

2.1.8 Ausgaben

Identifikator Datentyp	Beschreibung		
scStateMachine *L_ICIA_scStateMachine*	Daten der Kommunikationszustandsmaschine Dieser Wert muss mit den entsprechenden Eingangs-/Ausgangsvariablen der Funktionsbausteine **L_ICIA_PROFIBUS_In** und **L_ICIA_PROFIBUS_Out** verbunden werden, um einen konsistenten Betrieb der PROFIBUS-Funktionsbausteine zu gewährleisten. Eine detaillierte Beschreibung finden Sie auf der nächsten Seite.		
xInit *BOOL*	Statussignal: Initialisierung der GSD/GSE-Konfiguration läuft		
	FALSE	GSD/GSE-Konfiguration ohne Fehler abgeschlossen	
	TRUE	GSD/GSE-Konfiguration läuft/wurde mit Fehlern abgeschlossen	
xError *BOOL*	Statussignal: Fehler während der GSD/GSE-Konfiguration		
	FALSE	kein Fehler während der GSD/GSE-Konfiguration.	
	TRUE	Während der GSD/GSE-Konfiguration ist ein Fehler aufgetreten: • Initialisierungssequenz kann nicht beendet werden – Statussignal *xInit* bleibt auf TRUE. • Weitere Informationen finden Sie in der *wError*-Ausgabe.	
eErrorID *WORD*	Aktuelle Fehler-ID:		
	0:	kein Fehler aktiv	
	110:	GSD/GSE-Konfiguration konnte nicht identifiziert werden – bitte wählen Sie eine GSD/GSE-Konfiguration aus der Liste im Kapitel aus.4.1	
	Hinweise: -		

Benutzerdefinierte Variablenstruktur *L_ICIA_scStateMachine*

Diese variable Struktur umfasst allgemeine Daten der Feldbuskommunikation, abhängig vom aktiven Feldbustyp:

Kennung Datentyp	Beschreibung
eFieldBusType *L_ICIA_eFieldBusType*	Aktuelle Fehler-ID: *PROFIBUS_2133*: Die Funktionsbausteine **L_ICIA_PROFIBUS** verhalten sich wie das PROFIBUS-Modul EMF2133IB für 9300. **Hinweise:** Bislang werden nur „PROFIBUS_2131" und „PROFIBUS_2133" unterstützt.
byGsdConfig *BYTE*	Nummer der aktiven GSD/GSE-Konfiguration, die während der Kommunikationsinitialisierung gefunden wurde Eine vollständige Übersicht über alle möglichen GSD/GSE-Konfigurationen finden Sie im Kapitel „4.1 ".
byGsdGroup *BYTE*	Gruppe der aktiven GSD/GSE-Konfiguration der aktiven GSD/GSE-Konfiguration, die während der Kommunikationsinitialisierung eingerichtet wurde Eine vollständige Übersicht über alle möglichen GSD/GSE-Konfigurationen finden Sie im Kapitel „4.1 ".
byGsdGroup *BYTE*	Aktuelle Fehler-ID: 1: kein Parameterkanal / Prozessdaten (Drivecom-Steuerung) 2: Konsistente Drivecom-Parameterkanal-/Prozessdaten (Drivecom-Steuerung) 3: Konsistenter Drivecom-Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung) 4: Drivecom-Parameterkanal / Prozessdaten (Drivecom-Steuerung) 5: Drivecom-Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung) 6: kein Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung) 7: kein Parameterkanal / Prozessdaten (Lenze-Gerätesteuerung) 8: konsistenter Drivecom-Parameterkanal / Prozessdaten (Lenze-Gerätesteuerung) 9: konsistenter Drivecom-Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung) 10: Drivecom-Parameterkanal / Prozessdaten (Lenze-Gerätesteuerung) 11: Drivecom-Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung) 12: kein Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung) **Hinweise:** Eine vollständige Übersicht über alle möglichen GSD/GSE-Konfigurationen finden Sie im Kapitel „4.1 ".
byPzdSize *BYTE*	Größe der Prozessdaten (PZD), skaliert in [Byte]
wDrivecomCtrl *BYTE*	Drivecom-Steuerwort Diese Variable wird nur in einer Konfiguration mit Drivecom-Prozessdatenkommunikation (*byGsdGroup* = 1 … 6) verwendet. Die Bedeutung der einzelnen Steuerbits von *wDrivecomCtrl* wird im Anhang im Kapitel „4.4 " erläutert.
wDrivecomStat *BYTE*	Drivecom-Statuswort Diese Variable wird nur in einer Konfiguration mit Drivecom-Prozessdatenkommunikation (*byGsdGroup* = 1 … 6) verwendet. Die Bedeutung der einzelnen Steuerbits von *wDrivecomStat* wird im Anhang im Kapitel „4.5 " erläutert.
eDrivecomState *L_ICIA_eDrivecomState*	Aktueller Zustand der Drivecom-Zustandsmaschine: 0: NOT_READY_TO_SWITCH_ON 32: EINSCHALTSPERRE 1: BEREIT_ZUM_EINSCHALTEN 3: EINGESCHALTET 23: SCHNELLSTOP_AKTIV 7: BETRIEB_AKTIVIERT 15: FEHLER_REAKTION_AKTIV 8: FEHLER **Hinweise:** Diese Variable wird nur in einer Konfiguration mit Drivecom-Prozessdatenkommunikation (*byGsdGroup* = 1 … 6) verwendet. Die Drivecom-Zustandsmaschine ist in den Kapiteln2.2.2 und2.3.2 dargestellt.
xInit *BOOL*	Statussignal: Initialisierung der GSD/GSE-Konfiguration läuft FALSE: GSD/GSE-Konfiguration ohne Fehler abgeschlossen TRUE: GSD/GSE-Konfiguration läuft/wurde mit Fehlern abgeschlossen **Hinweis:** Dieses Signal spiegelt das Ausgangssignal *L_ICIA_PROFIBUS_Base.xInit* wider.
xError *BOOL*	Statussignal: Fehler während der GSD/GSE-Konfiguration FALSE: kein Fehler während der GSD/GSE-Konfiguration. TRUE: Während der GSD/GSE-Konfiguration ist ein Fehler aufgetreten: • Die Initialisierungssequenz kann nicht beendet werden – das Statussignal *xInit* bleibt auf TRUE. • Weitere Informationen finden Sie unter dem Ausgang *L_ICIA_PROFIBUS_Base.wError*. **Hinweis:** Dieses Signal spiegelt das Ausgangssignal *L_ICIA_PROFIBUS_Base.xError* wider.

Kennung Datentyp	Beschreibung	
adwRawDataIn *ARRAY [0..15] OF DWORD*	Rohdaten an der Feldbus-Schnittstelle Das variable Datenarray ist eine Kopie der Feldbus-Roh-Eingangsdaten, die am Eingang *adwFieldBusIn* des Funktionsbausteins **L_ICIA_PROFIBUS_In** empfangen wurden.	
adwRawDataOut *ARRAY [0..15] OF DWORD*	Rohausgangsdaten auf der Feldbus-Schnittstelle Das variable Datenarray ist eine Kopie der Feldbus-Rohausgangsdaten, die am Ausgang *adwFieldBusOut* des Funktionsblocks **L_ICIA_PROFIBUS_Out** generiert werden.	
AxisState *MC_ReadAxisInfo*	Diese Struktur enthält wichtige Statussignale des i950-Antriebs:	
	LimitSwitchPos:	Positiver Endschalter hat ausgelöst (d. h. auf *L_TF2P_SpeedControl-Base1.scCtrlBasicMotion.xHWLimitPos*)
	LimitSwitchNeg:	Negativer Endschalter hat ausgelöst (d. h. auf *L_TF2P_SpeedControl-Base1.scCtrlBasicMotion.xHWLimitNeg*)
	Simulation:	Achse wird im virtuellen Modus betrieben Bei einer i950-Achse ist dieses Signal immer FALSE.
	CommunicationReady:	Die Motion-Bus-Kommunikationsschnittstelle zwischen Achstreiber (*AXIS_REF*) und Motorsteuerung ist in Betrieb Bei einer i950-Achse ist dieses Signal immer TRUE.
	ReadyForPowerOn:	Der Antrieb ist bereit für die Einschaltung (d. h. über das Steuersignal *L_TF2P_SpeedControlBase1.xEnableOperation*). Dieser Signalzustand umfasst die folgenden Zustände: • Antrieb ist fehlerfrei • kein STO-Befehl ist aktiv (Safe Torque Off) • Die Gleichstrom-Busspannung ist eingeschaltet
	PowerOn:	i950-Antrieb ist eingeschaltet (gleicher Status wie *L_TF2P_SpeedControl-Base1.xOperationEnabled*)
	IsHomed:	Die Nullposition des Messsystems des i950-Antriebs ist bekannt
	AxisError:	Fehler im Achstreiber (*AXIS_REF*)
	AxisWarning:	Warnung im Achstreiber (*AXIS_REF*)
	Antriebsfehler:	Fehler in der Motorsteuerung des Wechselrichters
	DriveWarning:	Warnung in der Motorsteuerung des Umrichters
	SWLimitSwitchPos:	Positive Software-Begrenzung wurde ausgelöst
	SWLimitSwitchNeg:	Negative Software-Begrenzung ausgelöst
	ReadyForMotion:	Antrieb ist bereit für die Entgegennahme von Bewegungsbefehlen Dieser Signalzustand umfasst die folgenden Zustände: • Antrieb ist freigegeben • Antrieb ist fehlerfrei • eine Motorbremse (falls vorhanden) hat sich geöffnet
	STOActive:	STO-Befehl ist aktiv (Safe Torque Off)
	VoltageEnabled:	Die Gleichstrom-Busspannung ist eingeschaltet
	MotorMagnetised:	Magnetisierung des Motors abgeschlossen
	QSPApplActive:	Schnellstoppbefehl des Achsantriebs (*AXIS_REF*) ist aktiv
	QSPDriveActive:	Quickstop-Befehl der Motorsteuerung des Umrichters ist aktiv
	Weitere Informationen zu **MC_ReadAxisInfo** finden Sie in der Online-Hilfe zum »PLC Designer«.	

Achtung:

Die oben aufgeführten Variablen sind schreibgeschützt! Ändern Sie niemals eine dieser Variablen, da dies unvorhersehbare Folgen für die Feldbuskommunikation und das Verhalten des Antriebs haben kann!

2.2 Funktionsbaustein *L_ICIA_PROFIBUS_In*

Der Funktionsbaustein **L_ICIA_PROFIBUS_In** liest 16 Doppelwörter der Feldbus-Eingangsdaten in das Eingangsdatenarray *adwFieldBusIn*. Sobald eine gültige GSD/GSE-Konfiguration erkannt wurde (*scStateMachine.xInit* = FALSE), werden die Rohdaten des Eingangssignals *adwFieldBusIn* des Funktionsbausteins **L_ICIA_PROFIBUS_In** auf ...

- Prozessdaten PZD
- Parameterdaten PAR (optional, falls ausgewählt, siehe Kapitel „2.1.2 “)

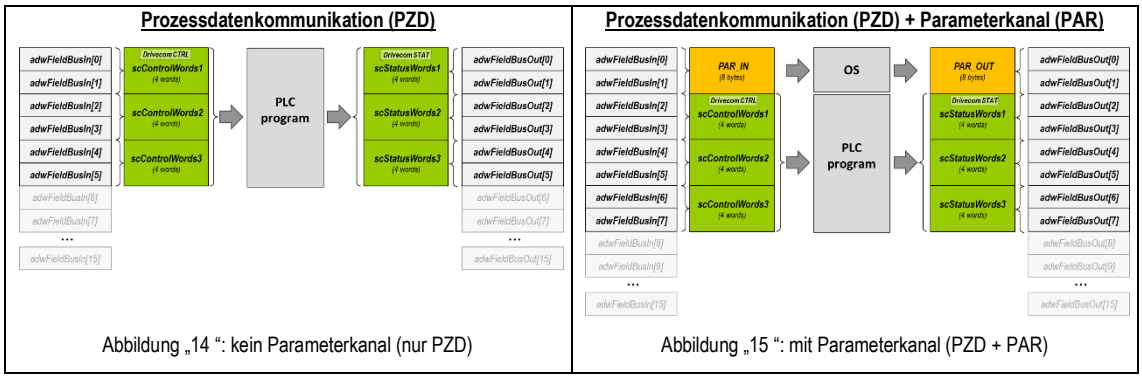


Abbildung „14 “: kein Parameterkanal (nur PZD)

Abbildung „15 “: mit Parameterkanal (PZD + PAR)

Hinweis:

Das PROFIBUS-Steckmodul i950 verarbeitet bis zu 16 Doppelwörter an Eingangsdaten. Der Funktionsbaustein **L_ICIA_PROFIBUS_In** verarbeitet nur die Doppelwörter 0 bis 7. Die Doppelwörter 8 bis 15 werden bei der Auswertung der Feldbus-Rohdaten nicht berücksichtigt.

Weisen Sie dem Eingangssignal *L_ICIA_PROFIBUS_In.adwFieldBusIn* jedoch immer ein Datenarray *ARRAY [0..15] OF DWORD* zu.

2.2.1 Prozessdaten (PZD)

Der Prozessdatenaustausch ist in jedem Fall Teil der Feldbuskommunikation. Der Funktionsbaustein **L_ICIA_PROFIBUS_In** verarbeitet die Prozesseingangsdaten des Feldbussystems und wandelt die auf *adwFieldBusIn* empfangenen Rohdaten in die aus der Geräteserie 8200/9300 bekannten Datenstrukturen um.

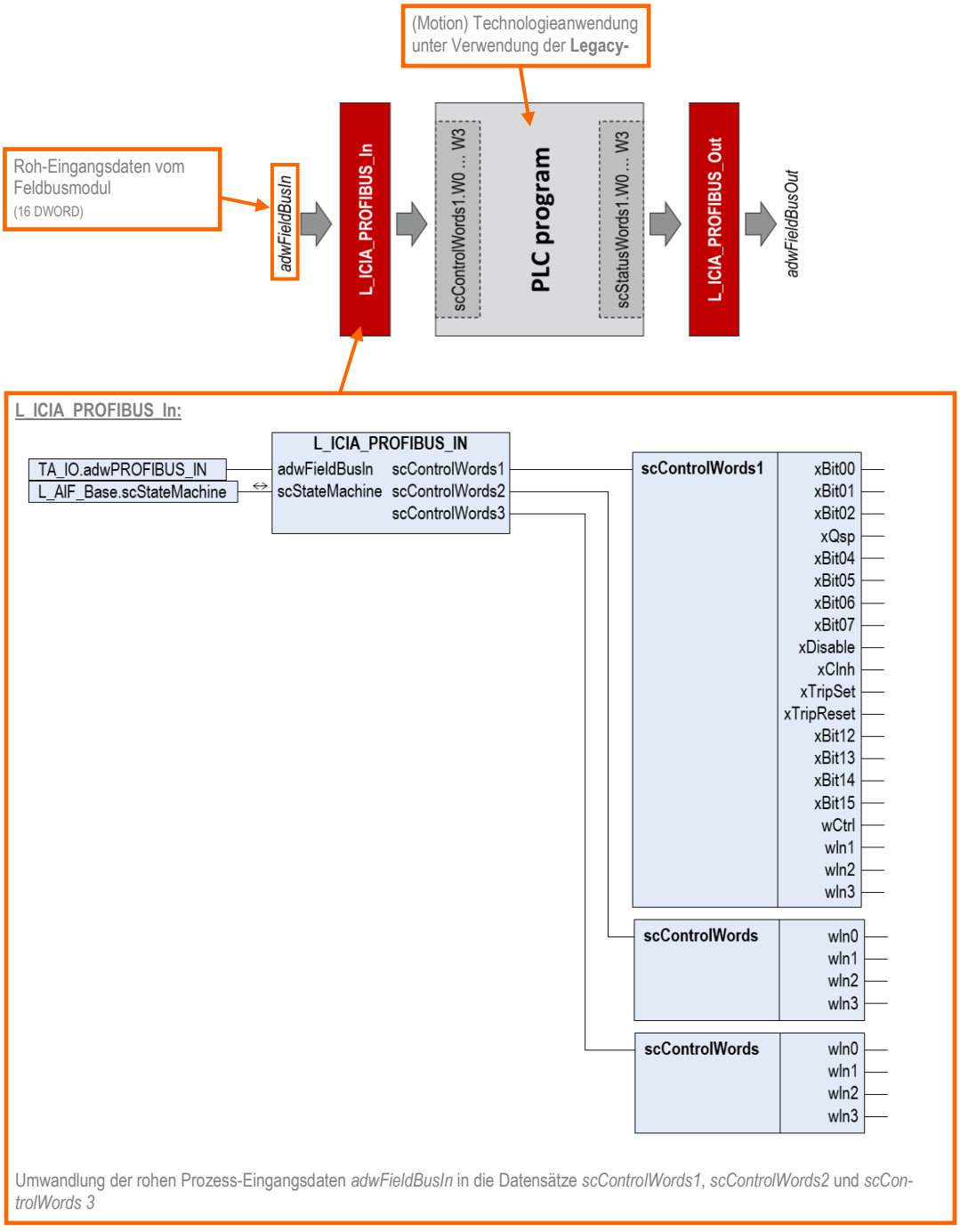


Abbildung16 : Prinzip der Verarbeitung von Prozesseingangsdaten / detaillierte Signalliste der *scControlWords*-Schnittstellen des Funktionsblocks **L_ICIA_PROFIBUS_In**

2.2.2 Drivecom-Zustandsmaschine

Je nach GSD/GSE-Konfiguration wird das erste Prozess-Eingangsdatenwort *scControlWords1.wCtrl* über die Drivecom-Zustandsmaschine verarbeitet:

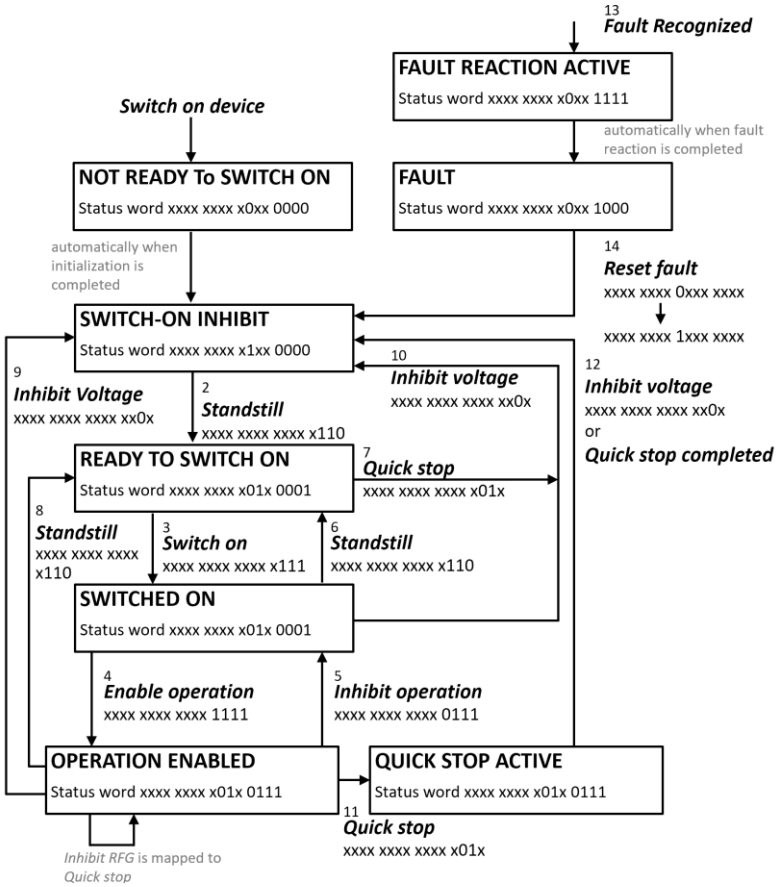


Abbildung17 : Flussdiagramm der Drivecom-Zustandsmaschine (wirkt sich auf das Steuer-/Statuswort 1 aus)

Der aktuelle Zustand der Drivecom-Zustandsmaschine wird in der Variablen *scStateMachine.eDrivecomState* angezeigt.

