

## 0.1 Implementationsphase

In diesem Unterkapitel findet sich die Dokumentation der Implementation sämtlicher Modelle aus der Projektierungsphase wieder. Zunächst wird im Unterunterabschnitt 0.1.1 die Montage der Laboranlage kurz dargestellt. Dazu wird aus der Konfiguratorskizze unter Berücksichtigung der Maße des Laborraumes und der entsprechenden Anforderungen an den Aufbau des Positioniersystems die Anlage aufgebaut. Nachdem alle Gehäuseelemente an der Wand und dem Boden verankert sind können die Steuerungskomponenten, die Aktuatoren und die Sensoren an diesen befestigt werden. Weiterhin umfasst der nachfolgende Abschnitt die Verdrahtung der elektrischen Komponenten nach entwickeltem Stromlaufplan und der Netzwerkdarstellung aus dem ??.

Im zweiten Unterabschnitt (Unterunterabschnitt 0.1.2) wird die Implementation der modellierten Diagramme zur Steuerungssoftware vorgenommen. Da durch die Wahl der Steuerung (Logic Motion Controller von Schneider Electric), wie bereits in der Anforderungsanalyse festgestellt, die Umgebung zur Programmierung der Systemsoftware festgelegt ist, besteht die Möglichkeit Templates aus dieser zu nutzen, welche den Softwareentwicklungsprozess vereinfachen. Der Unterabschnitt zur Software-Implementation behandelt somit die schrittweise Darstellung der Umsetzung der Automatisierungssoftware aus dem von Schneider Electric bereitgestellten *MotionTemplateFull*.

### 0.1.1 Hardware-Implementation

Für die Umsetzung der Hardware wird die Konfiguratorgrafik aus der Konzeptphase aufgegriffen und gilt als Grundlage für die reale Umsetzung der Hardwarebereiche des Systems. Da im Konfigurator bereits die Kernanforderungen an die Systemhardware berücksichtigt wurden, müssen beim Bau und der Montage des Gehäuses bzw. des Anlagengerüsts nur noch die nicht-funktionalen Anforderungen an dieses und den umliegenden Raum berücksichtigt werden.

Durch die Form und die Ausmaße des Laborraumes ergibt sich eine maximale Höhe des Gehäuses von 2230mm. Die horizontale Ausdehnung der Anlage wird nicht durch den Raum begrenzt. Die Entscheidung wurde aufgrund von subjektiven Anschaulichkeitskriterien getroffen. Resultat ist eine horizontale Ausdehnung der x-Achse von 2000mm, was ungefähr der Gangbreite im Laborraum entspricht. Folglich ist der bewegliche Teil des Positioniersystems mittig zum Durchgang im Raum ausgerichtet. Sowohl rechts als auch links neben der Positioniereinheit ist ein Bereich von jeweils 600 mm reserviert, in dem Ablagepositionen an der Wand befestigt werden können. Die rechte Seite der Anlagenkonstruktion besitzt zusätzlich noch ein weiteres Aluminium Profil, welches später benötigt wird, um den Schaltschrank und die Steuerungshardware am System zu fixieren. Nachfolgende Grafik zeigt das an der Laborraumwand montierte Anlagengerüst, an welches im nächsten Schritt die Steuerungshardware befestigt wird.



Abbildung 1: Anlagengerüst des Gehäuses vom mehrachsigen Positioniersystem an der hinteren Laborwand im Raum G 422

Im nächsten Schritt der Hardware-Implementation werden die Steuerung (LMC400c) das Netzgerät (LXM 62P) und der Servoregler (LXM 62D) an Querverstrebungen der beiden rechten Profile verschraubt. Es gilt die Montageanleitung zu beachten.

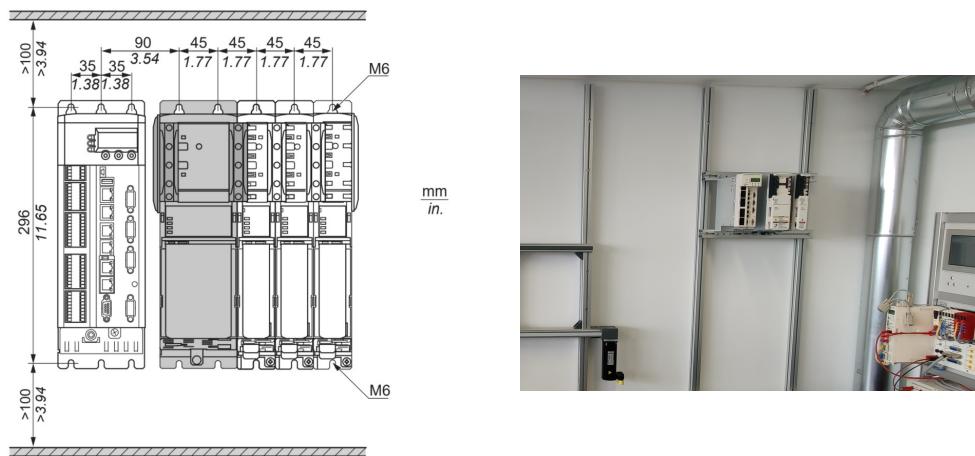


Abbildung 2: Installation der Steuerungshardware am Gehäuseaufbau des Positioniersystems

Bevor die in der Konfiguratorskizze aufgeführten Sensoren und Aktuatoren verbaut

werden können, sind noch weitere Profile notwendig, um den Gehäusebau zu komplettieren. 500 mm von der Wand entfernt auf Höhe der äußeren Profile des Positionierbereiches werden zwei senkrechte Aluminiumstempel an Decke und Boden befestigt. Diese sind das vordere Ende des mehrachsigen Positioniersystems. Das Gehäusegerüst ist nun vollständig und umschließt den Bereich, in dem Positionieraufgaben durchgeführt werden können mit der Laboranlage.

Zwischen den beiden linken Profilen und den beiden rechten Profilen werden Plexiglasscheiben angebracht, die das Hineingreifen in den Fahrerbereich der Anlage von den Seiten verhindern sollen. An der Front wird ein Lichtvorhang bestehend aus Emitter und Receiver montiert. Dieser befindet sich auf der Innenseite an den zuletzt angebauten senkrechten Stempeln. Der Lichtvorhang dient ebenso wie die Plexiglasscheiben zum Schutz von Leib und Leben. Im Gegensatz zu den Scheiben erlaubt der Lichtvorhang jedoch im unbewegten Zustand des Systems das Eindringen von Personen in den Arbeitsbereich.

Folgende Grafik zeigt zu den soeben genannten Komponenten zusätzlich noch die in der Anforderungsanalyse ermittelten vier Endlagesensoren, die Servomotoren für x- und z-Achse, sowie E-Ketten zu den beweglichen Achsen.



Abbildung 3: Installation von Sensoren und Aktuatoren des mehrachsigen Positioniersystems

Der Schaltschrankneinbau und die Verdrahtung des Systems stellt den letzten Schritt in der Hardware-Implementierung dar. Als Schaltschrank wurde ein Modell der Firma Rittal gewählt. Die Breite des Schrankes ist durch den gewählten Aufbau bereits festgelegt und beträgt 600mm. Aufgrund der Anzahl der Klemmen und dem Wunsch **ea!**-Module

sowie weitere Steuerungskomponenten physisch von normalen Klemmen zu trennen, wurde entschieden eine Schaltschrankschaltung zu wählen, die in der Höhe zwei Hutschienen unterbringt. Die gewählte Schrankhöhe liegt deswegen bei 380mm. Zuletzt muss die Tiefe des Schaltschranks ausreichend sein, um die tiefste Komponente, die im Schaltschränk verbaut werden soll, unterbringen zu können. Da bereits die kleinste verfügbare Variante des gewählten Schrankes diese Anforderung erfüllt, hat der zu verbauende Schaltschränk eine Tiefe von 210mm.

Wie bereits angedeutet ist in der Grafik unten zu erkennen, dass die untere der beiden Hutschienen sämtliche Klemmen beherbergt inklusive der Absicherungen für die jeweilige Spannungsebene, so wie im Stromlaufplan geplant. An der oberen Hutschiene befinden sich die Modicon TM5 ea!-Module, die als Erweiterung für die Ein- und Ausgänge des **lmc!** dienen. Die im Bild als rot gefärbte Komponente zu erkennende Steuerung, ist der Safety Logic Controller, der für die Berechnung der Sicherheitsfunktionen des Systems verantwortlich ist. Dazu besitzt dieser jeweils vier digitale Ein- und Ausgänge.

Rechts daneben auf derselben Schiene ist die Wago PFC200 Steuerung angebracht, welche mithilfe ihrer Energieklemme die Leistungsaufnahme des Systems messen soll.

Ganz Rechts auf der Hutschiene ist ein Ethernetswitch montiert, an welchen beide Steuerungen (LMC400c und Wago PFC200) per Ethernetkabel angeschlossen sind. Durch den Einsatz des Switches führt nach Fertigstellung der Verdrahtung nur ein Kabel zur Programmierung der beiden Steuerungen aus dem Schaltschränk heraus.

Weiterhin ist auf der rechten Schrankwand der Hauptschalter platziert. Über dieser aktiviert oder deaktiviert die Stromversorgung des Schaltschranks und somit auch aller Systemkomponenten.

Die Verdrahtung erfolgt nach Stromlaufplan. Dieser beinhaltet auch die Kopplung der einzelnen Module aus der PacDrive 3 Serie, die verbaut wurden (LMC400c, LXM 62P, LXM 62D, Modicon TM5 Module, Modicon TM5 SLC100 Sicherheitsmodule).