2.2.3 Inkompatibilitätsliste

Die folgenden Funktionen sind im Funktionsbaustein **L_ICIA_PROFIBUS_In** nicht implementiert:

- Die Ausgabe *scControlWords1.wCtrl.xTripSet* findet keine entsprechende Funktion in den FAST-Technologiemodulen. Der Anwender kann dieses Signal auswerten, um einen benutzerdefinierten Fehler zu setzen.
- Ein Unterspannungszustand während des Antriebsbetriebs führt zu einem Fehler, da die PLCopen-Zustandsmaschine verletzt wird. Bei 9300 führte ein Unterspannungszustand während des Antriebsbetriebs nur zu einer Meldung.
- Der STO-Befehl von i950 muss freigegeben werden, um das gleiche Verhalten der Drivecom-Zustandsmaschine wie bei 9300 zu erreichen. Wenn der STO-Befehl von i950 aktiv ist, bleibt die Drivecom-Zustandsmaschine im Zustand *„Switch-On Inhibited" (Einschalten gesperrt*). Bei Verwendung von GSD-Konfigurationen mit Lenze-Gerätesteuerung (AR) hält der STO-Befehl das *xDisable*-Steuersignal aktiv, sodass der Antrieb nicht aktiviert werden kann.
- Der Funktionsbaustein **L_ICIA_PROFIBUS_In** unterstützt die folgenden Gerätesteuerungsmethoden:
    - Drivecom
    - Lenze-Gerätesteuerung (AR)

  Die PROFIdrive-Steuerungsmethode wird nicht unterstützt.

2.2.4 Schnittstelle

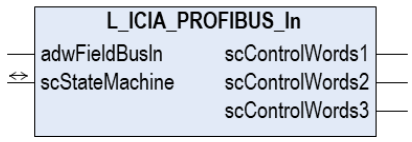


Abbildung18 : Schnittstelle des Funktionsbausteins **L_ICIA_PROFIBUS_In**

2.2.5 Aufgabeninformation

Aufruf möglich von:	☒ Freilauf-Task	☒ zeitgesteuerter Aufgabe (INTERVAL)	☐ ereignisgesteuerte Aufgabe (EVENT)	☐ Unterbrechungsaufgabe

Hinweis:
Stellen Sie sicher, dass Sie die *CAA*-Speicherbibliothek in Ihr SPS-Projekt aufgenommen haben, um eine fehlerfreie Erstellung Ihres Codes zu gewährleisten.

2.2.6 Ein- und Ausgänge

Bezeichner **Datentyp**	**Beschreibung**
scStateMachine *L_ICIA_scStateMachine*	Daten der Kommunikationszustandsmaschine Verbinden Sie den entsprechenden Ausgang *scStateMachine* des Funktionsbausteins **L_ICIA_PROFIBUS_Base,** um einen konsistenten Betrieb der PROFIBUS-Funktionsbausteine sicherzustellen. Eine detaillierte Beschreibung dieser variablen Struktur finden Sie im Kapitel „2.1.8 “.

2.2.7 Eingänge

Kennung **Datentyp**	**Beschreibung**
adwFieldBusIn *ARRAY [0..15] OF DWORD*	Eingabe der Feldbus-Rohdaten Diese Werte können direkt den Eingangsdaten der Feldbus-IO-Schnittstelle zugeordnet werden.

2.2.8 Ausgänge

Kennung **Datentyp**	**Beschreibung**
scControlWords1 *L_ICIA_scControlWords1*	AIF-Feldbus-Eingangsdaten (erste Gruppe) Die Werte bestehen aus einer Datenstruktur mit vier Wörtern, die der Struktur des AIF-IN-Systemblocks des Servoumrichters 9300 entspricht. Eine detaillierte Beschreibung finden Sie auf der nächsten Seite.
scControlWords2 *L_ICIA_scControlWords*	AIF-Feldbus-Eingangsdaten (zweite Gruppe) Die Werte bestehen aus einer Datenstruktur mit vier Wörtern, die der Struktur des AIF-IN-Systemblocks des Servoumrichters 9300 ServoPLC folgt. Eine detaillierte Beschreibung finden Sie auf den nächsten Seiten.
scControlWords3 *L_ICIA_scControlWords*	AIF-Feldbus-Eingangsdaten (dritte Gruppe) Die Werte umfassen eine Datenstruktur mit vier Wörtern, die der Struktur des AIF-IN-Systemblocks des 9300 ServoPLC-Wechselrichters folgt. Eine detaillierte Beschreibung finden Sie auf den nächsten Seiten.

Benutzerdefinierte Variablenstruktur *L_ICIA_scControlWords1*

Diese Struktur implementiert die aus der Servoumrichter-Serie 9300 bekannte AIF-IN-Schnittstelle. Sie umfasst die folgenden Elemente:

Bezeichner Datentyp	Beschreibung
xBit00 *BIT*	Bit 0 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit01 *BIT*	Bit 1 des Steuerworts FALSE: Steuerfunktion deaktiviert WAHR: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit02 *BIT*	Bit 2 des Steuerworts FALSE: Steuerfunktion deaktiviert WAHR: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xQsp *BIT*	Bit 3 des Steuerworts: Schnellstopp aktivieren FALSE: Schnellstopp nicht aktiviert TRUE: Schnellhalt aktiviert **Hinweis:** Dieses Bit muss mit einem Schnellstoppbefehl in der Anwendung verbunden sein (d. h. durch die Funktionsbausteine **MC_Stop**, **L_MC1P_SetQuickStopAppl**, … implementiert werden).
xBit04 *BIT*	Bit 4 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit05 *BIT*	Bit 5 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit06 *BIT*	Bit 6 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit07 *BIT*	Bit 7 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xDisable *BIT*	Bit 08 des Steuerworts: Antrieb deaktivieren FALSE: Antrieb nicht deaktivieren (xCInh=FALSE führt zum Einschalten des Laufwerks) TRUE: Antrieb deaktivieren (xCInh=FALSE hat keine Auswirkung) **Hinweise:** - Verwenden Sie dieses Bit, um den Betrieb des Antriebs zu sperren. Bei xDisable=TRUE muss der Antrieb ausgeschaltet bleiben, auch wenn *xCInh* auf FALSE steht. - Wenn *xDisable* auf TRUE gesetzt ist, bleibt der Status „Antrieb bereit" auf FALSE.
xCInh *BIT*	Bit 09 des Steuerworts: Sperren des Antriebscontrollers FALSE: Antriebssteuerung aktiviert TRUE: Antriebssteuerung gesperrt **Hinweise:** - Das Bit wird zum Einschalten des Antriebs verwendet (d. h. mithilfe des Funktionsblocks **MC_Power**). - Wenn *xDisable* auf TRUE gesetzt ist, hat das Steuerbit *xCInh* keine Wirkung.

Bezeichner Datentyp	Beschreibung
xTripSet *BIT*	Bit 10 des Steuerworts: Benutzerfehler am Antrieb setzen FALSCH: Benutzerfehler wird ausgelöst WAHR: Es wird kein Benutzerfehler ausgelöst **Hinweis:** Da es im Betriebssystem des i950 keine entsprechende Funktion gibt, hat das xTripSet-Bit keine praktische Bedeutung.
xTripReset *BIT*	Bit 11 des Steuerworts: Befehl zum Zurücksetzen des Fehlers FALSE=>TRUE Befehl zum Zurücksetzen des Fehlers **Hinweise:** - Das Bit wird verwendet, um einen Fehler am Antrieb zurückzusetzen (d. h. mithilfe des Funktionsblocks **MC_Reset**). - Das Zurücksetzen eines Fehlers funktioniert nur, wenn die Ursache des Fehlers nicht mehr vorliegt.
xBit12 *BIT*	Bit 12 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit13 *BIT*	Bit 13 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit14 *BIT*	Bit 14 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit15 *BIT*	Bit 15 des Steuerworts FALSE: Steuerfunktion deaktiviert TRUE: Steuerfunktion aktiviert **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
wCtrl *WORD*	Steuerwort Dieses Steuerwort spiegelt die oben aufgeführten 16 Steuerbits im WORD-Format wider.
wIn1 *WORD*	Eingabe einer 16-Bit-Ganzzahl In der Regel wird das zweite WORD in scControlWords1 als Drehzahlsollwert des Antriebs interpretiert, skaliert in [%] (0 … 16384 = 0,0 … 100,0[%]). Die Definition der Bedeutung in der Anwendung obliegt jedoch dem Anwender.
wIn2 *WORD*	Eingabe eines freien 16-Bit-WORD-Werts
wIn3 *WORD*	Eingabe eines freien 16-Bit-WORD-Werts

Tipp:

Möchten Sie *scControlWords1.wIn3* zu einem 32-Bit-Wert zusammenführen? Die Funktion **PackWordsToDword**[6] bietet diese Funktion. Verwenden Sie sie wie folgt:

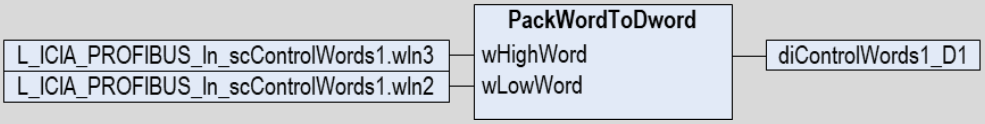


Abbildung19 : Umwandlung von zwei 16-Bit-*WORD*-Werten in einen 32-Bit-*DWORD*-Wert

[6] enthalten die *CAA*-Speicherbibliothek

Benutzerdefinierte Variablenstruktur *L_ICIA_scControlWords*

Diese Struktur implementiert die erweiterte AIF-IN-Schnittstelle, die aus der Wechselrichterreihe 9300 ServoPLC bekannt ist. Sie wird auf die Objekte scControlWords2 und scControlWords3 angewendet und umfasst die folgenden Elemente:

Bezeichner Datentyp	Beschreibung
wIn0 *WORD*	Eingabe eines freien 16-Bit-WORD-Werts
wIn1 *WORD*	Eingabe eines freien 16-Bit-WORD-Werts
wIn2 *WORD*	Eingabe eines freien 16-Bit-WORD-Werts
wIn3 *WORD*	Eingabe eines freien 16-Bit-WORD-Wertes

2.3 Funktionsbaustein *L_ICIA_PROFIBUS_Out*

Der Funktionsbaustein **L_ICIA_PROFIBUS_Out** liest die aus der Geräteserie 8200/9300 bekannte AIF-Datenstruktur und überträgt deren Informationen auf die 16 Feldbus-Ausgangs-Doppelwörter eines Datenarrays. Sobald eine gültige GSD/GSE-Konfiguration erkannt wurde (*scStateMachine.xInit* = FALSE), werden die folgenden Daten auf das Ausgangs-Datenarray *adwFieldBusOut* abgebildet:

- Prozessdaten PZD aus den AIF-OUT-Objekten
- Parameterdaten PAR (optional, falls ausgewählt, siehe Kapitel „2.1.2 ")

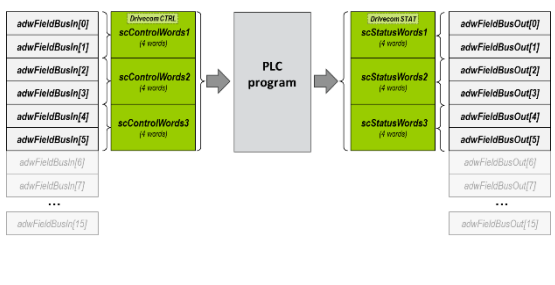


Abbildung „20 ": kein Parameterkanal (nur PZD)

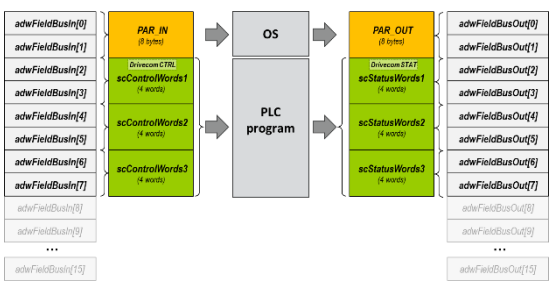


Abbildung „21 ": mit Parameterkanal (PZD + PAR)

Hinweis:

Das PROFIBUS-Steckmodul i950 verarbeitet bis zu 16 Doppelwörter an Ausgangsdaten. Der Ausgangsdatenbereich des Funktionsbausteins **L_ICIA_PROFIBUS_Out** (Ausgang *adwFieldBusOut*) umfasst den gesamten Bereich von 16 Doppelwörtern, auch wenn nur die Doppelwörter 0 bis 7 verwendet werden.

2.3.1 Prozessdaten (PZD)

In jedem Fall ist der Prozessdatenaustausch Teil der Feldbuskommunikation. Der Funktionsbaustein **L_ICIA_PROFIBUS_Out** generiert die Rohdaten auf *adwFieldBusOut* aus den aus der Serie 8200/9300 bekannten AIF-OUT-Objekten.

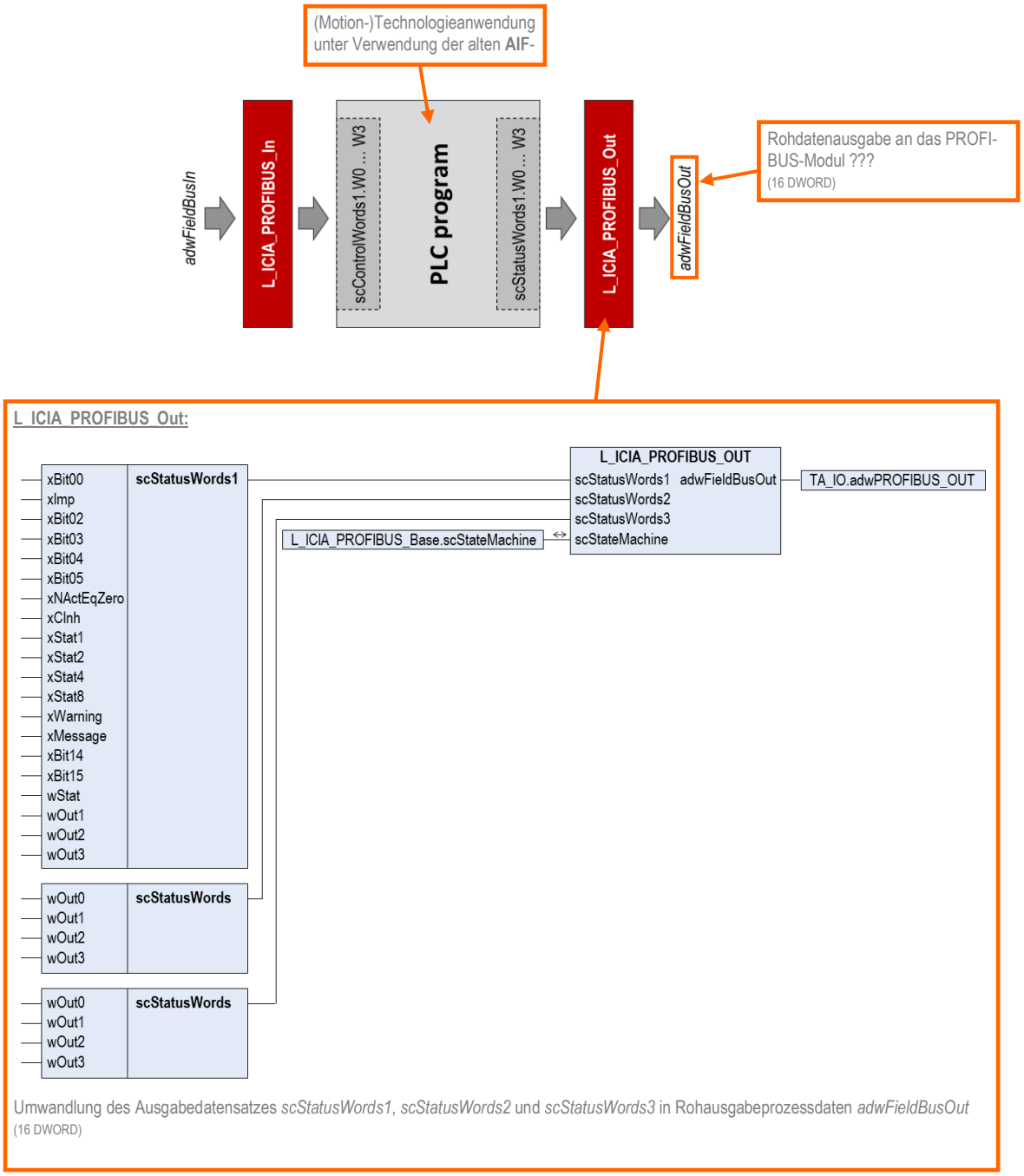


Abbildung22 : Prinzip der Prozessausgangsdatenverarbeitung / detaillierte Signalliste der *scStatusWords*-Schnittstellen des Funktionsbausteins **L_ICIA_PROFIBUS_In**

Tipp:
Verwenden Sie den benutzerdefinierten Funktionsbaustein **L_STAT,** um die Statussignale auf *L_ICIA_PROFIBUS_Out.xStat1* ... *L_ICIA_PROFIBUS_Out.xStat8* zu generieren.