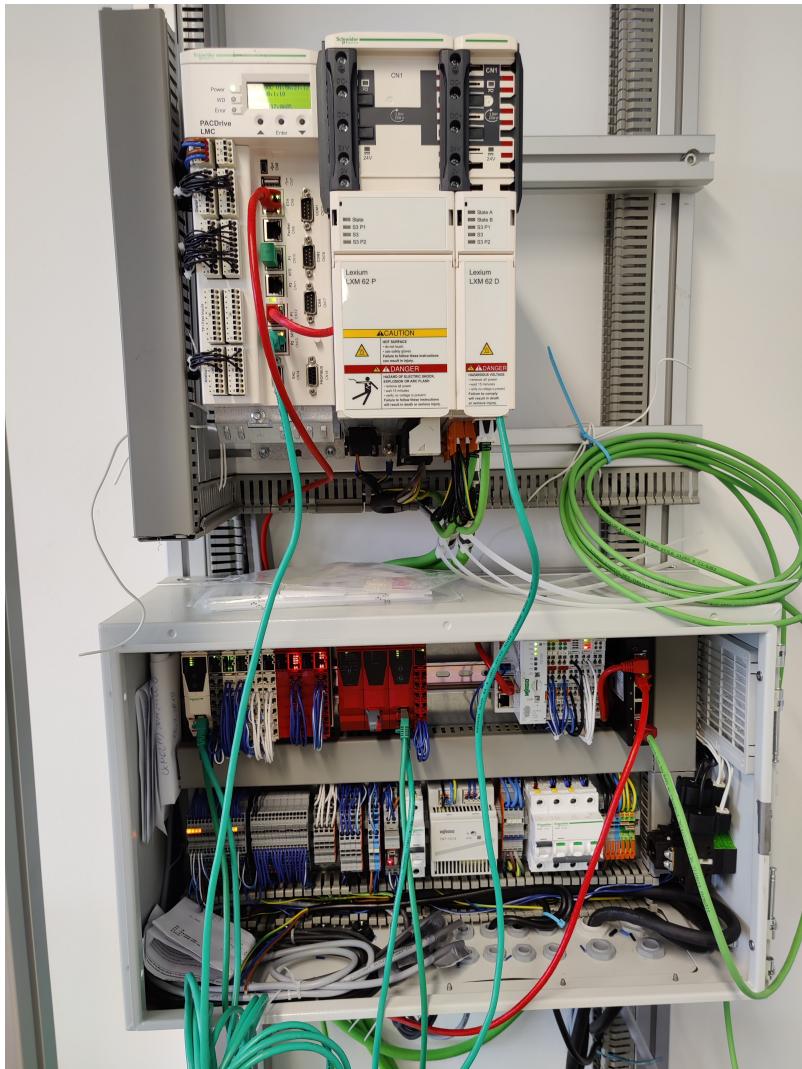


Abbildung 4: Einbau und Verdrahtung des Schaltschanks

### 0.1.2 Software-Implementation

In diesem Unterabschnitt wird die Realisierung der Automatisierungssoftware dokumentiert. Grundlegend soll aus der Modellierung der Software im vorhergegangenen Kapitel (Projektierung) das Programm für das mehrachsige Positioniersystem geschrieben werden. Wie bereits in der Einleitung zu diesem Unterkapitel erwähnt, resultiert die Wahl der Steuerungskomponenten, also konkret die Wahl Komponenten aus der PacDrive 3 Serie von Schneider Electric auszuwählen, in dem Zwang mit der auf Codesys 3.5 basierten Entwicklungsumgebung MachineExpert zu arbeiten. Diese besitzt bereits mehrere Templateprojekte für verschiedene Anwendungsfälle. Die Firma Schneider Electric empfiehlt

die Nutzung des jeweiligen Programmtemplates für den gewünschten Anwendungsfall. Grund dafür ist die Minimierung vom Programmieraufwand. Ziel soll es sein lediglich Konfigurationen an einem modularen Template vorzunehmen, sodass dieses die eigenen Anforderungen erfüllt.

Begründet durch diese Aussagen wird zunächst davon ausgegangen, dass durch die Nutzung des *MotionTemplateFull* mit überschaubarem Programmier- und Parametriereraufwand die Sammlung der eigenen Anforderungen an das Automatisierungsprogramm und dessen Funktionen erfüllt werden. Dies gilt im anschließenden Unterkapitel in den jeweiligen Testfällen zu bestätigen.

Zunächst findet eine schrittweise Auflistung zur Umsetzung der Steuerungssoftware aus dem bereitgestellten Template statt. Diese unterteilt sich in folgende Vorgehensschritte:

- Anlegen des Projektes
- Zuweisen der Ein- und Ausgänge der Steuerungen und deren Erweiterungsmodule (engl. Mapping)
- Parametrierung und Inbetriebnahme des Servoreglers für die x-Achse
- Parametrierung und Inbetriebnahme des Servoreglers für die z-Achse
- Parametrierung und Inbetriebnahme des Netzteils
- Implementation der funktionalen Sicherheit

Für das **Anlegen des Projektes** wird die Software MachineExpert LogicBuilder auf den Rechnern des Labores benötigt. Alternativ kann auch die veraltete Software SoMachine LogicBuilder genutzt werden. Soll von jedem PC aus das System programmiert und zunächst auch gesteuert werden, so muss die Programmiersoftware vorhanden sein. Ist dies der Fall, so muss anschließend wie folgt vorgegangen werden:

1. *LogigBuilder* Programm am Laborcomputer starten
2. Neues Projekt aus Projektvorlage *MotionTemplateFull* für LMC300/400/402/600/800 anlegen
3. Im Projektbaum oben links Doppelklick auf *LMC\_PacDrive*
4. Anschließend im Reiter *Steuerungsauswahl* den LMC400c auswählen (vorher sichergehen, dass der Controller eingeschaltet ist, sonst kann dieser nicht gefunden werden)
5. Im Projektbaum unter *Sercos\_Master* (*Sercos Master*) alle untergeordneten Geräte markieren und anschließen entfernen

6. Etwas tiefer im Projektbaum Doppelklick auf *Geräteaddressierung*
7. Im neu geöffneten Fenster oben rechts den Button *Sercos Scan starten* drücken
8. Es erscheinen neue tabellarisch angeordnete Einträge in der Mitte des Fensters (Einträge sind rot eingefärbt)
9. Unten rechts im offenen Fenster auf *Geräteparameter übernehmen* klicken (Alle vorhandenen Einträge sollten die Farbe zu grün wechseln)
10. Den *IEC Bezeichner* des Netzteils ändern zu *PSM\_PowerSupply*
11. Den *IEC Bezeichner* des von Drive A und B ändern zu *DRV\_Slave1* bzw. *DRV\_Slave2*
12. Oben links im offenen Fenster über Combobox drei neue LXM62DxS hinzufügen
13. *IEC Bezeichner* des ersten neuen Eintrages auf *DRV\_Master* ändern (Wichtig: Gerät sollte auf virtuell eingestellt sein)
14. *IEC Bezeichner* der beiden anderen Einträge ändern zu *DRV\_Slave3* bzw. *DRV\_Slave4* (Wichtig: Geräte sollte auf virtuell eingestellt sein)