2.3.2 Drivecom-Zustandsmaschine

Abhängig von der GSD/GSE-Konfiguration wird das erste Prozessausgangsdatenwort *scStatusWords1.wStat* über die Drivecom-Zustandsmaschine verarbeitet:

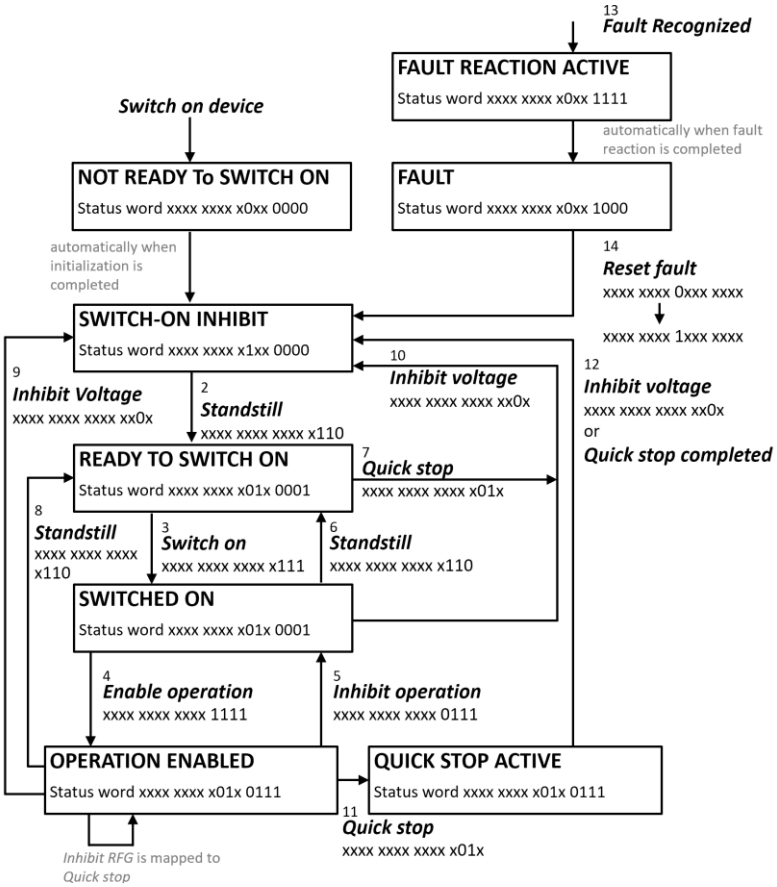


Abbildung23 : Flussdiagramm der Drivecom-Zustandsmaschine (wirkt sich auf Steuer-/Statuswort 1 aus)

Der aktuelle Zustand der Drivecom-Zustandsmaschine wird in der Variablen *scStateMachine.eDrivecomState* angezeigt.

2.3.3 Inkompatibilitätsliste

Die folgenden Funktionen sind im Funktionsbaustein **L_ICIA_PROFIBUS_Out** nicht implementiert:

- Die Statusbits *DCTRL-STAT*1*, ... *DCTRL-STAT*8* umfassen nicht den gesamten Umfang der 9300 Zustände. Die rot markierten Zustände werden nicht unterstützt:

Wert	*DCTRL-STAT*8*	*DCTRL-STAT*4*	*DCTRL-STAT*2*	*DCTRL-STAT*1*	Anmerkung
0	**0**	**0**	**0**	**0**	Initialisierung nach dem Anschließen der Versorgungsspannung
1	0	0	0	1	Sperrmodus, Neustartschutz ist aktiv C0142
3	**0**	**0**	**1**	**1**	Antrieb befindet sich im Controller-Sperrmodus
4	**0**	**1**	**0**	**0**	Flug-Neustart aktiv
5	**0**	**1**	**0**	**1**	DC-Bremse aktiv
6	**0**	**1**	**1**	**0**	Controller aktiviert
7	**0**	**1**	**1**	**1**	Die Freigabe einer Überwachungsfunktion führte zu einer „Meldung“.
8	**1**	**0**	**0**	**0**	Die Freigabe einer Überwachungsfunktion führte zu einer „Auslösung“.
10	**1**	**0**	**1**	**0**	Die Freigabe einer Überwachungsfunktion führte zu einem „FAIL-QSP”.
15	**1**	**1**	**1**	**1**	Kommunikationsfehler (PROFIBUS-Kommunikationsmodul⇔ -Frequenzumrichter)

- Gemäß PLCopen führt eine Unterspannung am DC-Bus zu einem Fehler statt zu einer Meldung. Vor dem Neustart des Antriebs muss der Benutzer den Antriebsfehler zurücksetzen.
- Der Funktionsbaustein **L_ICIA_PROFIBUS_In** unterstützt die folgenden Gerätesteuerungsmethoden:
  - Drivecom
  - Lenze-Gerätesteuerung (AR)

  Die Steuerungsmethode PROFIdrve wird nicht unterstützt.

2.3.4 Schnittstelle

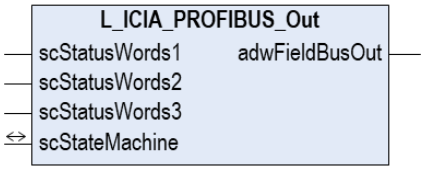


Abbildung24 : Schnittstelle des Funktionsblocks **L_ICIA_PROFIBUS_Out**

2.3.5 Aufgabeninformation

Aufruf möglich aus:	☒ Freilauf-Task	☒ zeitgesteuerter Aufgabe (INTERVAL)	☐ ereignisgesteuerte Aufgabe (EVENT)	☐ Unterbrechungsaufgabe

Hinweis:
Stellen Sie sicher, dass Sie die *CAA*-Speicherbibliothek in Ihr SPS-Projekt aufgenommen haben, um eine fehlerfreie Erstellung Ihres Codes zu gewährleisten.

2.3.6 Ein- und Ausgänge

Bezeichner Datentyp	Beschreibung
scStateMachine *L_ICIA_scStateMachine*	Daten der Kommunikationszustandsmaschine Verbinden Sie den entsprechenden Ausgang *scStateMachine* des Funktionsblocks **L_ICIA_PROFIBUS_Base,** um einen konsistenten Betrieb der AIF-Funktionsblöcke sicherzustellen. Eine detaillierte Beschreibung dieser variablen Struktur finden Sie im Kapitel „2.1.8 “.

2.3.7 Eingänge

Bezeichner Datentyp	Beschreibung
scStatusWords1 *L_ICIA_scStatusWords1*	AIF-Feldbus-Ausgangsdaten (erste Gruppe) Die Werte bestehen aus einer Datenstruktur mit vier Wörtern, die der Struktur des AIF-OUT-Systemblocks des Servoumrichters 9300 entspricht. Eine detaillierte Beschreibung finden Sie auf der nächsten Seite.
scStatusWords2 *L_ICIA_scStatusWords*	AIF-Feldbus-Ausgangsdaten (zweite Gruppe) Die Werte bestehen aus einer Datenstruktur mit vier Wörtern, die der Struktur des AIF-OUT-Systemblocks des 9300 ServoPLC-Umrichters folgt. Eine detaillierte Beschreibung finden Sie auf den nächsten Seiten.
scStatusWords3 *L_ICIA_scStatusWords*	AIF-Feldbus-Ausgangsdaten (dritte Gruppe) Die Werte bestehen aus einer Datenstruktur mit vier Wörtern, die der Struktur des AIF-OUT-Systemblocks des 9300 ServoPLC-Wechselrichters folgt. Eine detaillierte Beschreibung finden Sie auf den nächsten Seiten.

2.3.8 Ausgänge

Kennung Datentyp	Beschreibung
adwFieldBusOut *ARRAY [0..15] OF DWORD*	Ausgabe der Feldbus-Rohdaten Diese Werte können direkt den Ausgabedaten der Feldbus-IO-Schnittstelle zugeordnet werden.

Benutzerdefinierte Variablenstruktur *scStatusWords1*

Diese Struktur implementiert die aus der Servoumrichter-Serie 9300 bekannte Schnittstelle AIF-OUT1. Sie umfasst die folgenden Elemente:

<table>
<tr><th>Bezeichner
Datentyp</th><th>Beschreibung</th></tr>
<tr><td>xBit00
BIT</td><td>Bit 0 des AIF-OUT-Statusworts
FALSE: Status inaktiv
TRUE: Status aktiv
<u>Hinweis:</u> Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.</td></tr>
<tr><td>xImp
BIT</td><td>Bit 1 des AIF-OUT-Statusworts: Impulshemmung aktiv
FALSE: Die Leistungsstufe des Antriebs ist aktiv und versorgt den Motor mit Spannung/Strom.
TRUE: Die Leistungsstufe des Antriebs ist inaktiv und es wird kein Strom an den Motor angelegt.
<u>Hinweis:</u> Dieses Bit muss mit dem entsprechenden Signal in der Anwendung verbunden sein (z. B. über das Statussignal xImpActive des Funktionsblocks L_TB2P_AxisInterface).</td></tr>
<tr><td>xBit02
BIT</td><td>Bit 2 des AIF-OUT-Statusworts
FALSE: Status inaktiv
TRUE: Status aktiv
<u>Hinweis:</u> Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei verbunden werden.</td></tr>
<tr><td>xBit03
BIT</td><td>Bit 3 des AIF-OUT-Statusworts
FALSE: Status inaktiv
TRUE: Status aktiv
<u>Hinweis:</u> Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.</td></tr>
<tr><td>xBit04
BIT</td><td>Bit 4 des AIF-OUT-Statusworts
FALSE: Status inaktiv
TRUE: Status aktiv
<u>Hinweis:</u> Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.</td></tr>
<tr><td>xBit05
BIT</td><td>Bit 5 des AIF-OUT-Statusworts
FALSE: Status inaktiv
TRUE: Status aktiv
<u>Hinweis:</u> Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei verbunden werden.</td></tr>
<tr><td>xNActEqZero
BIT</td><td>Bit 6 des AIF-OUT-Statusworts: Antriebsdrehzahlsignal ist Null
FALSE: Antrieb bewegt sich (absolute Antriebsdrehzahl ist größer als das Drehzahl-Toleranzfenster)
TRUE: Antrieb steht still (absolute Antriebsgeschwindigkeit unterhalb des Geschwindigkeitstoleranzfensters)
<u>Hinweis:</u> Generieren Sie dieses Signal durch eine geeignete Logik (d. h. <code>(ABS(MCTRL_nNAct_v)<=scPar.wC0019_Nmin)</code>).</td></tr>
<tr><td>xCInh
BIT</td><td>Bit 7 des AIF-OUT-Statusworts: Antriebsregler sind gesperrt
FALSE: Positions-/Drehzahl-/Stromregelung ist aktiv
TRUE: Positions-/Drehzahl-/Stromregelung ist zurückgesetzt
<u>Hinweis:</u> Dieses Bit muss mit dem entsprechenden Signal in der Anwendung verbunden sein (z. B. über das Statussignal Status des Funktionsblocks MC_Power).</td></tr>
<tr><td>xStat1
xStat2
xStat4
xStat8
BIT</td><td>Bits 8 bis 11 des AIF-OUT-Statusworts: Anzeige des Antriebszustands
<table>
<tr><th>xStat8</th><th>xStat4</th><th>xStat2</th><th>xStat1</th><th></th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Initialisierung nach Anschluss der Versorgungsspannung</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Antrieb befindet sich im Status „Controller inhibit“</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Controller ist aktiviert</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>eine Überwachungsfunktion in einer „Meldung” ausgelöst</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Eine Überwachungsfunktion, die bei einem „Fehler“ ausgelöst wird</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>eine Überwachungsfunktion, die bei einem „FAIL-QSP” ausgelöst wird</td></tr>
</table>
<u>Hinweise:</u> Diese Bits müssen mit dem entsprechenden Signal in der Anwendung verbunden sein (z. B. über die Statussignale des Funktionsblocks L_TB2P_AxisInterface). Einige aus 9300 bekannte Zustände können möglicherweise nicht angezeigt werden (siehe Kapitel „2.3.3 “).</td></tr>
</table>

Kennung Datentyp	Beschreibung
xWarnung *BIT*	Bit 12 des AIF-OUT-Statusworts: Warnung aktiv FALSE: keine Antriebswarnung aktiv TRUE: Eine Antriebswarnung ist aktiv. **Hinweis:** Dieses Bit muss mit dem entsprechenden Signal in der Anwendung verbunden sein (z. B. über die Statussignale des Funktionsbausteins **MC_ReadAxisError**).
xMessage *BIT*	Bit 13 des AIF-OUT-Statusworts: Meldung ist aktiv (d. h. Unter-/Überspannungszustand) FALSE: keine Meldung aktiv TRUE: eine Meldung ist aktiv (d. h. Unter-/Überspannungszustand) **Hinweis:** Dieses Bit muss mit dem entsprechenden Signal in der Anwendung verbunden sein (d. h. über das invertierte Statussignal *xVoltageEnabled* des Funktionsblocks **L_TB2P_AxisInterface**).
xBit14 *BIT*	Bit 14 des AIF-OUT-Statusworts FALSE: Status inaktiv TRUE: Status aktiv **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei belegt werden.
xBit15 *BIT*	Bit 15 des AIF-OUT-Statusworts FALSE: Status inaktiv TRUE: Status aktiv **Hinweis:** Dieses Bit hat keine feste Bedeutung, kann aber vom Benutzer frei verbunden werden.
wStat *WORD*	AIF-OUT-Statuswort Das wStat-Signal ist logisch mit den Bits *xBit00* … *xBit15* OR-verknüpft. Damit bleibt es dem Anwender überlassen, ob der Status einzeln über die Booleschen Eingänge *xBit00* … *xBit15* oder über das wStat-Statuswort zusammengestellt wird.
wOut1 *WORD*	Ausgabe einer 16-Bit-Ganzzahl Typischerweise wird das zweite WORD auf AIF-OUT1 als Drehzahlsollwert des Antriebs interpretiert, skaliert in [%] (0 … 16384 = 0,0 … 100,0[%]). Es bleibt jedoch dem Anwender überlassen, die Bedeutung in der Anwendung zu definieren.
wOut2 *WORD*	Ausgabe eines freien 16-Bit-WORD-Werts
wOut3 *WORD*	Ausgabe eines freien 16-Bit-WORD-Werts

Tipp:

Möchten Sie einen 32-Bit-Wert in zwei 16-Bit-Werte auf *scStatusWords1.wOut2* und *ScStatusWords1.wOut3* aufteilen? Der Funktionsblock **UnpackDword**[7] bietet diese Funktion. Verwenden Sie ihn wie folgt:

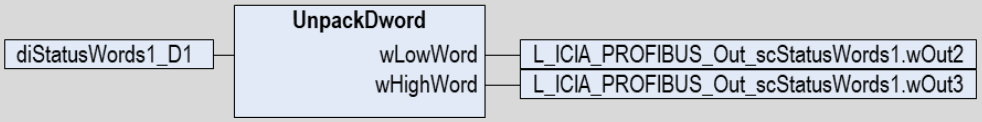


Abbildung25 : Umwandlung eines 32-Bit-*DWORD*-Werts in zwei 16-Bit-*WORD*-Werte

[7] enthält die *CAA*-Speicherbibliothek

Benutzerdefinierte Variablenstruktur *L_ICIA_scStatusWords*

Diese Struktur implementiert die erweiterte AIF-OUT-Schnittstelle, die aus der Wechselrichter-Serie 9300 ServoPLC bekannt ist. Sie wird auf die Objekte scStatusWords2 und scStatusWords3 angewendet und umfasst die folgenden Elemente:

Bezeichner Datentyp	Beschreibung
wOut0 *WORD*	Ausgabe eines freien 16-Bit-WORD-Werts
wOut1 *WORD*	Ausgabe eines freien 16-Bit-WORT-Werts
wOut2 *WORD*	Ausgabe eines freien 16-Bit-WORT-Werts
wOut3 *WORD*	Ausgabe eines freien 16-Bit-WORD-Werts

3 Anwendungsbeispiel

3.1 Inbetriebnahme-Sequenz (Motion-Anwendung)

In der Regel wird PROFIBUS in neuen Maschinen nicht verwendet, da es modernere Feldbussysteme wie EtherCAT oder PROFInet gibt. Der PROFIBUS-Feldbus kommt darüber hinaus in bestehenden Maschinen zum Einsatz. Dieses Dokument konzentriert sich auf ältere Lenze-Servoumrichter[8] , die nun durch die neueste Gerätegeneration der i950-Antriebe ersetzt werden müssen. Im besten Fall benötigt das Ersatzgerät i950 einen funktionalen Zwilling des bisherigen Servoumrichters. Anstelle der bekannten Funktionsblockverbindung des GDC basiert das SPS-Programm des i950 auf den Technologiemodulen von Lenze mit einigen geringfügigen Erweiterungen, um eine 100-prozentige Funktionskompatibilität zwischen dem bisherigen und dem aktuellen Antriebssystem zu erreichen.

Das folgende Beispiel zeigt, wie ein 9300-Servoumrichter im Drehzahlregelungs[9] us auf einen kompatiblen i950-Signalfluss migriert werden kann, wobei das Lenze-Technologiemodul **L_TF2P_SpeedControlBase** verwendet wird.

start

Voraussetzungen:

- »PLC Designer« ist bereits auf Ihrem PC geöffnet[10] .
- In »PLC Designer« ist kein Projekt geöffnet.

Schritt 1

Erstellen Sie ein neues Projekt in »PLC Designer«:

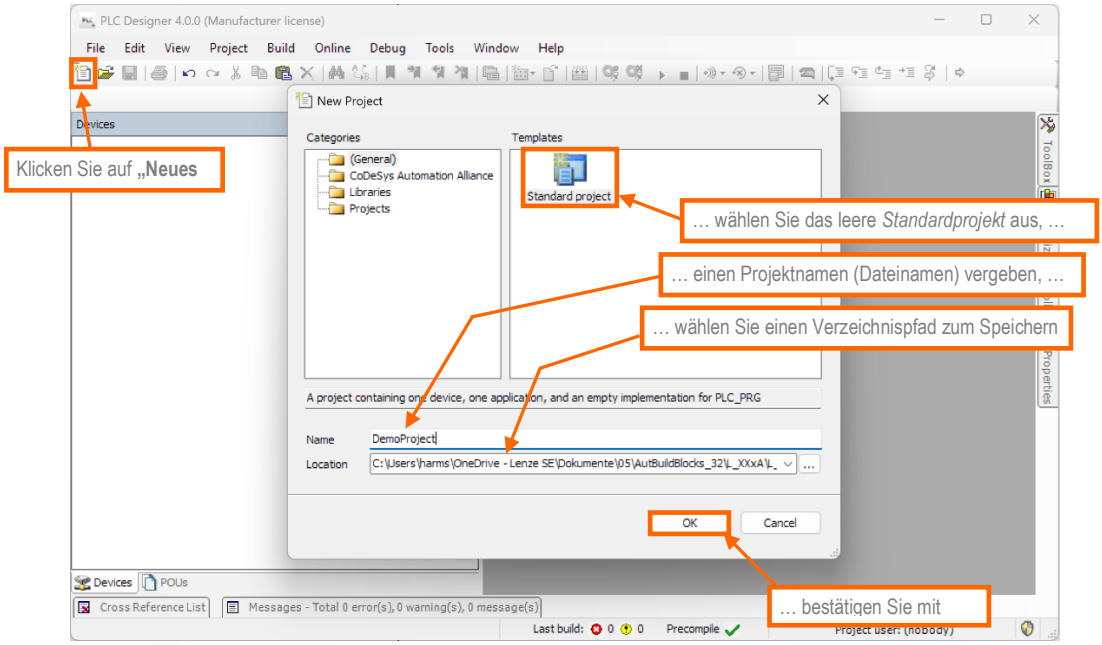


Abbildung26 : Erstellung eines neuen Projekts in „PLC Designer“

[8] insbesondere die Servoumrichter-Serie 9300
[9] Grundkonfiguration „Drehzahlregelung über AIF” (C0005/000 = 1003)
[10] In diesem Beispiel verwenden wir „PLC Designer“ V4.x.

Schritt 2

Geben Sie das Zielsystem i950 an:

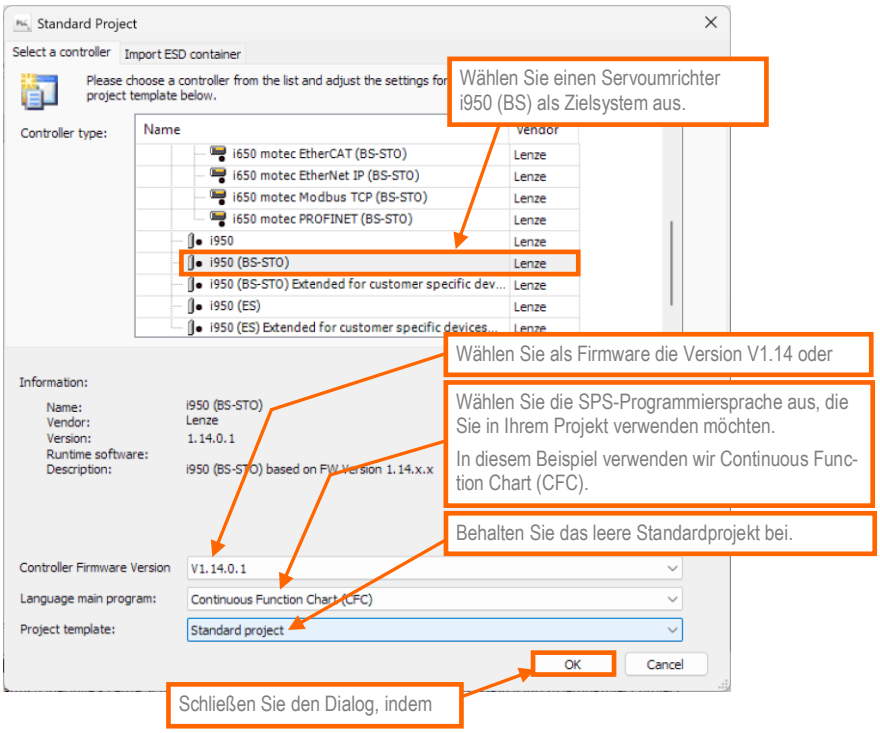


Abbildung27 : Wählen Sie einen i950 (BS) mit Firmware-Version V1.14 oder höher als Zielsystem aus.

Schritt 3

Führen Sie einen **Build**-Prozess aus, um den Zugriff auf die Inbetriebnahmedialoge zu ermöglichen:

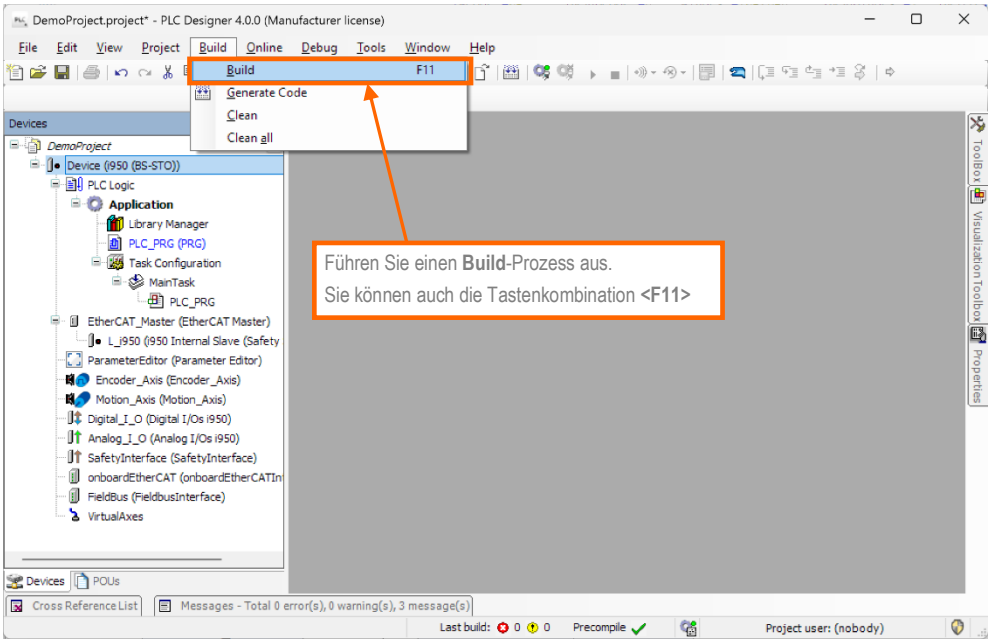


Abbildung28 : Erstellen Sie das Projekt, um Zugriff auf die Inbetriebnahmedialoge von »PLC Designer« zu erhalten.

Schritt 4

Legen Sie die wichtigen Daten in den Inbetriebnahmedialogen des Geräts fest:

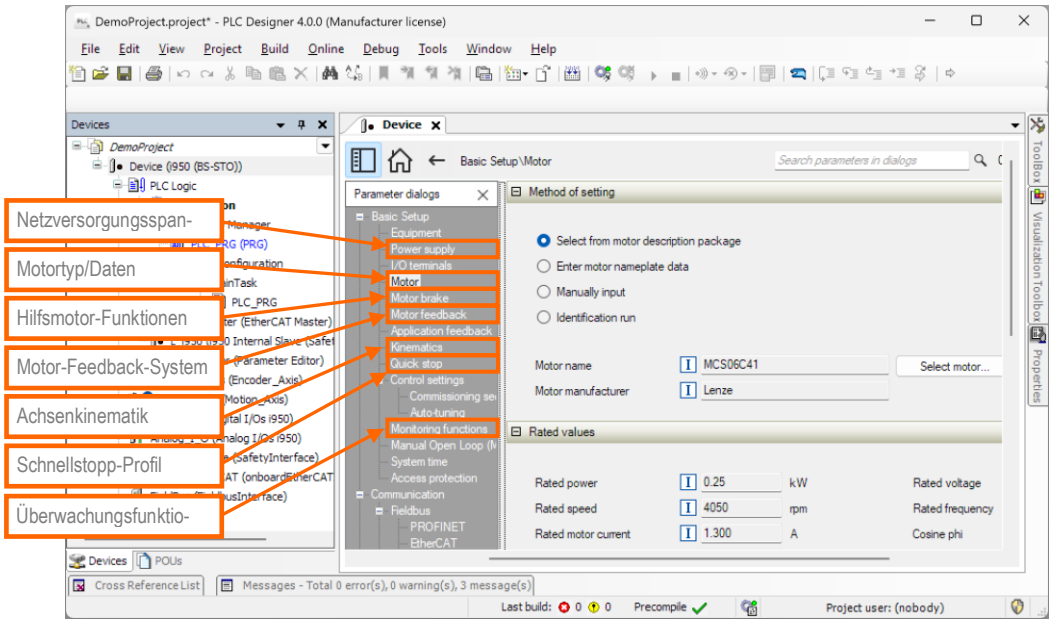


Abbildung29 : Grundeinstellungen des i950-Antriebs

- Netzspannung
- Motordaten
- Motorbremse (falls montiert/verdrahtet)
- Motor-Feedbacksystem
- Achsenkinematik (Getriebeübersetzung, Vorschubkonstante, ...)
- Schnellstopp-Profilparameter
- Überwachungsfunktionen (Folgefehler, Endschalter, ...)

Tipps:

- Verwenden Sie den Motorkatalog von »PLC Designer«, um die Motordaten schnell zu finden/einzustellen.
- Die Auto-Tuning-Funktion des i950 ermöglicht es, optimale Reglereinstellungen für das dynamische Verhalten des Servoantriebs zu finden.

Schritt 5

Öffnen Sie den **Bibliotheks-Manager,** um die Bibliothek *L_TF2P_TechModulesFollowingPositioning* zu Ihrem Projekt hinzuzufügen:

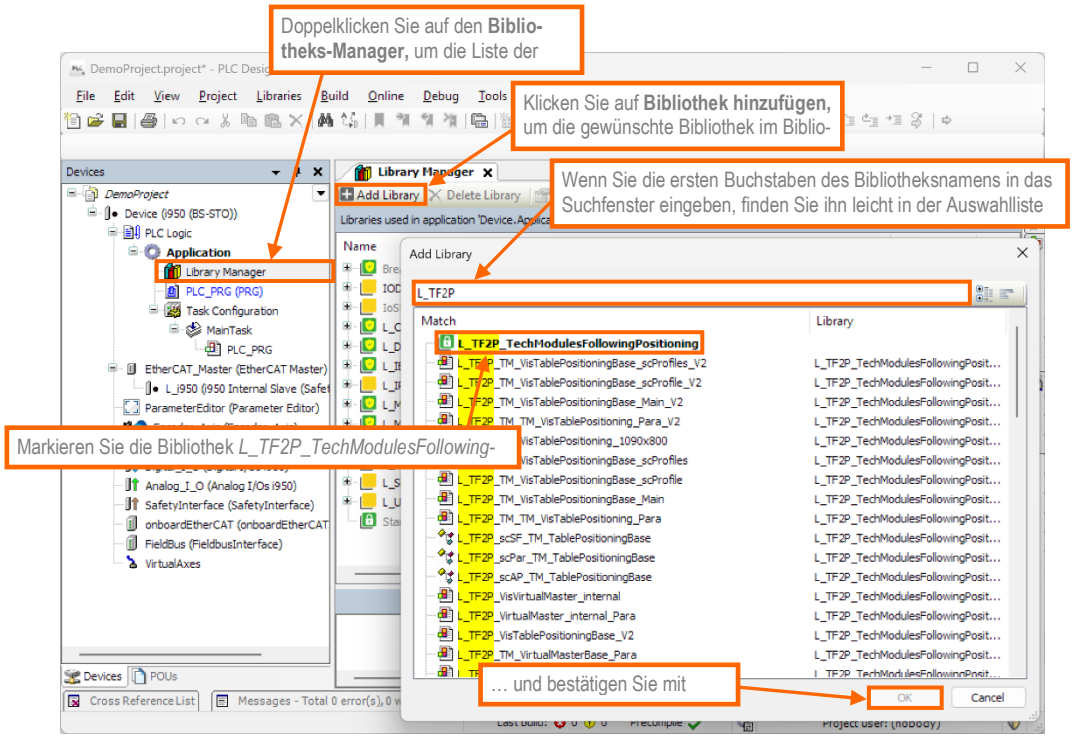


Abbildung30 : Hinzufügen der Bibliothek „*L_TF2P_TechModulesFollowingPositioning*“ zu Ihrem Projekt

Schritt 6

Fügen Sie auf die gleiche Weise wie in Schritt 5 beschrieben auch die Bibliothek *„L_TB2P_TechModulesBasic“* in den **Bibliotheksmanager** Ihres Projekts ein.

Schritt 7

Öffnen Sie das Programm **PLC_PRG** und schreiben Sie ein kleines CFC-Programm wie folgt:

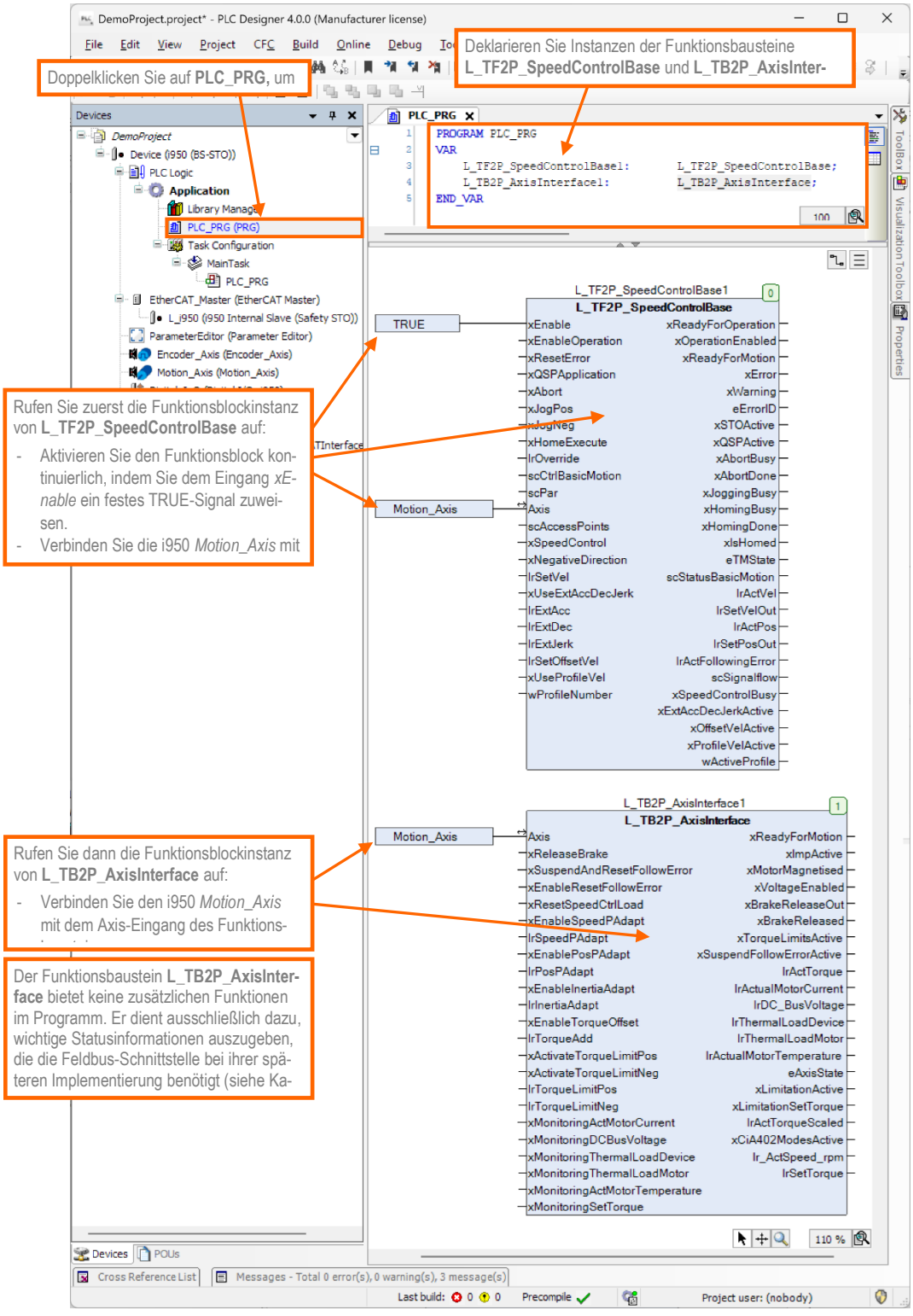


Abbildung31 : Aufruf der Funktionsbausteine **L_TF2P_SpeedControlBase1** und **L_TB2P_AxisInterface1** in **PLC_PRG**

Schritt 8

Fügen Sie ein leeres Visualisierungsfeld ein:

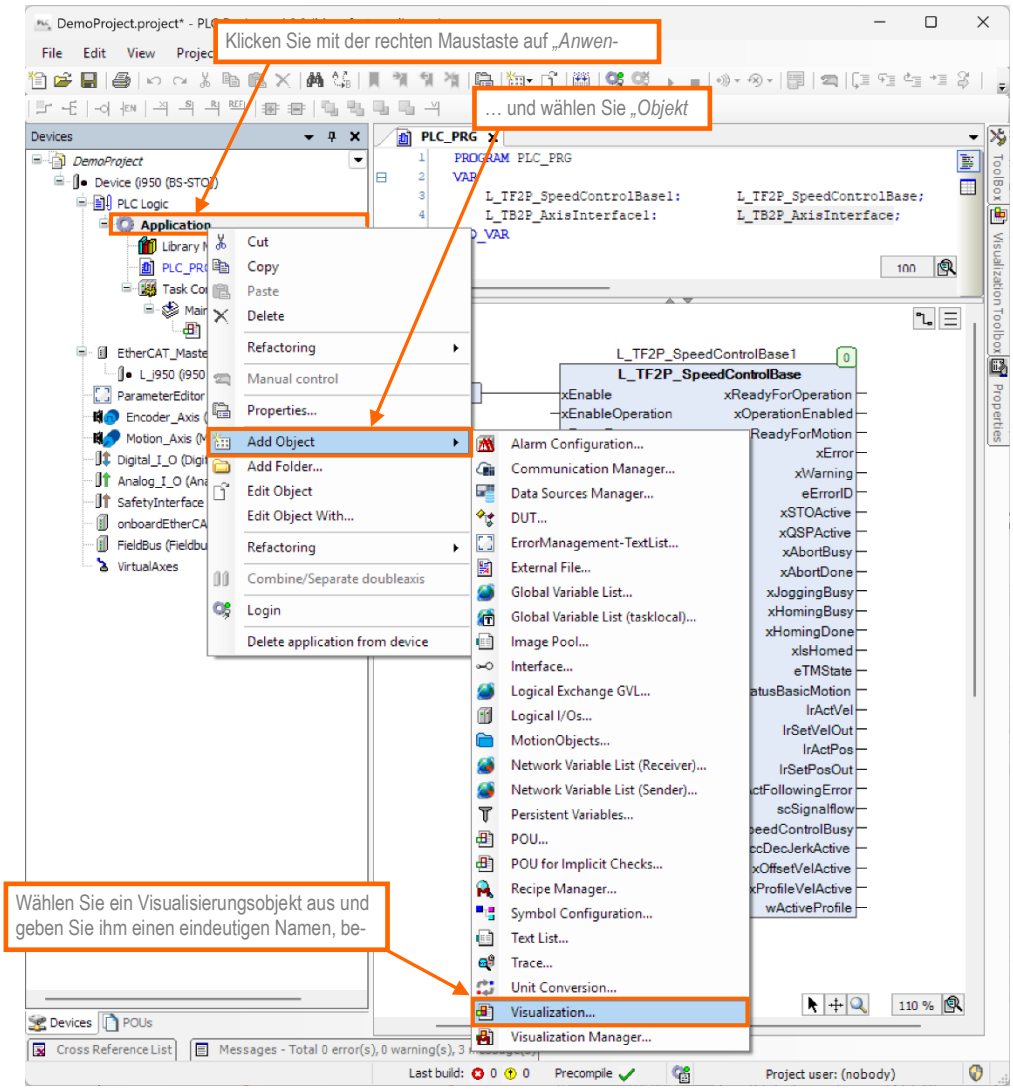


Abbildung32 : Hinzufügen eines Visualisierungsbildschirms zur Bedienung des Funktionsblocks **L_TF2P_SpeedControlBase**