Geräte der Steuerungskonfiguration							Gescannte Geräte		
Topolog. Adresse	IEC-Bezeichner	Typ	Geräte-Seriennr.	Motor-Seriennumm	Anwendungstyp	SERCOS-A	Identifikationsmodus	Betriebsart	
1	PSM_PowerSupply	LXM62PS	2910053203		PowerSupply	1	Topologische A... ▾	Real ▾	...
2	DRV_Slave1	LXM62dS	2910018244-A	2800573624	Drive A	2	Topologische A... ▾	Real ▾	...
3	DRV_Slave2	LXM62dS	2910018244-B	2800573623	Drive B	3	Topologische A... ▾	Real ▾	...
4	SLC_TMSCSLCx...	TM5CSLC10...	B360170926		Modular Safety O...	4	Topologische A... ▾	Real ▾	...
5	BC_TMNS31	TM5NS31	B37C0196882		Modular IO Device	5	Topologische A... ▾	Real ▾	...
6	DRV_Slave3	LXM62dS			DRV_LexumS2	100	Topologische A... ▾	Virtuell ▾	...
7	DRV_Slave4	LXM62dS			DRV_LexumS2	100	Topologische A... ▾	Virtuell ▾	...
8	DRV_Master	LXM62dS			DRV_LexumS2	100	Topologische A... ▾	Virtuell ▾	...

Farbenlegende:  
■ Login fehlerfrei möglich   ■ Kein fehlerfreier Login möglich   ■ Kein gescanntes Gerät zugeordnet  
■ Login fehlerfrei möglich; Unterschiede in irrelevanten Werten werden jedoch hervorgehoben  
■ Login fehlerfrei möglich; gescanntes Gerät wird jedoch nicht zugeordnet  
■ Login möglich; eingesetzter Identifikationsmodus wird jedoch nicht unterstützt

Werte aller zugeordneten Geräte übernehmen

Abbildung 5: Geräteaddressierung der SERCOS III Busteilnehmer

Um die anliegenden Sensor- und Eingabegerätesignale an den SPS-Eingängen, sowie die Aktuatoren und Indikatoren an den SPS-Ausgängen nutzen zu können, müssen den Hardwareadressen im Programm Variablen zugewiesen werden (engl. **Mapping**). Damit alle **ea!**-Variablen an einem Ort als auch global im gesamten Programm verfügbar sind, sollte eine globale Variablenliste angelegt werden, in der alle Variablen eingetragen werden können. Folgendes Bild zeigt die globale Variablenliste für die Ein- und Ausgänge des mehrachsigen Positioniersystems.

```

2 | VAR_GLOBAL
3 | // Modicon TM5 digital Inputs
4 | q_ixIN_00: BOOL; // Endl. Oben
5 | q_ixIN_01: BOOL; // Endl. Unten
6 | q_ixIN_02: BOOL; // Endl. Links
7 | q_ixIN_03: BOOL; // Endl. Rechts
8 | q_ixIN_04: BOOL; // Tst. Rtg. Oben
9 | q_ixIN_05: BOOL; // Tst. Rtg. Unten
10 | q_ixIN_06: BOOL; // Tst. Rtg. Links
11 | q_ixIN_07: BOOL; // Tst. Rtg. Rechts
12 | q_ixIN_08: BOOL; // Tst. Grün
13 | q_ixIN_09: BOOL; // Tst. Rot
14 | q_ixIN_10: BOOL; // Tst. Weiß Oben
15 | q_ixIN_11: BOOL; // Tst. Weiß Unten
16 | q_ixIN_12: BOOL; // Schit. Links (Betr. Mod.)
17 | q_ixIN_13: BOOL; // Schit. Rechts (Betr. Mod.)
18 | q_ixIN_14: BOOL; // LV. OSSDI
19 | q_ixIN_15: BOOL; // LV. OSSD2
20 |
21 | // Modicon TM5 digital Outputs
22 | q_qxOUT_00: BOOL; // Sign. A. Rot
23 | q_qxOUT_01: BOOL; // Sign. A. Grün
24 | q_qxOUT_02: BOOL; // Tst. LED Grün
25 | q_qxOUT_03: BOOL; // Tst. LED Rot
26 | q_qxOUT_04: BOOL; // Tst. LED Weiß Oben
27 | q_qxOUT_05: BOOL; // Tst. LED Weiß Unten
28 | q_qxOUT_06: BOOL; // ---
29 | q_qxOUT_07: BOOL; // ---
30 | q_qxOUT_08: BOOL; // ---
31 | q_qxOUT_09: BOOL; // ---
32 | q_qxOUT_10: BOOL; // ---
33 | q_qxOUT_11: BOOL; // ---
34 | q_qxOUT_12: BOOL; // ---
35 | q_qxOUT_13: BOOL; // ---
36 | q_qxOUT_14: BOOL; // ---
37 | q_qxOUT_15: BOOL; // ---
38 |
39 | // Modicon TM5 analog Inputs
40 | q_ixIN_00: REAL; // Pot. Oben (z-Achse)
41 | q_ixIN_01: REAL; // Pot. Unten (x-Achse)
42 | q_ixIN_02: REAL; // ---
43 | q_ixIN_03: REAL; // ---
44 |
45 | // Modicon TM5 analog Outputs
46 | q_qxOUT_00: REAL; // ---
47 | q_qxOUT_01: REAL; // ---
48 | q_qxOUT_02: REAL; // ---
49 | q_qxOUT_03: REAL; // ---
50 |
51 | // Modicon TM5 SIC100 safe digital Inputs
52 | q_ixSAFE_IN_00: BOOL; // Not-Halt Offnen
53 | q_ixSAFE_IN_01: BOOL; // Not-Halt Schlieder
54 | q_ixSAFE_IN_02: BOOL; // LV. OSSDI
55 | q_ixSAFE_IN_03: BOOL; // LV. OSSD2
56 |
57 | // Modicon TM5 SIC100 safe digital Outputs
58 | q_ixSAFE_OUT_00: BOOL; // ---
59 | q_ixSAFE_OUT_01: BOOL; // ---
60 | q_ixSAFE_OUT_02: BOOL; // ---
61 | q_ixSAFE_OUT_03: BOOL; // ---
62 |
63 | // Release of the safety Outputs (required by the safety Controller)
64 | q_ixRelease_00: BOOL;
65 | q_ixRelease_01: BOOL;
66 | q_ixRelease_02: BOOL;
67 | q_ixRelease_03: BOOL;
68 |
69 | // Safety Variables
70 | hardwareOK : BOOL; // Check if Safety Modules are working
71 | hError_Out : BOOL; // Error Status from Safety Blocks
72 | hNotHalt : BOOL; // Emergency Stop Status
73 | hOSSDI : BOOL; // Lightcurtain 1
74 | hOSSD2 : BOOL; // Lightcurtain 2
75 |
76 END_VAR