Schritt 9

Fügen Sie für einen ersten Test die Visualisierungsvorlage des Technologiemoduls **L_TF2P_SpeedControlBase** ein, um es über den Visualisierungsbildschirm zu bedienen:

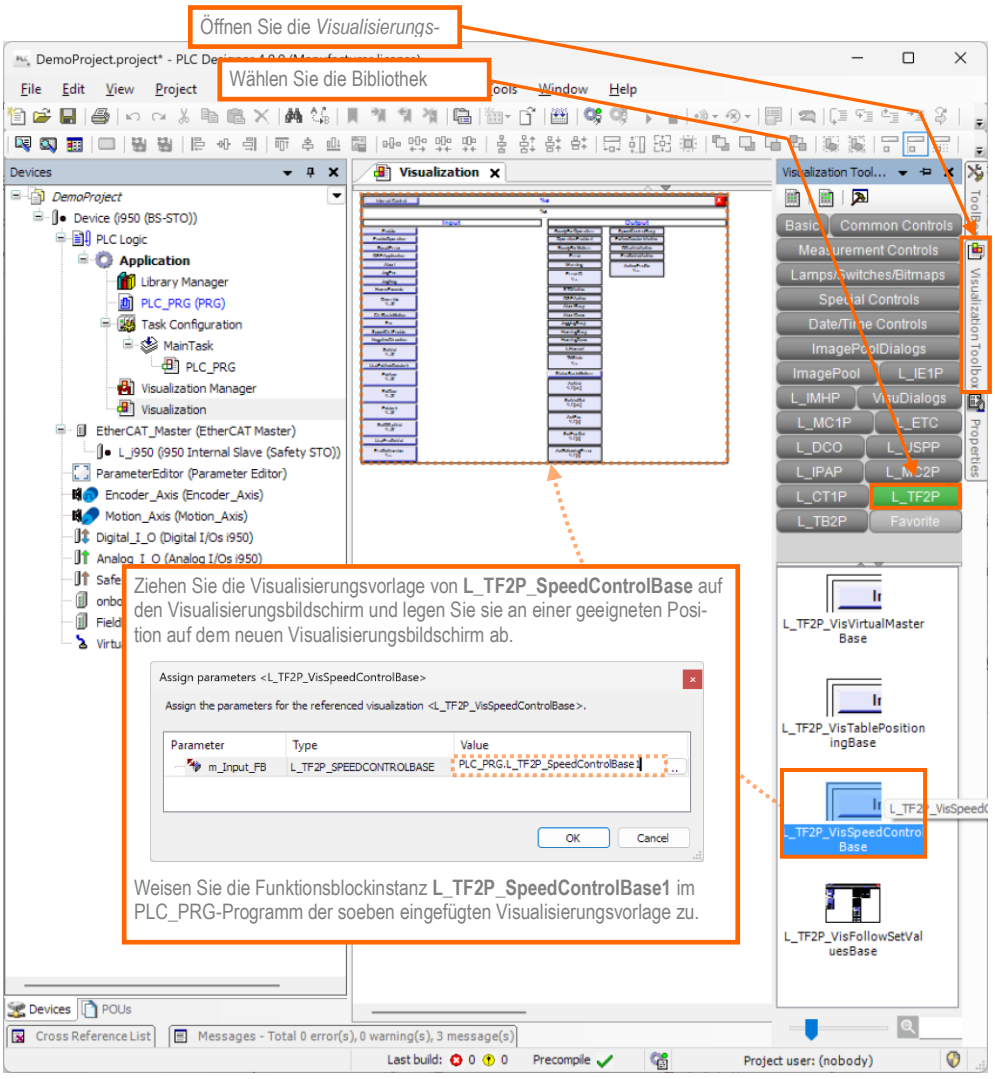


Abbildung33 : Hinzufügen der Visualisierungsvorlage von **L_TF2P_SpeedControlBase**

Schritt 10

Testen Sie Ihr SPS-Programm:

- Schalten Sie die Netzspannung und die 24-V-Steuerspannung Ihres i950-Antriebs ein.
- Laden Sie das Projekt auf Ihren i950-Antriebsregler herunter und starten Sie das SPS-Programm.
- Geben Sie den STO-Befehl auf dem i950-Antrieb frei.

Schritt 11

Bedienen Sie den i950-Antrieb über den Visualisierungsbildschirm in verschiedenen Betriebsmodi und überprüfen Sie alle Funktionen:

Aktivieren Sie die interne Steuerung des Funktionsblocks über den Visualisierungsbildschirm.

Schalten Sie den i950-Antrieb ein, indem Sie auf *EnableOperation*

Nachdem Sie den Antriebsbetrieb aktiviert haben, können Sie einen ersten Test durchführen, indem Sie den Antrieb manuell über die

Aktivieren Sie den Drehzahlregelungsmodus, indem Sie

Legen Sie im Eingabefeld *SetVel* eine Zielgeschwindigkeit fest. Der Antrieb sollte sich nun in positiver Richtung drehen.

Wenn Sie möchten, können Sie auch die Parameter für Beschleunigung, Verzögerung und Ruck

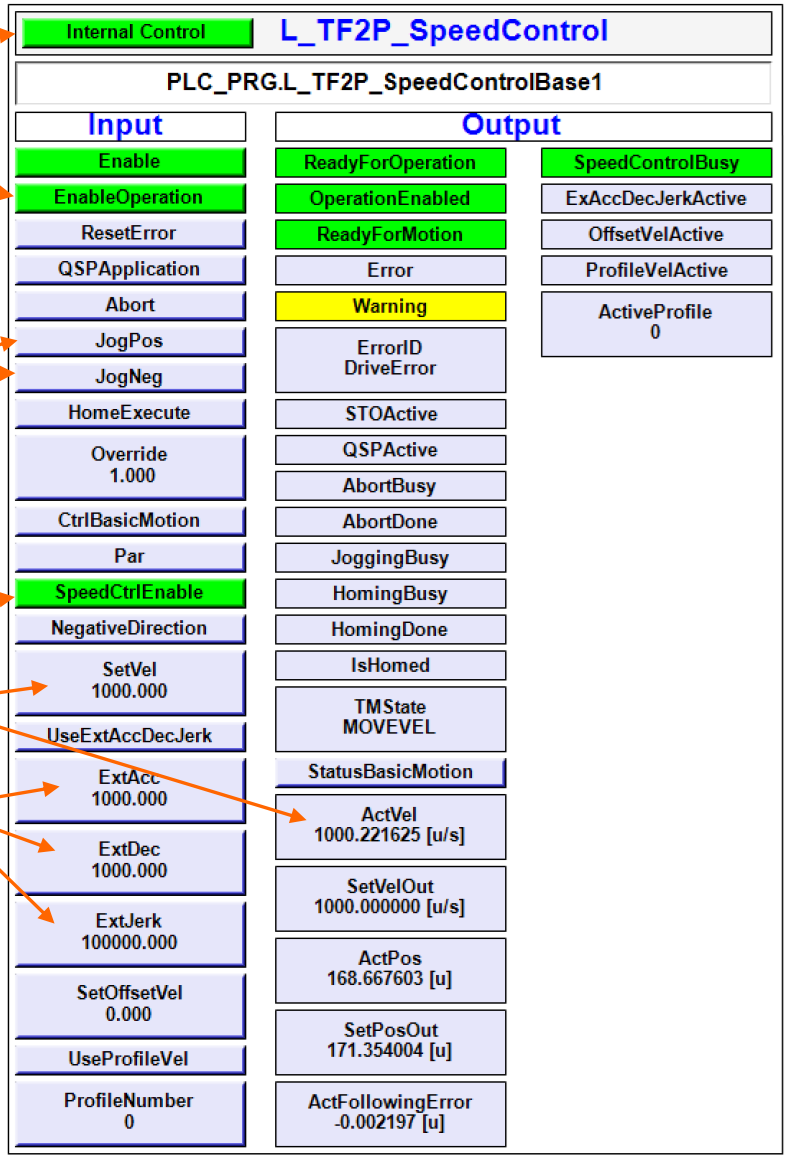


Abbildung34 : Visualisierungsbildschirm von **L_TF2P_SpeedControlBase**

Ende

3.2 Inbetriebnahmesequenz (PROFIBUS)

Im folgenden Kapitel wird beschrieben, wie Sie die PROFIBUS-Kommunikation mit Hilfe der Funktionsbausteine **L_ICIA_CommunicationInterface** in Betrieb nehmen.

Start

Voraussetzungen:

- Das Feldbussystem ist gemäß den PROFIBUS-Spezifikationen verdrahtet.
- Die Logik-SPS (PROFIBUS-Master) sowie alle PROFIBUS-Slave-Geräte werden mit Steuerspannung (24 V_{DC}) versorgt.
- Das SPS-Programm des i950 ist im »PLC Designer« geöffnet, aber noch nicht online.
- Der Anwendungssignalfluss wurde wie im vorherigen Kapitel „3.1 “ beschrieben im SPS-Programm des i950 implementiert, beispielsweise bei der Migration von Bewegungsanwendungen von Mitbewerbern oder älteren Lenze-Geräten.

Schritt 1

Öffnen Sie das **Bibliotheks-Repository** und installieren Sie die Bibliothek *L_ICIA_CommunicationInterface*:

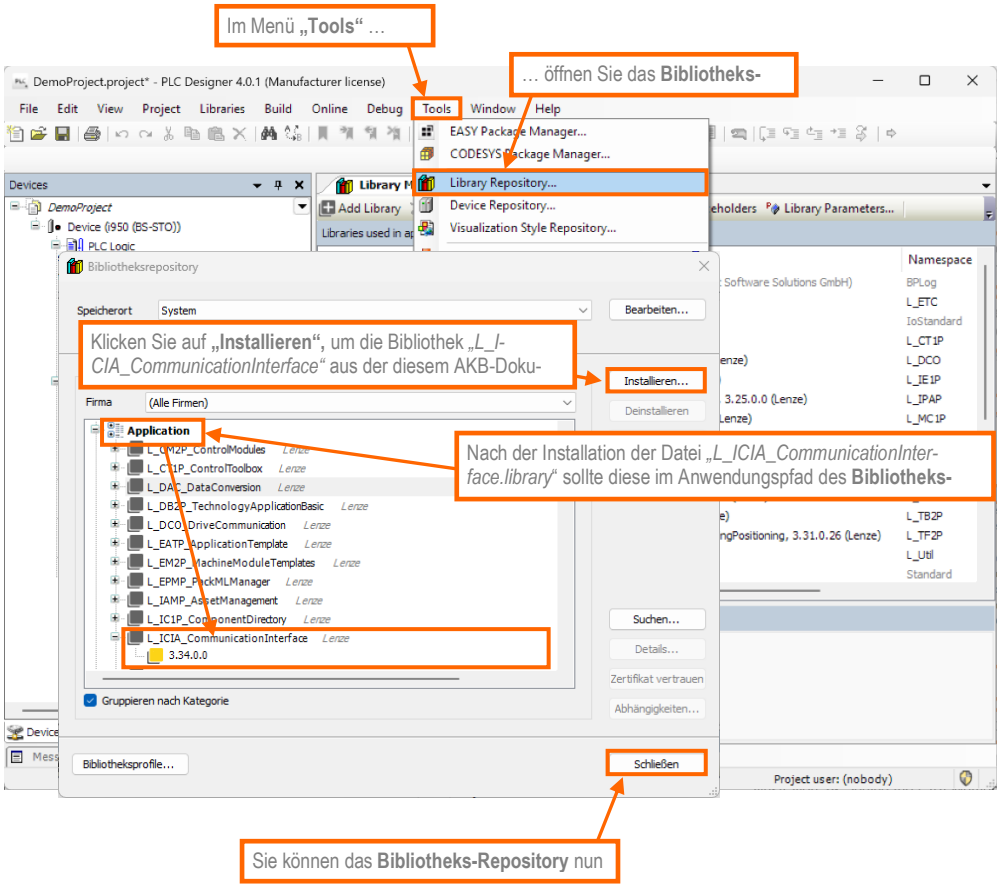


Abbildung35 : Hinzufügen der Bibliothek „*L_ICIA_CommunicationInterface*“ zum **Bibliotheks-Repository**

Schritt 2

Öffnen Sie den **Bibliotheksmanager,** um die Bibliothek *„L_ICIA_CommunicationInterface“* zu Ihrem Projekt hinzuzufügen:

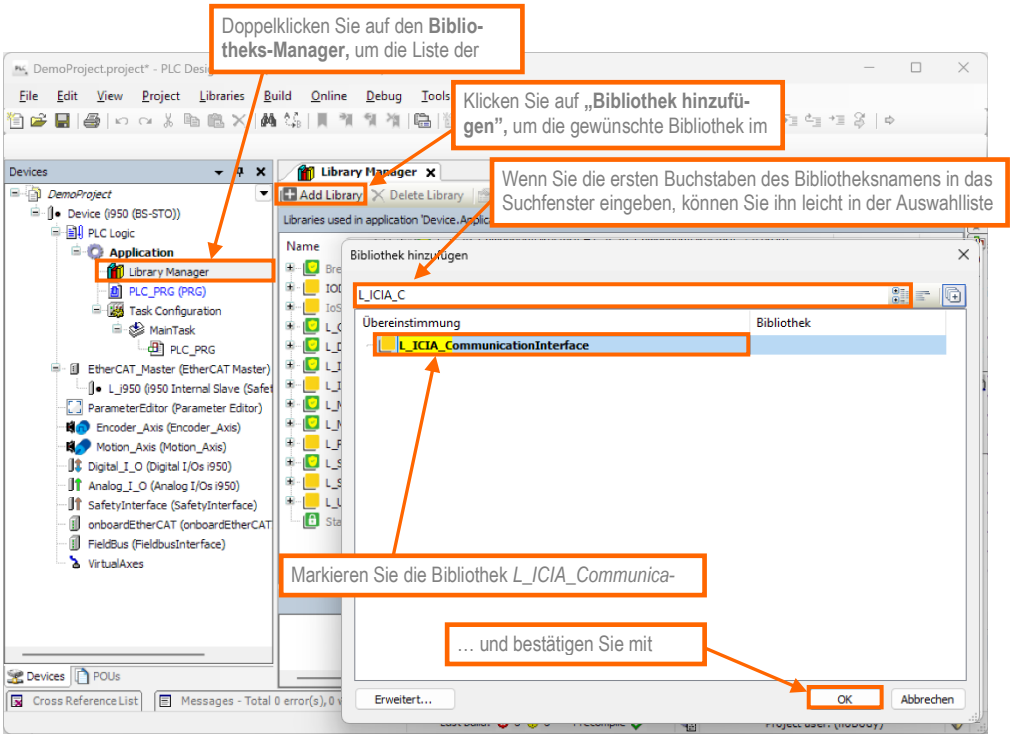


Abbildung36 : Hinzufügen der Bibliothek *„L_ICIA_CommunicationInterface“* zu Ihrem Projekt

Schritt 3

Fügen Sie wie in Schritt 2 gezeigt auch die folgenden Bibliotheken in den **Bibliotheksmanager** Ihres Projekts ein:

- *CAA-Speicher* (V03.05)
- *L_SI9P_IoDrvi900* (V03.33 – Datei im Anhang zum AKB-Dokument 202500431)

Hinweis

Lösen Sie im **Library Manager** den Platzhalter für die Bibliothek *L_SI9P_IoDrvi900* auf und ändern Sie ihn von einer geräteabhängigen Version in eine feste Version V03.33 oder höher.

Tipp

Um sicherzustellen, dass Sie die richtigen Bibliotheken in Ihrem Projekt haben, können Sie das Projektarchiv *TM_SpeedControl.projectarchive* öffnen, das dem AKB-Artikel 202500431 beigefügt ist.

Erstellen Sie Ihr Projekt auf der Grundlage dieses Projektarchivs.

Schritt 4

Deklarieren Sie die folgenden globalen Schnittstellenvariablen-Arrays für die Feldbuskommunikation in einem separaten GVL-Element *TA_IO*:

```
{attribute 'qualified_only'}
VAR_GLOBAL
    adwPROFIBUS_IN:     ARRAY [0..15] OF DWORD;          // Rohdateneingabe von PROFIBUS
    adwPROFIBUS_OUT:    ARRAY [0..15] OF DWORD;          // Rohdatenausgabe an PROFIBUS
END_VAR
```

Schritt 5

Ordnen Sie die in Schritt 4 deklarierten Variablen-Arrays wie folgt der Feldbus-Schnittstelle zu:

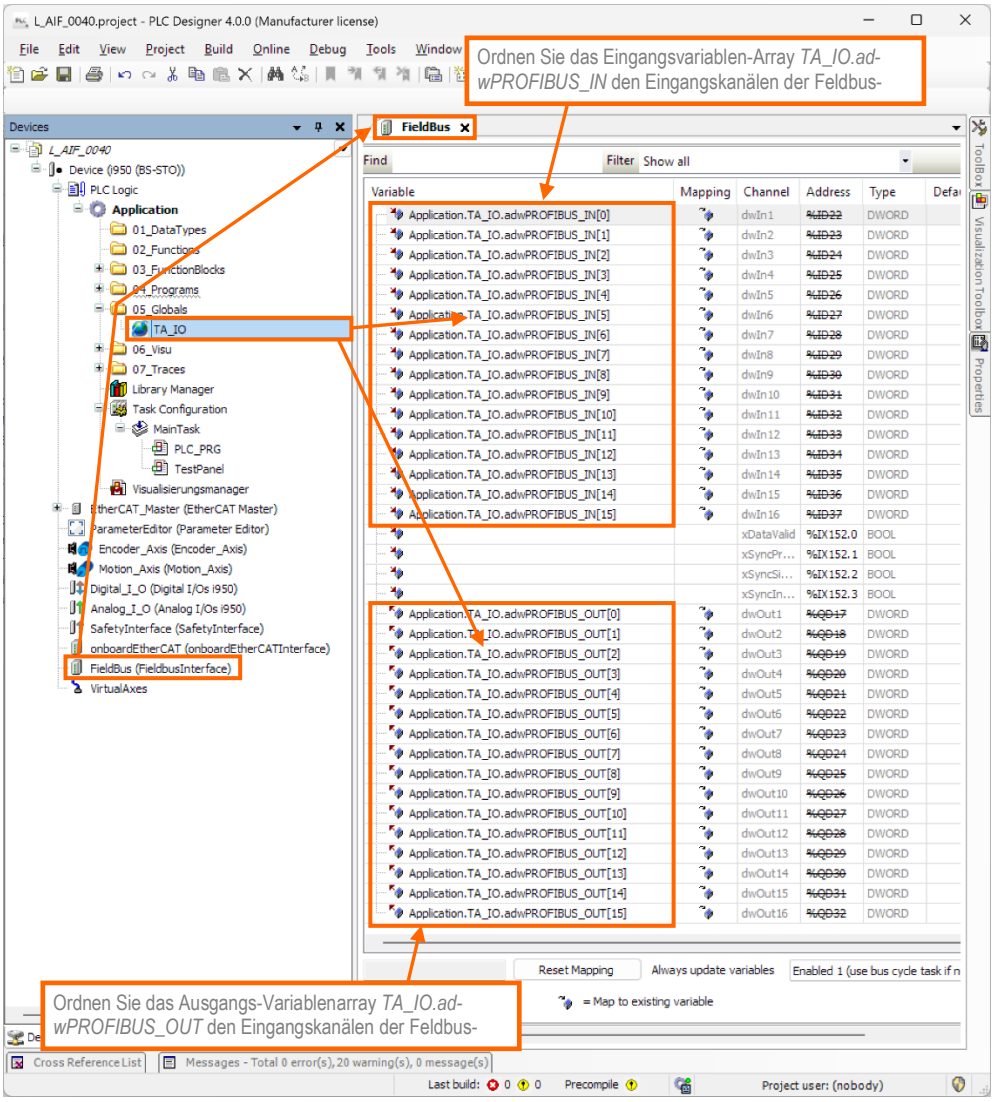


Abbildung37 : Zuordnung globaler Variablen-Arrays zur Feldbus-Schnittstelle des i950

Schritt 6

Deklarieren Sie im Bewegungsprogramm *PLC_PRG* des i950 die folgenden Funktionsbausteine:

```
VAR
   L_ICIA_PROFIBUS_Base1:L_ICIA_PROFIBUS_Base;   // Handhabung der grundlegenden PROFIBUS-Kommunikation
   L_ICIA_PROFIBUS_In1:  L_ICIA_PROFIBUS_In;     // Lesen/Verarbeiten von Feldbus-Eingängen an
                                                 // scControlWords
   L_ICIA_PROFIBUS_Out1: L_ICIA_PROFIBUS_Out;    // Verarbeitung/Schreiben von Feldbus-Ausgängen aus
                                                 // scStatusWords
   L_STAT1:              L_STAT;                 // Erzeugung des 4-Bit-Musters des Antriebsstatus
   L_MC1A_ZeroDetect1:   L_MC1A_ZeroDetect;      // Erkennung der Drehrichtung/Nullgeschwindigkeit
END_VAR
```

Schritt 7

Bereiten Sie die Parameterzuordnungsliste „[11] ” vor, indem Sie die Deklarationsliste im Bewegungsprogramm *„PLC_PRG”* des i950 erweitern:

```
…
ascParReference:     ARRAY [0..15] OF L_ICIA_sc93ParReference;      // Parameterreferenzliste
…
```

?