```

Abbildung 6: Globale Variablenliste der ea!-Variablen

Nachfolgend wird das Mapping schrittweise für die Ein- und Ausgangsmodule der mit dem LMC verbundenen Modicon TM5 Geräte vorgenommen. Bei den zuzuordnenden Variablen handelt es sich um die im Datenmodell aufgelisteten Variablen. Dieses dient somit als Grundlage für die Zuweisung der Ein- und Ausgänge.

- Zuweisung der digitalen Eingänge des Modicon *TM5SDI16D* Modul:

1. Im Geräetebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) ->*BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SDI16D* (*TM5SDI16D*)

2. Öffnen des Reiters *SERCOS III Module E/A-Abbild*

3.

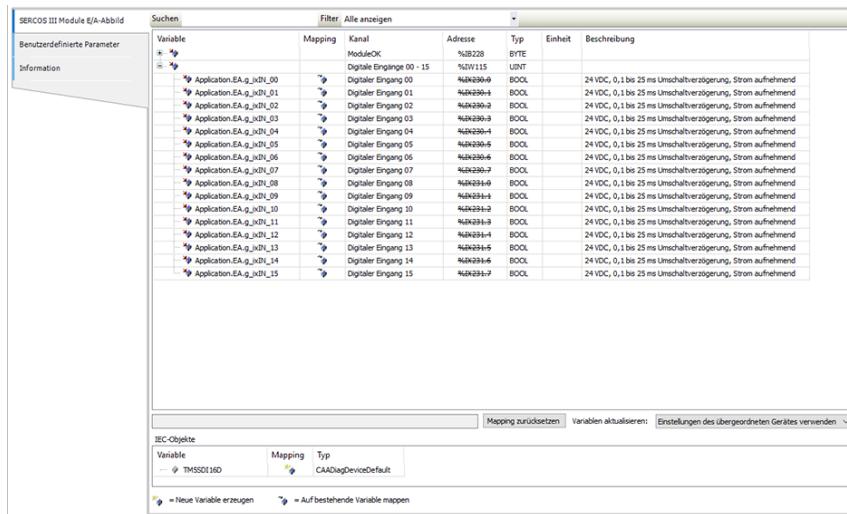


Abbildung 7: Zuweisung der digitalen Eingänge des TM5SDI16D Moduls

- Zuweisung der digitalen Ausgänge des Modicon *TM5SDO16T* Modul:

- Im Gerätebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) -> *BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SDO16T* (*TM5SDO16T*)
- Öffnen des Reiters *SERCOS III Module E/A-Abbild*

3.

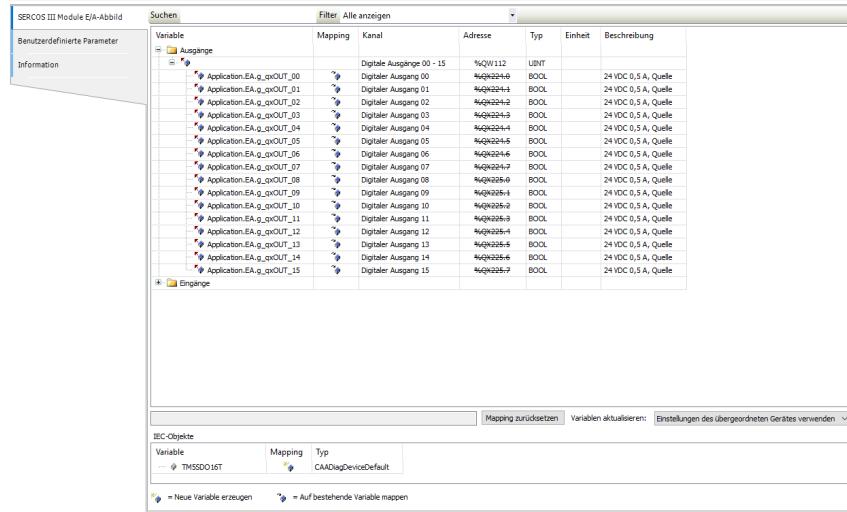


Abbildung 8: Zuweisung der digitalen Ausgänge des TM5SDO16T Moduls

- Zuweisung der sicheren digitalen Eingänge des Modicon *TM5SDI4DFS* Modul:

- Im Gerätebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) -> *BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SDI4DFS*
- Öffnen des Reiters *TM5 Modul E/A-Abbild*

3.

TMS Modul E/A-Abbild						
	Suchen	Filter	Alle anzeigen			
TMS Modul Parameter	Variable	Mapping	Kanal	Adresse	Typ	Einheit
Benutzerdefinierte Parameter	= Application.EA_g_xsAFE_IN_00	"	SafeDigitalInput01	%I0:248	BYTE	
	= Application.EA_g_xsSAFE_IN_01	"	SafeDigitalInput02	%I0:249	BOOL	Sicherer digitaler Eingang 01 (24VDC)
	= Application.EA_g_xsSAFE_IN_02	"	SafeDigitalInput03	%I0:250	BOOL	Sicherer digitaler Eingang 02 (24VDC)
	= Application.EA_g_xsSAFE_IN_03	"	SafeDigitalInput04	%I0:251	BOOL	Sicherer digitaler Eingang 03 (24VDC)
			SafeDigitalInput05	%I0:252	BOOL	Sicherer digitaler Eingang 04 (24VDC)
			SafeDigitalInput06	%I0:253	BOOL	Zwei Kanalauflösung äquivalent sicherer digitaler Eingang 01/02
			SafeDigitalInput07	%I0:254	BOOL	Zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04
			SafeDigitalInput08	%I0:255	BOOL	Zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04
			SafeDigitalInput09	%I0:256	BOOL	Zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04
			SafeDigitalInput10	%I0:257	BOOL	Zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04
			SafeDigitalInput11	%I0:258	BOOL	
			SafeDigitalInput12	%I0:259	BOOL	Status Kanal sicherer digitaler Eingang 02 (1=0Q)
			SafeDigitalInput13	%I0:260	BOOL	Status Kanal sicherer digitaler Eingang 03 (1=0Q)
			SafeDigitalInput14	%I0:261	BOOL	Status Kanal sicherer digitaler Eingang 04 (1=0Q)
			SafeDigitalInput15	%I0:262	BOOL	Status zwei Kanalauflösung äquivalent sicherer digitaler Eingang 01/02
			SafeDigitalInput16	%I0:263	BOOL	Status zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04
			SafeDigitalInput17	%I0:264	BOOL	Status zwei Kanalauflösung äquivalent sicherer digitaler Eingang 03/04

IEC-Objekte

Variable	Mapping	Typ
M5SDO4DFS	CADaylightDeviceDefault	

= Neue Variable erzeugen    = Auf bestehende Variable mappen

Abbildung 9: Zuweisung der sicheren digitalen Eingänge des TM5SDI4DFS Moduls

- Zuweisung der sicheren digitalen Ausgänge des Modicon *TM5SDO4TFS* Modul:

1. Im Gerätebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) -> *BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SDO4TFS*
2. Öffnen des Reiters *TM5 Modul E/A-Abbild*

3.

TMS Modul E/A-Abbild						
	Suchen	Filter	Alle anzeigen			
TMS Modul Parameter	Variable	Mapping	Kanal	Adresse	Typ	Einheit
Benutzerdefinierte Parameter	= Application.EA_g_xsSAFE_OUT_00	"	SafeDigitalOutput1-4	%Q0:234	BYTE	Sicherer digitaler Ausgang 01 (Zustimmignal, 24VDC / 0.5A)
	= Application.EA_g_xsSAFE_OUT_01	"	SafeDigitalOutput01	%Q0:235	BOOL	Sicherer digitaler Ausgang 02 (Zustimmignal, 24VDC / 0.5A)
	= Application.EA_g_xsSAFE_OUT_02	"	SafeDigitalOutput02	%Q0:236	BOOL	Sicherer digitaler Ausgang 03 (Zustimmignal, 24VDC / 0.5A)
	= Application.EA_g_xsSAFE_OUT_03	"	SafeDigitalOutput03	%Q0:237	BOOL	Sicherer digitaler Ausgang 04 (Zustimmignal, 24VDC / 0.5A)
			Status Module	%I0:252	BOOL	

IEC-Objekte

Variable	Mapping	Typ
TM5SDO4TFS	CADaylightDeviceDefault	

= Neue Variable erzeugen    = Auf bestehende Variable mappen

Abbildung 10: Zuweisung der sicheren digitalen Ausgänge des TM5SDO4TFS Moduls

- Zuweisung der analogen Eingänge des Modicon *TM5SAI4L* Modul:

1. Im Gerätebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) -> *BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SAI4L*
2. Öffnen des Reiters *TM5 Module E/A-Abbild*

3.

The screenshot shows the 'TMS Module E/A-Abbildung' configuration window. The main table lists analog inputs (Analogaufgang) mapped to addresses %QW113 to %QW119. The table includes columns for Variable, Mapping, Kanal, Adresse, Typ, Einheit, and Beschreibung. Below the table, there is an IEC-Objekte section with a table for variable creation and mapping. Buttons for 'Mapping zurücksetzen', 'Variablen aktualisieren', and 'Einstellungen des übergeordneten Gerätes verwenden' are visible at the bottom.

Suchen							Filter	Alle anzeigen
Variable	Mapping	Kanal	Adresse	Typ	Einheit	Beschreibung		
% Application.EA_g_AIN_00			%DB236	BYTE				
		Status Eingang 00						
% Application.EA_g_AIN_01			%DB237	BYTE				
% Application.EA_g_AIN_02			%DB238	BOOL				
% Application.EA_g_AIN_03			%DB239	BOOL				
% Application.EA_g_AIN_04			%DB240	BOOL				
% Application.EA_g_AIN_05			%DB241	BOOL				
% Application.EA_g_AIN_06			%DB242	BOOL				
% Application.EA_g_AIN_07			%DB243	BOOL				
% Application.EA_g_AIN_08			%DB244	BOOL				
% Application.EA_g_AIN_09			%DB245	BOOL				
% Application.EA_g_AIN_10			%DB246	BYTE				
		Analoginput 0-3						

IEC-Objekte		
Variable	Mapping	Typ
TMSSAI4L		CAADaqDeviceDefault

=> Neue Variable erzeugen    => Auf bestehende Variable mappen

Abbildung 11: Zuweisung der analogen Eingänge des TM5SAI4L Moduls

- Zuweisung der analogen Ausgänge des Modicon *TM5SAO4L* Modul:

- Im Gerätebaum öffnen der Geräteeinstellungen des Moduls unter *Sercos\_Master* (*Sercos Master*) ->*BC\_TM5NS31* (*BC\_TM5NS31*) -> *TM5SAO4L*
- Öffnen des Reiters *TM5 Module E/A-Abbild*

3.