Enthält Ihre GSD-Konfiguration einen Drivecom V0-Parameterkanal?

NO ⇒ 10

JA

Schritt 8

Erweitern Sie die in Schritt 5 begonnene Parameterzuordnungsliste um die erforderlichen Initialisierungswerte, wie in Kapitel „2.1.2 ” beschrieben:

```
…
ascParReference: ARRAY [0..15] OF L_ICIA_sc93ParReference; // Parameterreferenzliste
  (wCode:=51, bySubCode:=0, wIndex:=16#606C, bySubIndex:=0, bySize:=4, diNum:=1171875, diDen:=524288),
  (wCode:=53, bySubCode:=0, wIndex:=16#6079, bySubIndex:=0, bySize:=4, diNum:=10,      diDen:=1),
  (wCode:=63, bySubCode:=0, wIndex:=16#2D49, bySubIndex:=5, bySize:=2, diNum:=1000,    diDen:=1),
  (wCode:=52, bySubCode:=0, wIndex:=16#2D82, bySubIndex:=0, bySize:=4, diNum:=10,       diDen:=1),
  (wCode:=54, bySubCode:=0, wIndex:=16#2DD1, bySubIndex:=5, bySize:=4, diNum:=10,       diDen:=1),
  (wCode:=61, bySubCode:=0, wIndex:=16#2D84, bySubIndex:=1, bySize:=2, diNum:=10,       diDen:=1),
  (wCode:=64, bySubCode:=0, wIndex:=16#2D40, bySubIndex:=7, bySize:=2, diNum:=1,        diDen:=1),
   …
  (wCode:=84, bySubCode:=0, wIndex:=16#2C01, bySubIndex:=2, bySize:=4, diNum:=100,      diDen:=1),
  (wCode:=85, bySubCode:=0, wIndex:=16#2C01, bySubIndex:=3, bySize:=4, diNum:=10,      diDen:=1)];
…
```

Schritt 9

Rufen Sie zunächst den Funktionsblock **L_ICIA_PROFIBUS_Base** in Ihrem SPS-Programm auf und verbinden Sie die Variablen wie gezeigt:

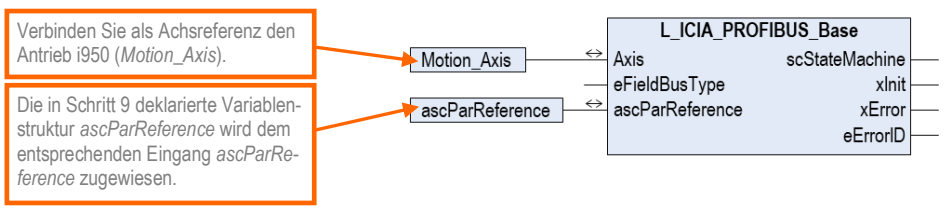


Abbildung38 : Aufruf des Funktionsbausteins **L_ICIA_PROFIBUS_Base** am Anfang des SPS-Programms

[11] Wenn kein Parameterkanal verwendet wird, ist die Deklaration dennoch als Dummy-Zuweisung erforderlich.

Schritt 10

Rufen Sie den Funktionsbaustein **L_ICIA_PROFIBUS_In** direkt nach dem Funktionsbaustein **L_ICIA_PROFIBUS_Base** in Ihrem SPS-Programm auf und verbinden Sie die Variablen wie gezeigt:

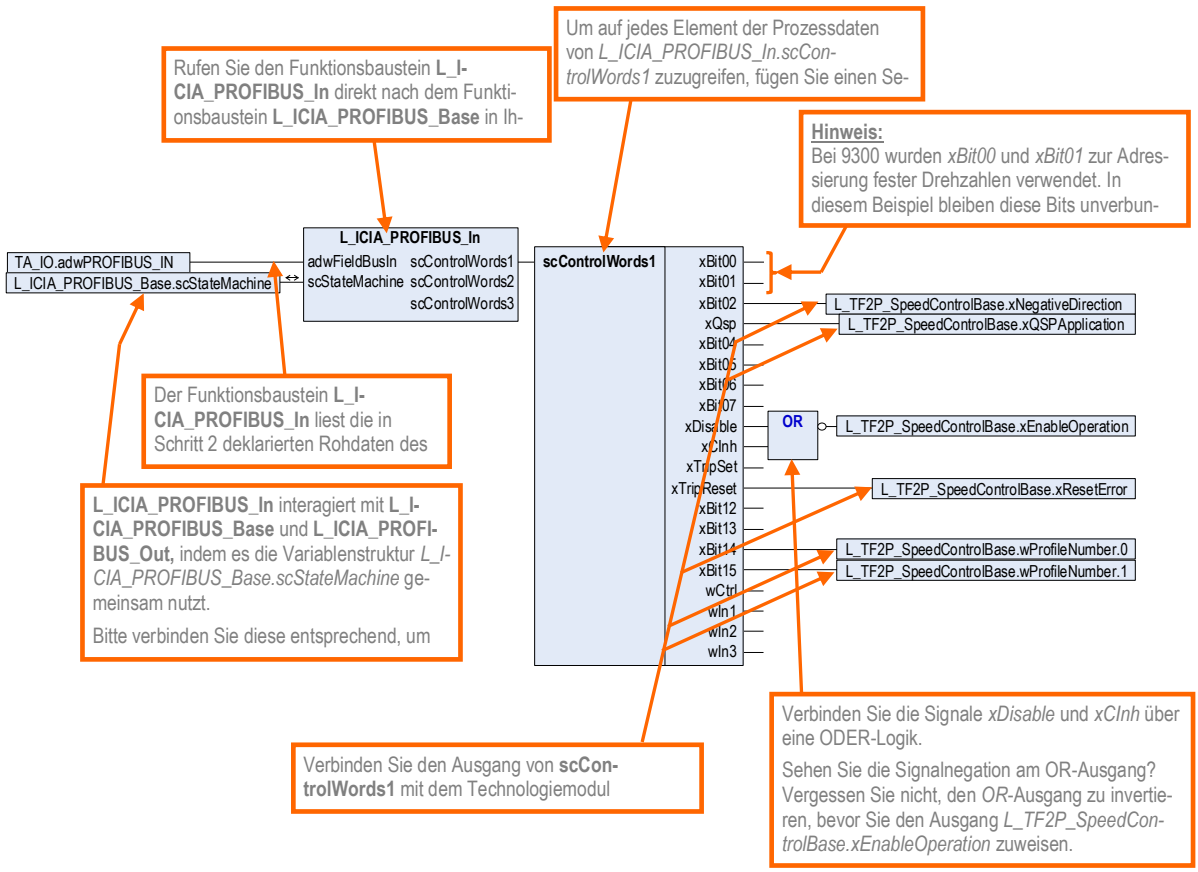


Abbildung39 : Aufruf des Funktionsblocks **L_ICIA_PROFIBUS_In** direkt hinter dem Funktionsblock **L_ICIA_PROFIBUS_Base**

Schritt 11

Lesen Sie den maximalen Drehzahlwert n_{max}[12] aus den Legacy-Antrieben[13] und wandeln Sie ihn mit Hilfe der kinematischen Parameter der Antriebsachse in eine Referenzgeschwindigkeit V_{max} um.

Beispiel: n_{max} = 3000[U/min]

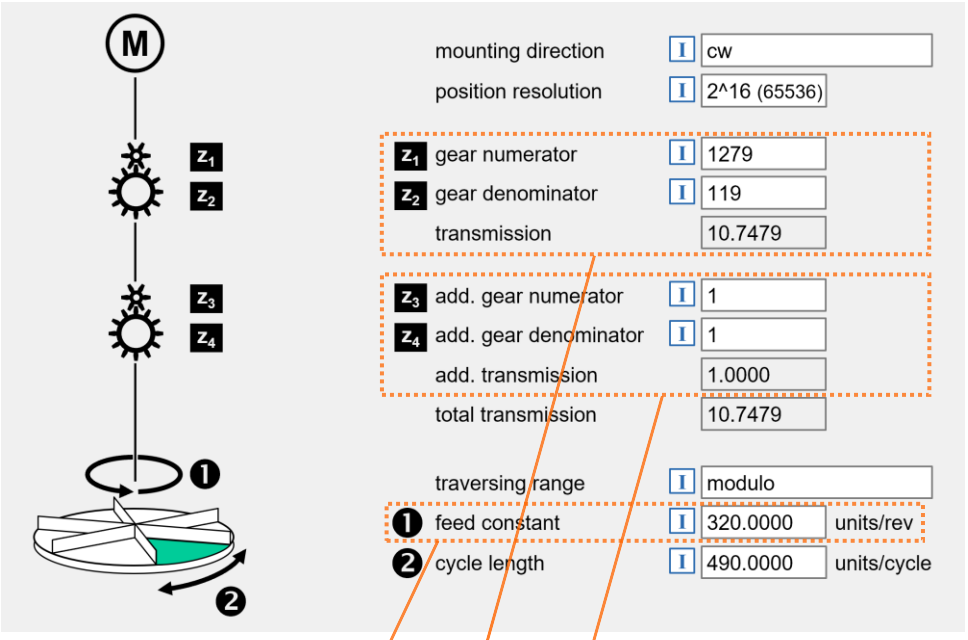


Abbildung40 : Kinematische Parameter des i950-Antriebs

$$V_{max} = \frac{n_{max}}{60\frac{s}{min}} \cdot FeedConstant \cdot \frac{1}{i_{gear}} \cdot \frac{1}{i_{gear,add}} =$$

$$= \frac{n_{max}}{60\frac{s}{min}} \cdot 0x500A{:}032 \cdot \frac{0x500A{:}034}{0x500A{:}033} \cdot \frac{0x500A{:}026}{0x500A{:}025} =$$

$$= \frac{3000\frac{rev.}{min}}{60\frac{s}{min}} \cdot 320.0000\frac{units}{rev.} \cdot \frac{119}{1279} \cdot \frac{1}{1} = 1488.663 \ldots \frac{units}{s}$$

Schritt 12

Deklarieren Sie eine konstante Variable *C_lrMaxVelocity* mit einem Initialisierungswert von $V_{(max)}$, wie im vorherigen Schritt berechnet:

```
VAR CONSTANT
   C_lrMaxVelocity:  LREAL := 1488,663;          // maximale Antriebsgeschwindigkeit, skaliert in [Ein-
heiten/s]
END_VAR
```

[12] skaliert in [U/min]
[13] Bei den älteren Lenze-Geräten wurde die maximale Drehzahl im Code C0011/000 eingestellt.

Schritt 13

Lesen und skalieren Sie den Drehzahlsollwert auf *L_ICIA_PROFIBUS_In.scControlWords1.wIn1* und leiten Sie ihn über den folgenden Signalfluss an das Drehzahlregelungsmodul **L_TF2P_SpeedControlBase** weiter. Ändern Sie Ihr Programm wie folgt:

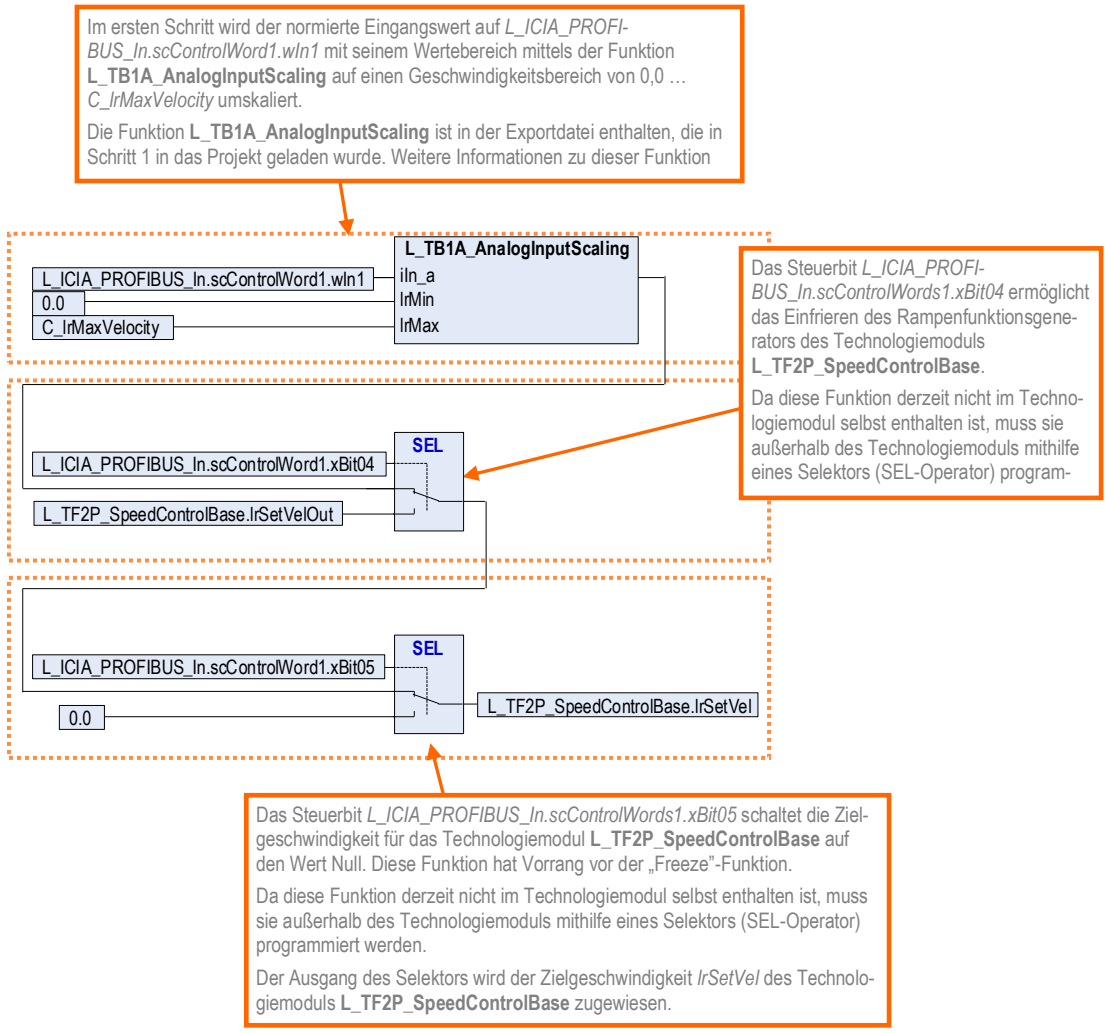


Abbildung41 : Lesen/Skalieren/Ändern des Sollgeschwindigkeitswerts *lrSetVel* von **L_TF2P_SpeedControlBase**

Schritt 14

Um das Statuswort im Funktionsbaustein **L_ICIA_PROFIBUS_Out** vorzubereiten, sind drei Vorbereitungsschritte erforderlich:

Im ersten Schritt deklarieren Sie eine Variable *lrZeroSpeedThreshold* für ein Geschwindigkeitstoleranzfenster. Das Toleranzfenster definiert das Verhalten/die Stabilität des Nullgeschwindigkeitssignals.

```
…
lrZeroSpeedThreshold:      LREAL := 1.0;     // Nullgeschwindigkeitstoleranzschwelle, skaliert in [Ein-
heiten/s]
…
```

Initialisieren Sie die Variable mit einem Schwellenwert für die Geschwindigkeit. Immer wenn die tatsächliche Geschwindigkeit des Antriebs kleiner/gleich dem Schwellenwert ist, wird das Statusflag (n=0) auf den Wert TRUE gesetzt.

Schritt 15

Rufen Sie im nächsten Schritt den Funktionsblock **L_MC1A_ZeroDetect** am Ende Ihres Programms auf und verbinden Sie ihn wie folgt im Signalfluss:

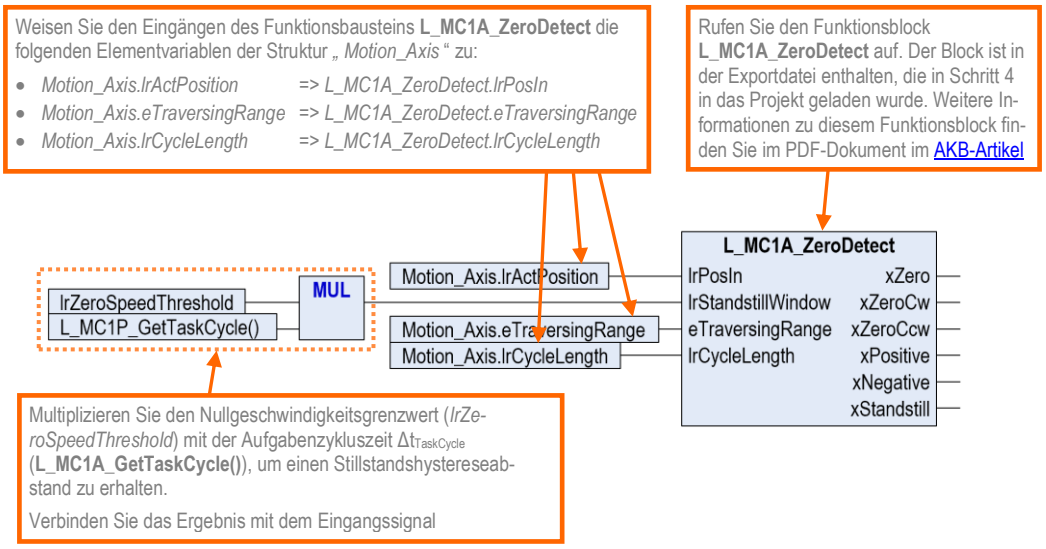


Abbildung42 : Nullgeschwindigkeitserkennung mit dem Funktionsbaustein **L_MC1A_ZeroDetect**

Schritt 16

Rufen Sie am Ende Ihres Programms den Funktionsbaustein **L_STAT** auf und verbinden Sie die folgenden Variablen:

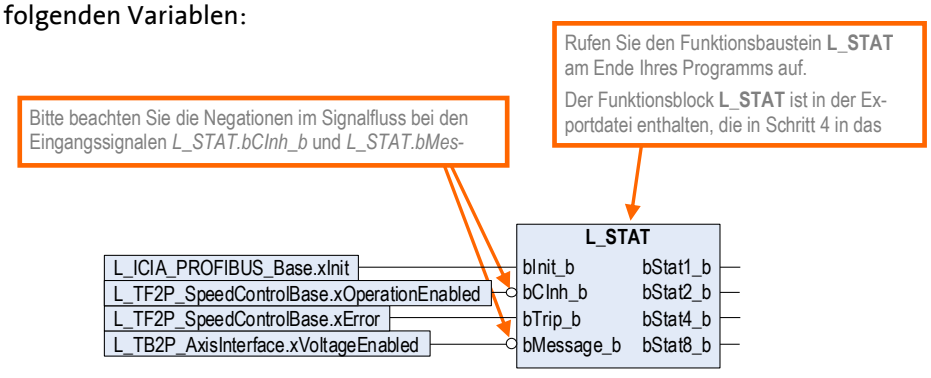


Abbildung43 : Generierung der Statusbits *bStat1_b … bStat8_b* mittels des Funktionsblocks **L_STAT**

Schritt 17

Fügen Sie einen Composer-Block von *scStatusWords1* ein und verbinden Sie die folgenden Signale:

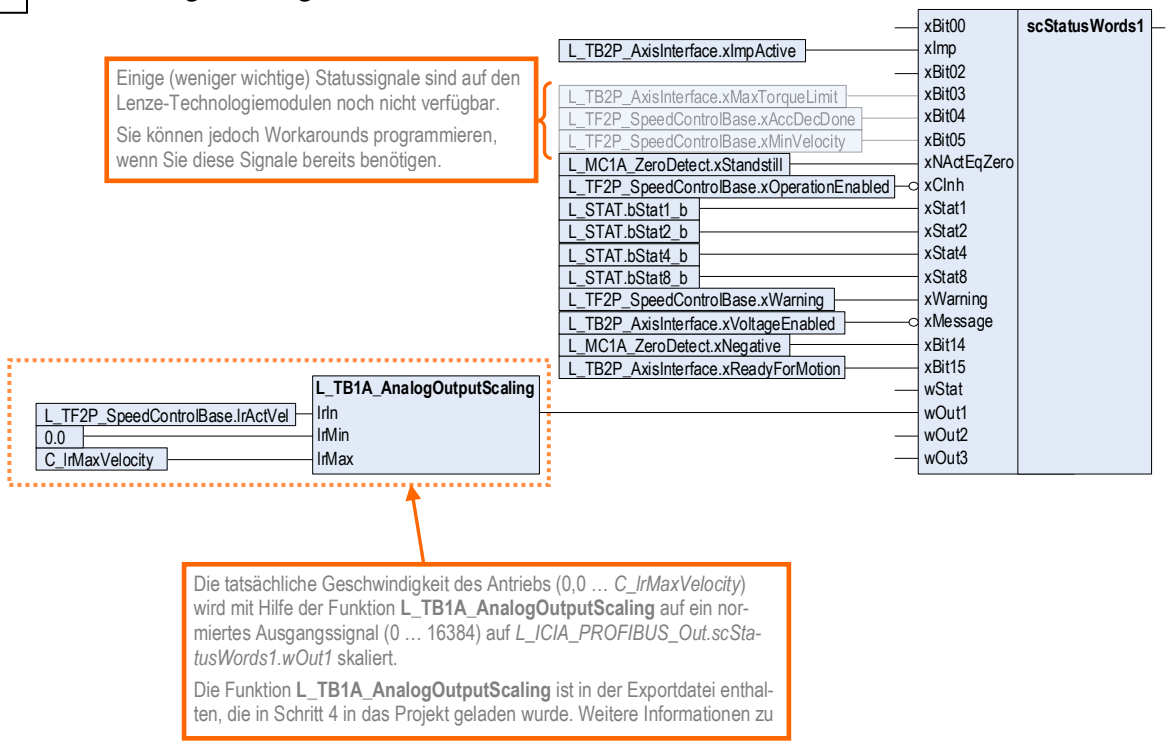


Abbildung44 : Zuordnung von Statussignalen/normalisierter Motordrehzahl zum Composer *scStatusWords1*

Schritt 18

Fügen Sie den Funktionsbaustein **L_ICIA_PROFIBUS_Out** am Ende Ihres Programms ein, verbinden Sie den Ausgang des Composers *L_ICIA_scStatusWords1* mit dem Eingang *scStatusWords1* des Bausteins **L_ICIA_PROFIBUS_Out** und weisen Sie die Generierung des AIF-Statusworts über den Funktionsbaustein **L_ICIA_PROFIBUS_Out** zu:

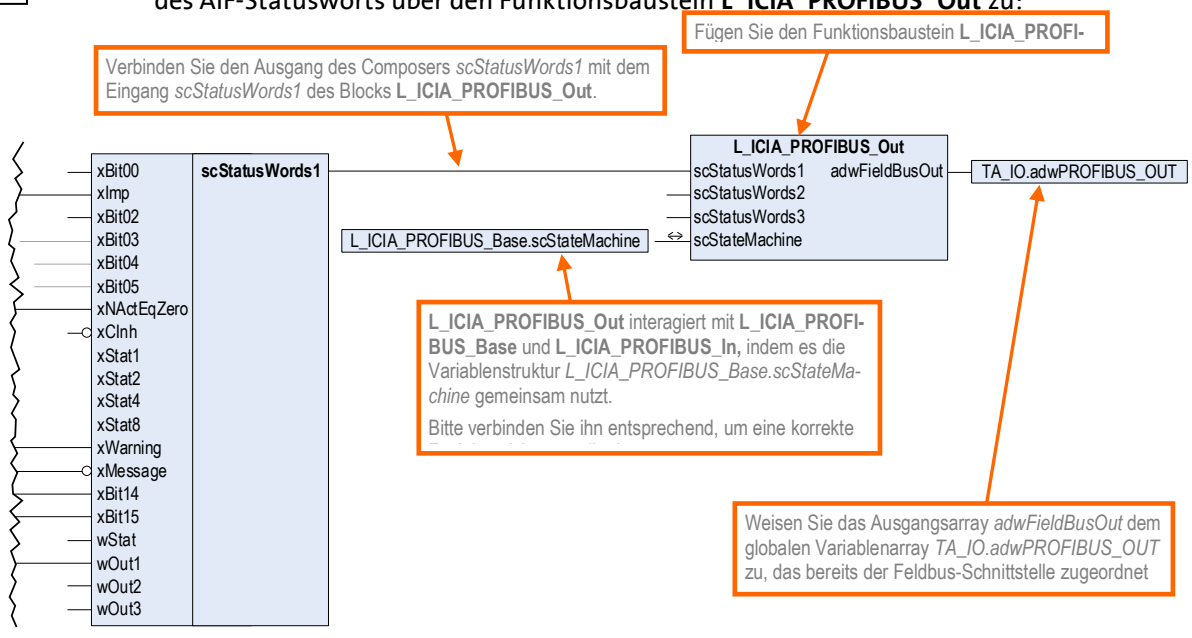
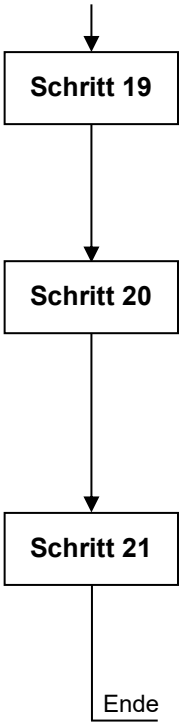


Abbildung45 : Aufruf des Funktionsbausteins **L_ICIA_PROFIBUS_Out** am Programmende und Zuordnung zu den Feldbus-Ausgangsvariablen



Stellen Sie die PROFIBUS-Stationsadresse des i950 im Index 0x2341:001 ein:

Index	Subindex	Name	Wert	Einheit
0x2341	1	PROFIBUS: Stationsadresse	4	

Stellen Sie sicher, dass der Index 0x2341:011 auf den Wert 1 (*„EMF2133IB (ID:2133)"*)[14] oder auf den Wert 2 (*„EMF2131IB (ID:2131)"*)[15] gesetzt ist.

Index	Subindex	Name	Wert	Einheit
0x2341	11	PROFIBUS: Kompatibilitätsmodus	EMF2133IB (ID: 0x2133) [1]	

Kompilieren, laden und starten Sie das Projekt auf dem i950-Laufwerk. Sie können nun das i950-Laufwerk auf die gleiche Weise wie das 9300-Laufwerk steuern.

Tipp

Nach dem Herunterladen muss der PROFIBUS-Slave möglicherweise neu gestartet werden, da sich die Werte in den Indizes 0x2341:001 und 0x2341:011 geändert haben.

Starten Sie die PROFIBUS-Kommunikation mit den aktuellen Einstellungen mit dem folgenden Befehl neu:

Index	Unterindex	Name	Wert	Einheit
0x2340	0	PROFIBUS-Kommunikation	keine Aktion / kein Fehler [0] Neustart mit aktuellen Werten [1 [2] Kommunikation stoppen [5] in Bearbeitung [10] Aktion abgebrochen [11] Fehler [12]	

[14] Diese Einstellung legt fest, welcher Gerätetyp über PROFIBUS an die Logik-SPS gemeldet wird. Eine Einstellung von 0x2341:11=1 lässt die Logik-SPS glauben, dass es sich bei dem angeschlossenen Gerät um ein 8200/9300 mit einem PROFIBUS-Modul EMF2133IB handelt.
[15] Eine Einstellung von 0x2341:11=2 lässt die Logik-SPS glauben, dass es sich bei dem angeschlossenen Gerät um ein 8200/9300 mit einem PROFIBUS-Modul EMF2131IB handelt.

4 Anhang

4.1 Unterstützte GSD-Konfigurationen[16]

#	GSD-Konfiguration	PROFIBUS-Konfiguration-swert(e)	Entsprechender Wert in 0x2348:003
1. kein Parameterkanal / Prozessdaten (Drivecom-Steuerung)			
1	PZD(1W)	0x70	xx0170
…	…	…	…
12	PZD(12W)	0x7B	xx017B
2. Konsistente Drivecom-Parameterkanal-/Prozessdaten (Drivecom-Steuerung)			
13	PAR(kons.) + PZD(1W)	0xF3, 0x70	xx02F370
…	…	…	..
24	PAR(cons.) + PZD(12W)	0xF3, 0x7B	xx02F37B
3. Konsistenter Drivecom-Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung)			
25	PAR(cons.) + PZD(1W cons.)	0xF3, 0xF0	xx02F3F0
…	…	…	…
36	PAR(cons.) + PZD(12W cons.)	0xF3, 0xFB	xx02F3FB
4. Drivecom-Parameterkanal / Prozessdaten (Drivecom-Steuerung)			
37	PAR + PZD(1W)	0x73, 0x70	xx027370
…	…	…	…
48	PAR + PZD(12W)	0x73, 0x7B	xx02737B
5. Drivecom-Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung)			
49	PAR + PZD(1W kons.)	0x73, 0xF0	xx0273F0
…	…	…	…
60	PAR + PZD (12 W Verbrauch)	0x73, 0xFB	xx0273FB
6. Kein Parameterkanal / konsistente Prozessdaten (Drivecom-Steuerung)			
61	PZD(1 W kons.)	0xF0	xx01F0
…	…	…	…
72	PZD (12 W Dauerleistung)	0xFB	xx01FB
7. Keine Parameterkanal-/Prozessdaten (Lenze-Gerätesteuerung)			
73	PZD(1 W) AR	0x00, 0x00, 0x00, 0x70	xx0400000070
…	…	…	…
84	PZD(12W) AR	0x00, 0x00, 0x00, 0x7B	xx040000007B
8. Konsistente Drivecom-Parameterkanal-/Prozessdaten (Lenze-Gerätesteuerung)			
85	PAR(kons.) + PZD(1W) AR	0x00, 0x00, 0x00, 0xF3, 0x70	xx05000000F370
…	…	…	…
96	PAR(kons.) + PZD(12W) AR	0x00, 0x00, 0x00, 0xF3, 0x7B	xx05000000F37B
9. Konsistenter Drivecom-Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung)			
97	PAR(cons.) + PZD(1W cons.) AR	0x00, 0x00, 0x00, 0xF3, 0xF0	xx05000000F3F0
…	…	…	…
108	PAR(cons.) + PZD(12W cons.) AR	0x00, 0x00, 0x00, 0xF3, 0xFB	xx05000000F3FB
10. Drivecom-Parameterkanal / Prozessdaten (Lenze-Gerätesteuerung)			
109	PAR + PZD(1W) AR	0x00, 0x00, 0x00, 0x73, 0x70	xx050000007370
…	…	…	…
120	PAR + PZD(12W) AR	0x00, 0x00, 0x00, 0x73, 0x7B	xx05000000737B
11. Drivecom-Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung)			
121	PAR + PZD(1W kons.) AR	0x00, 0x00, 0x00, 0x73, 0xF0	xx0500000073F0
…	…	…	…
132	PAR + PZD(12 W cons.) AR	0x00, 0x00, 0x00, 0x73, 0xFB	xx0500000073FB
12. Kein Parameterkanal / konsistente Prozessdaten (Lenze-Gerätesteuerung)			
133	PZD(1 W kons.) AR	0x00, 0x00, 0x00, 0xF0	xx04000000F0
…	…	…	…
144	PZD(12 W kons.) AR	0x00, 0x00, 0x00, 0xFB	xx04000000FB

[16] Keine Unterscheidung zwischen inkonsistenter/konsistenter Datenübertragung

4.2 AIF-IN-Schnittstelle des 9300

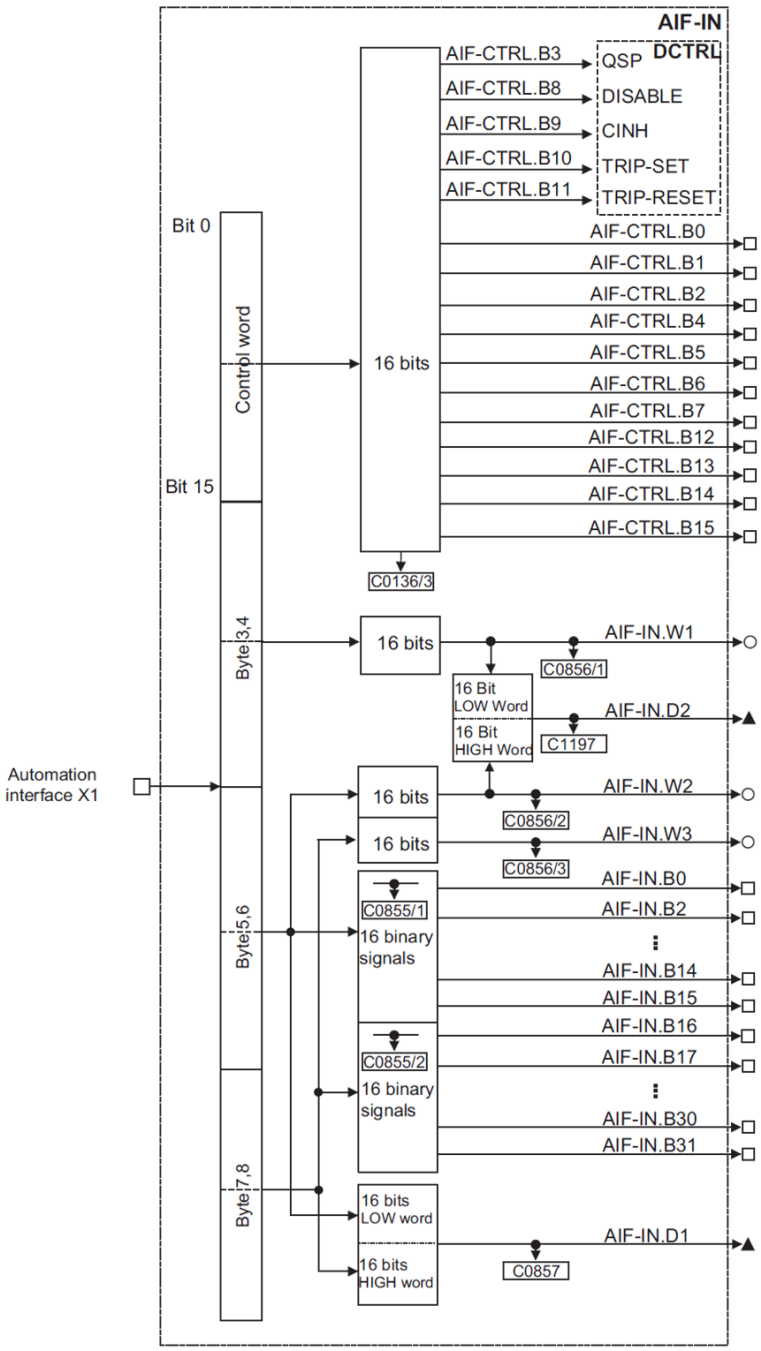


Abbildung46 : Signalfluss der AIF-IN-Schnittstelle am Servoumrichter 9300 (Auszug aus der GDC-Hilfe)