The screenshot shows the 'TMS Module E/A-Abbildung' configuration window. The main table lists analog outputs (Analogausgang) mapped to addresses %QW113 to %QW119. The table includes columns for Variable, Mapping, Ausgang, Adresse, Typ, Einheit, and Beschreibung. Below the table, there is an IEC-Objekte section with a table for variable creation and mapping. Buttons for 'Mapping zurücksetzen', 'Variablen aktualisieren', and 'Einstellungen des übergeordneten Gerätes verwenden' are visible at the bottom.

Suchen							Filter	Alle anzeigen
Variable	Mapping	Ausgang	Adresse	Typ	Einheit	Beschreibung		
% Application.EA_g_AOUT_00		Analoger Ausgang 00	%QW113	INT		#10 V / 0 bis 20 mA, Auflösung 12 Bits		
% Application.EA_g_AOUT_01		Analoger Ausgang 01	%QW114	INT		#10 V / 0 bis 20 mA, Auflösung 12 Bits		
% Application.EA_g_AOUT_02		Analoger Ausgang 02	%QW115	INT		#10 V / 0 bis 20 mA, Auflösung 12 Bits		
% Application.EA_g_AOUT_03		Analoger Ausgang 03	%QW116	INT		#10 V / 0 bis 20 mA, Auflösung 12 Bits		
% Application.EA_g_AOUT_04		ModuleOK	%DB246	BYTE				

IEC-Objekte		
Variable	Mapping	Typ
TMSSAO4L		CAADaqDeviceDefault

=> Neue Variable erzeugen    => Auf bestehende Variable mappen

Abbildung 12: Zuweisung der analogen Ausgänge des TM5SAO4L Moduls

- Zuweisung der Ein- bzw. Ausgänge der *SLC\_TM5CSLC* Sicherheitssteuerung für den Datenaustausch mit dem *lmc!*:

- Im Gerätebaum öffnen der Geräteeinstellungen der Sicherheitssteuerung unter *Sercos\_Master* (*Sercos Master*) ->*SLC\_TM5CSLCx00FS* (*SLC\_TM5CSLCx00FS*)
- Öffnen des Reiters *E/A-Abbild*

3.

Parameter	Suchen				
E/A-Abbildung	Filter	Alle anzeigen			
<b>Information</b>					
Variable	Mapping	Kanal	Adresse	Typ	Einheit
SLC2MC_BOOL		SLC2MC_BOOL0-7	%AB0	BYTE	
Application.EA.bError_Out		Bo00	%AB0-0	BOOL	
Application.EA.bHardwareOK		Bo01	%AB0-1	BOOL	
Application.EA.bIOSD1		Bo02	%AB0-2	BOOL	
Application.EA.bIOSD2		Bo03	%AB0-3	BOOL	
Application.EA.bIOSD3		Bo04	%AB0-4	BOOL	
Application.EA.bIOSD4		Bo05	%AB0-5	BOOL	
Application.EA.bIOSD5		Bo06	%AB0-6	BOOL	
Application.EA.bIOSD6		Bo07	%AB0-7	BOOL	
SLC2MC_BOOL8-15		SLC2MC_BOOL8-15	%AB1	BYTE	
SLC2MC_BOOL16-23		SLC2MC_BOOL16-23	%AB2	BYTE	
SLC2MC_BOOL24-31		SLC2MC_BOOL24-31	%AB3	BYTE	
SLC2MC_BOOL32-39		SLC2MC_BOOL32-39	%AB4	BYTE	
SLC2MC_BOOL40-47		SLC2MC_BOOL40-47	%AB5	BYTE	
SLC2MC_BOOL48-55		SLC2MC_BOOL48-55	%AB6	BYTE	
SLC2MC_BOOL56-63		SLC2MC_BOOL56-63	%AB7	BYTE	
SLC2MC_BOOL64-71		SLC2MC_BOOL64-71	%AB8	BYTE	
SLC2MC_BOOL72-79		SLC2MC_BOOL72-79	%AB9	BYTE	
SLC2MC_BOOL80-87		SLC2MC_BOOL80-87	%AB10	BYTE	
SLC2MC_BOOL88-95		SLC2MC_BOOL88-95	%AB11	BYTE	
SLC2MC_BOOLExt					
SLC2MC_INT					
SLC2MC_LINT					
SLC2MC_LINTExt					
UMC2SLC_BOOL					
UMC2SLC_BOOLExt					
UMC2SLC_INT					
UMC2SLC_LINT					
<b>IEC-Objekte</b>					
Variable	Mapping	Typ			
SLC_TM5CSLCx00FS		SLC_PLC			
<a href="#">Neue Variable erzeugen</a> <a href="#">Auf bestehende Variable mappen</a>					

Abbildung 13: Zuweisung der Variablen für den Datentransfer vom **slc!** zum **lmc!**

4.