4.3 AIF-OUT-Schnittstelle des 9300

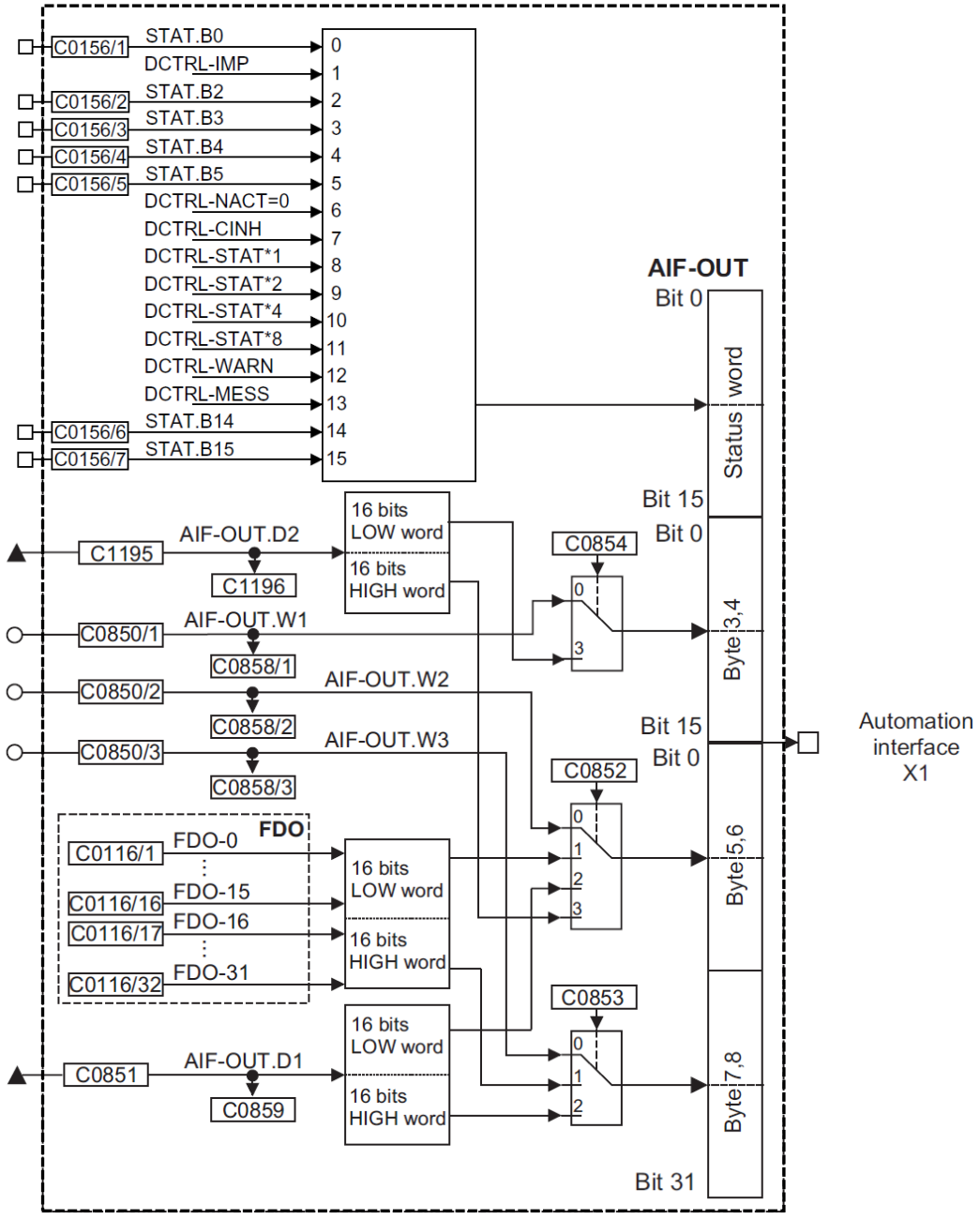


Abbildung47 : Signalfluss der AIF-OUT-Schnittstelle am Servoumrichter 9300 (Auszug aus der GDC-Hilfe)

4.4 Drivecom-Steuerwort

Bit	Name	Bedeutung
0	*Einschalten*	Befehlsbit: FALSE: Befehle 2, 6, 8 (Controller-Sperre) TRUE: Befehl 3 (Controller freigeben)
1	*Spannungshemmung*	Befehlsbit: Motor-Spannung deaktivieren/aktivieren FALSE: Spannung sperren TRUE: Spannung aktivieren
2	*Schnellstopp*	Befehlsbit: Schnellstopp aktivieren FALSE: Schnellstopp aktivieren TRUE: Schnellverschluss
3	*Betrieb aktivieren*	Befehlsbit: Antriebsbetrieb aktivieren FALSE: Antriebsbetrieb deaktivieren TRUE: Antriebsbetrieb aktivieren
4	*RFG-Sperre*	Befehlsbit: Schnellstopp der Anwendung (QSP) FALSE: Anwendungs-Schnellstopp (QSP) aktivieren TRUE: Anwendungs-Schnellstopp (QSP) freigeben **Hinweis:** Das negierte Signal dieses Bits wird direkt auf *scControlWords1.xQsp* ausgegeben.
5	*RFG-Stopp*	Befehlsbit: Stopp-Rampenfunktionsgenerator FALSE: Rampenfunktionsgenerator friert ein Der Antrieb hält die Istgeschwindigkeit aufrecht, auch wenn die Sollgeschwindigkeit auf *scControlWords1.iIn2* noch nicht erreicht ist. TRUE: Rampenfunktionsgenerator ist aktiv Der Antrieb beschleunigt/verzögert auf die Zielgeschwindigkeit auf *scControlWords1.iIn2*. **Hinweise:** • Das negierte Signal dieses Bits wird direkt auf *scControlWords1.xBit04* ausgegeben. • In der Basisanwendung „SpeedControl" hat Bit 5 (*RFG Stop*) eine geringere Priorität als Bit 6 (*RFG Zero*).
6	*RFG Zero*	Befehlsbit: Rampenabfahrt, Sollwert auf Null setzen FALSE: Null-Zielgeschwindigkeit Der Antrieb fährt auf eine Drehzahl von Null herunter. Der auf *scControlWords1.iIn2* empfangene Wert wird ignoriert. TRUE: Externe Sollgeschwindigkeit Der Antrieb folgt der Zielgeschwindigkeit auf *scControlWords1.iIn2*. **Hinweise:** • Das negierte Signal dieses Bits wird direkt auf *scControlWords1.xBit05* ausgegeben. • In der Basisanwendung „SpeedControl" hat Bit 6 (*RFG Zero*) Vorrang vor Bit 5 (*RFG Stop*).
7	*Fehlerrücksetzung*	Befehlsbit: Antriebsfehler zurücksetzen FALSE=>TRUE: Setzt einen Antriebsfehler zurück **Hinweise:** • Ein Laufwerksfehler kann nur zurückgesetzt werden, wenn die Fehlerursache zuvor beseitigt wurde. • Dieses Bit wird direkt auf *scControlWords1.xTripReset* ausgegeben.
8 … 10	(reserviert)	
11	Hersteller	freies Bit (direkt ausgegeben auf *scControlWords1.xBit07*)
12	Hersteller	freies Bit (direkt ausgegeben auf *scControlWords1.xBit12*)
13	Hersteller	freies Bit (direkt ausgegeben auf *scControlWords1.xBit13*)
14	Hersteller	Freies Bit (direkt ausgegeben auf *scControlWords1.xBit14*)
15	Hersteller	Freies Bit (direkt ausgegeben auf *scControlWords1.xBit15*)

4.5 Drivecom-Statuswort

Bit	Name	Bedeutung		
0	*Startbereit*	Informationen zur Gerätestatusmaschine:		
			FALSE	Der Gerätestatus ist niedriger als „*Bereit zum Starten*“.
			TRUE	Der Gerätestatus ist mindestens „*Bereit zum Starten*“.
1	*Eingeschaltet*	Informationen zur Gerätestatusmaschine:		
			FALSCH	Der Gerätestatus ist niedriger als „*Eingeschaltet*“.
			TRUE	Der Gerätestatus ist mindestens „*Eingeschaltet*“.
2	*Betrieb aktiviert*	Informationen zur Gerätestatusmaschine:		
			FALSCH	Der Gerätestatus ist niedriger als „*Betrieb aktiviert*“.
			TRUE	Der Gerätestatus ist mindestens „*Betrieb aktiviert*“.
3	*Fehler*	Das Gerät befindet sich im Fehlerzustand:		
			FALSCH	Auf dem Gerät ist kein Fehler aktiv.
			TRUE	Auf dem Gerät ist ein Fehler aktiv.
		Hinweis: Das Signal wird aus *scStatusWords1.xStat8*, *scStatusWords1.xStat10* und *scStatusWords1.xStat11* abgeleitet.		
4	*Spannung gesperrt*	Handshake-Signal: Rückgabe des Steuerbits 1 („*Spannungshemmung*“)		
			FALSE	kein Fehler auf dem Gerät aktiv
			TRUE	Ein Fehler ist auf dem Gerät aktiv.
		Hinweis: Das Signal wird direkt aus Bit 1 des Drivecom-Steuerworts kopiert (siehe vorheriges Kapitel „4.4 “).		
5	*Quick Stop*	Handshake-Signal: Rückgabe von Steuerbit 2 (*Quick Stop*)		
			FALSE	Schnellstoppbefehl ist auf dem Gerät aktiv
			TRUE	Es ist kein Schnellstoppbefehl auf dem Gerät aktiv.
		Hinweis: Das Signal wird direkt aus Bit 2 oder Bit 4 des Drivecom-Steuerworts kopiert (siehe vorheriges Kapitel „4.4 “).		
6	*Einschalten gesperrt*	Informationen zur Zustandsmaschine des Geräts:		
			FALSE	Das Gerät befindet sich nicht im Zustand „*Einschalten gesperrt*“.
			TRUE	Das Gerät befindet sich im Zustand „*Einschalten gesperrt*“.
7	*Warnung*	Das Gerät befindet sich im Warnzustand:		
			FALSE	Es ist keine Warnung auf dem Gerät aktiv.
			TRUE	Eine Warnung ist auf dem Gerät aktiv.
		Hinweis: Das Signal dieses Bits wird direkt aus *scStatusWords1.xWarning* kopiert.		
8	*Meldung*	Die Meldung ist auf dem Gerät aktiv:		
			FALSE	Es ist keine Meldung auf dem Gerät aktiv.
			TRUE	Eine Nachricht ist auf dem Gerät aktiv.
		Hinweise: • Ein Nachrichtenstatus tritt typischerweise bei Unterspannung auf (Hauptstrom abgeschaltet). • Das Signal dieses Bits wird direkt aus *scStatusWords1.xMessage* kopiert.		
9	*Fern*	Feldbus-Zugriffsberechtigung:		
			FALSE	-
			TRUE	(dieses Signal ist im Drivecom-Betriebsmodus immer auf TRUE gesetzt)
10	*Sollwert erreicht*	Status des internen Rampengenerators:		
			FALSE	Die tatsächliche Antriebsdrehzahl entspricht nicht dem Sollwert.
			TRUE	Die tatsächliche Antriebsdrehzahl entspricht dem Sollwert.
		Hinweise: • Bei der Standardgeschwindigkeitsregelung repräsentiert das Signal den Status „*Sollwert erreicht*“ des Geschwindigkeitsrampengenerators. In diesem Fall können die folgenden Drivecom-Befehlsbits das Signal „*Sollwert erreicht*“ unterdrücken: – *RFG-Sperre* (Befehlsbit 4) – *RFG-Stopp* (Befehlsbit 5) – *RFG Zero* (Befehlsbit 6) • Im Allgemeinen wird das Signal dieses Bits direkt aus *scStatusWords1.xBit04* kopiert.		
11	*Grenzwert*	Status der Drivecom-Drehzahlbegrenzung (nicht unterstützt):		
			FALSE	(dieses Signal ist im Drivecom-Betriebsmodus immer auf FALSE gesetzt)
			TRUE	-
12	Hersteller	freies Bit (Signal direkt aus *scStatusWords1.xBit14* kopiert)		
13	Hersteller	freies Bit (Signal direkt aus *scStatusWords1.xBit03* kopiert)		
14	Hersteller	freies Bit (Signal direkt aus *scStatusWords1.xBit02* kopiert)		
15	Hersteller	Freies Bit (Signal direkt aus *scStatusWords1.xBit05* kopiert)		

4.6 Drivecom DP V0 Parameterkanal (Tx)

Die folgende Tabelle beschreibt die Bedeutung der vom PLC an das Slave-Gerät (Antrieb) gesendeten Anforderung des Sendeparameterkanals (8 Byte):

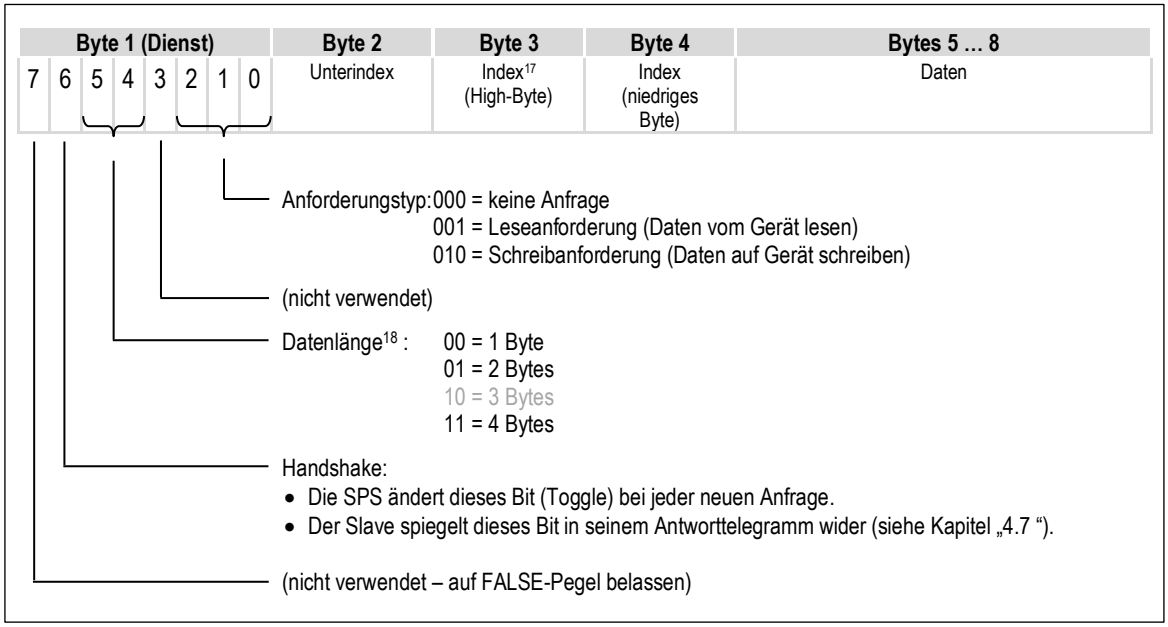


Abbildung „48 “: Struktur des Drivecom DP V0-Parameterkanals Tx-Telegramm am Servoumrichter 9300 (SPS => Antrieb)

[17] Die 9300-Indexnummer ergibt sich aus der Subtraktion der 9300-Codenummer von einem festen Wert von 24575 (=0x5FFF).
[18] Länge der Daten in Bytes 5 ... 8 (Daten/Fehler 1 ... 4), die in den Slave-Geräteindex gelesen/geschrieben werden sollen

4.7 Drivecom DP V0-Parameterkanal (Rx)

Die folgende Tabelle beschreibt die Bedeutung der Antwort des Empfangsparameterkanals (8 Byte), die vom Slave-Gerät (Antrieb) an die SPS zurückgegeben wird:

Byte 1 (Dienst)								Byte 2	Byte 3	Byte 4	Bytes 5 ... 8
7	6	5	4	3	2	1	0	Unterindex	Index[19] (High-Byte)	Index (niedriges Byte)	Daten / Fehlercode

Spiegelung der Anforderungstyp-Bits 0 … 2 (siehe Kapitel „2.1.4 ")

(nicht verwendet)

Datenlänge[20] : 00 = 1 Byte
01 = 2 Bytes
10 = 3 Bytes
11 = 4 Bytes

Spiegelung des Handshake-Bits 6 des Tx-Telegramms (siehe Kapitel 2.2.16):
- Die SPS ändert dieses (Toggle-)Bit bei jeder neuen Anfrage.
- Das Slave-Gerät kopiert das Bit in sein Antworttelegramm.

Statusbit:
Statusinformation vom Slave-Gerät an die SPS beim Senden der Anforderungsbestätigung. Dieses Bit informiert die Master-SPS darüber, ob die Anforderung fehlerfrei ausgeführt wurde.

0 = Anfrage ohne Fehler abgeschlossen
(Die Daten der Bytes 5 ... 8 stellen die aus dem Zielindex gelesenen Daten dar.

1 = Anfrage nicht abgeschlossen – ein Fehler ist aufgetreten.
(Die Daten der Bytes 5 ... 8 stellen die Fehlernummer dar.)

Die folgenden Fehlercodes werden zurückgegeben, wenn Bit 7 auf TRUE gesetzt ist:

Byte 5	Byte 6	Byte 7	Byte 8	Fehlerbeschreibung
0x00	0x00	0x00	0x08	Innerhalb der Watchdog-Zeit konnte keine Antwort auf eine Anfrage empfangen werden.
0x00	0x00	0x03	0x06	Zugriff auf diesen Parameter nicht zulässig.
0x00	0x00	0x07	0x06	Codenummer existiert nicht in der Parameterreferenzliste
0x00	0x00	0x08	0x06	Datentypen stimmen nicht überein
0x01	0xFE	0x00	0x08	Ungültiger Dienst (keine Lese- oder Schreibanforderung)
0x10	0x00	0x05	0x06	Zielindex-/Unterindexnummer existiert nicht auf dem Gerät
0x11	0x00	0x05	0x06	Die Untercode-Nummer ist in der Parameterreferenzliste nicht vorhanden.
0x12	0x00	0x05	0x06	Die Datenlänge des zu schreibenden Werts ist zu groß.
0x13	0x00	0x05	0x06	Die Datenlänge des zu lesenden Werts ist zu klein.
0x30	0x00	0x00	0x08	Schreibzugriff nicht verweigert, da Laufwerksbetrieb aktiviert ist
0x31	0x00	0x00	0x08	Obergrenze des Parameters nicht erreicht
0x32	0x00	0x00	0x08	Unterer Grenzwert des Parameters wird nicht erreicht

Abbildung49 : Struktur des Drivecom DP V0-Parameterkanals Rx-Telegramm am Servoumrichter 9300 (Antrieb => SPS)

[19] Die Indexnummer 9300 ergibt sich aus der Subtraktion der Codenummer 9300 von einem festen Wert von 24575 (=0x5FFF).
[20] Länge der Rückgabedaten in Bytes 5 ... 8 (Daten/Fehler 1 ... 4)