Parameter	Suchen				
E/A-Abbildung	Filter	Alle anzeigen			
<b>Information</b>					
Variable	Mapping	Kanal	Adresse	Typ	Einheit
SLC2MC_BOOL		SLC2MC_BOOL0-7	%QD0	BYTE	
SLC2MC_BOOLExt		Bo00	%QD0-0	BOOL	
SLC2MC_INT		Bo01	%QD0-1	BOOL	
SLC2MC_LINT		Bo02	%QD0-2	BOOL	
SLC2MC_LINTExt		Bo03	%QD0-3	BOOL	
UMC2SLC_BOOL		Bo04	%QD0-4	BOOL	
UMC2SLC_BOOLExt		Bo05	%QD0-5	BOOL	
UMC2SLC_INT		Bo06	%QD0-6	BOOL	
UMC2SLC_LINT		Bo07	%QD0-7	BOOL	
UMC2SLC_BOOL8-15		UMC2SLC_BOOL8-15	%QB1	BYTE	
UMC2SLC_BOOL16-23		UMC2SLC_BOOL16-23	%QB2	BYTE	
UMC2SLC_BOOL24-31		UMC2SLC_BOOL24-31	%QB3	BYTE	
UMC2SLC_BOOL32-39		UMC2SLC_BOOL32-39	%QB4	BYTE	
UMC2SLC_BOOL40-47		UMC2SLC_BOOL40-47	%QB5	BYTE	
UMC2SLC_BOOL48-55		UMC2SLC_BOOL48-55	%QB6	BYTE	
UMC2SLC_BOOL56-63		UMC2SLC_BOOL56-63	%QB7	BYTE	
UMC2SLC_BOOL64-71		UMC2SLC_BOOL64-71	%QB8	BYTE	
UMC2SLC_BOOL72-79		UMC2SLC_BOOL72-79	%QB9	BYTE	
UMC2SLC_BOOL80-87		UMC2SLC_BOOL80-87	%QB10	BYTE	
UMC2SLC_BOOL88-95		UMC2SLC_BOOL88-95	%QB11	BYTE	
SLC2MC_BOOLExt					
SLC2MC_INT					
SLC2MC_LINT					
SLC2MC_LINTExt					
<b>IEC-Objekte</b>					
Variable	Mapping	Typ			
SLC_TM5CSLCx00FS		SLC_PLC			
<a href="#">Neue Variable erzeugen</a> <a href="#">Auf bestehende Variable mappen</a>					

Abbildung 14: Zuweisung der Variablen für den Datentransfer vom **lmc!** zum **slc!**

Für die Nutzung der beiden Achsen des Systems muss die **Parametrierung und die Inbetriebnahme des Servoreglers** vorgenommen werden. Physisch gesehen handelt es sich zwar um ein Gerät, dass zwei Servoantriebe betreiben kann, in der Konfiguration im Programm werden die x-Achse und die z-Achse jedoch separat in Betrieb genommen. Die Parametrierung und die Inbetriebnahme erfolgt für beide Achsen analog, da es sich bei beiden um die selben Motoren handelt, die in der selben Konfiguration genutzt werden. Zunächst müssen einige physikalische Daten der beiden Achsen und deren zugehörigen Hardwarekomponenten in der Geräteparametrierung aufgenommen werden:

1. Im Projektbaum öffnen der Datei *Application -> TemplateFullProgrammingFramework -> EquipmentModules -> SR\_BravoModule (PRG) -> Init\_Slave1*
2. Bis zum Kommentar **\*\*\*Manual\*\*\*** scrollen (Zeile 171)
3. 

```
/* Manual */
stSlaveInterface.stManual.i_lrVel           := 50.0;      /* Velocity in units/s */
stSlaveInterface.stManual.i_lrAcc          := 10.0;      /* Acceleration in units/s^2 */
stSlaveInterface.stManual.i_lrDec          := 1000.0;    /* Deceleration in units/s^2 */
stSlaveInterface.stManual.i_lrJerk         := 1000.0;    /* Jerk in units/s^3 */
stSlaveInterface.stManual.i_lrMaxDistance  := 120.0;     /* Max pathlengths for one step in units */
stSlaveInterface.stManual.i_xEndless       := FALSE;     /* TRUE: jogging endless. Position between the periods */
stSlaveInterface.stManual.i_lrPeriod        := 360.0;     /* Period of the axis */
```

Abbildung 15: Einstellen der Bewegungsparameter für langsame manuelle Testfahrt

rVel gibt die Geschwindigkeit in mm/s an, rAcc die Beschleunigung in mm/s<sup>2</sup> und rDec die negative Beschleunigung mm/s<sup>2</sup>.

4. Im Projektbaum öffnen der Gerätedatei *Sercos\_Master (Sercos Master) -> DRV\_Slave1*

5.	
----	--

Abbildung 16: Parametrierung des Motors für die x-Achse

Der Wert *MotorTemperatureMonitoring* wurde zur Temperaturüberwachung des Servomotors auf *Thermisches Modell* gesetzt.

6.	
----	--

Abbildung 17: Parametrierung der Mechanik für die x-Achse

Der Wert *Direction* gibt den Drehsinn des Servomotors an. Die Einstellung *rechts / 1* bedeutet, dass bei der Bedienung des Motors über die Steuerungsvisualisierung die Eingabe *Jog+* zu einer Bewegung der Achse nach links führt. Der Wert *FeederConstant*

muss selbst berechnet werden aus dem Motordurchmesser ( $\pi * M60\text{mm}$ ). Der *JLoad* Wert gibt das Lastenträgheitsmoment in  $\text{kg} * \text{cm}^2$  an. Der einzusetzende Wert kann über die Software *MachineExpert MotionBuilder* bestimmt werden.

Die gleichen Schritte müssen für die z-Achse analog zu der obigen Beschreibung durchgeführt werden.

Für den sicheren Betrieb der Achsen ist es erforderlich im *MotionTemplateFull* Programmänderungen vorzunehmen, damit die Achsen bei den Endlagesensoren des Positioniersystems stoppen und nicht darüber hinaus bewegt werden können. Folgende Schritte sind dazu nötig:

1. Im Projektbaum öffnen der Datei *Application -> TemplateFullProgrammingFramework -> EquipmentModules -> SR\_BravoModule (PRG) -> SubModules\_Action*
2. Bis zum Abschnitt 3 Scrollen und diesen auskommentieren oder löschen
3. Abschnitt 4 (beinhaltet den Funktionsbaustein und Variablenzuweisungen für den *DRV\_Slave2* bzw. die z-Achse) kopieren und wieder einfügen (duplicieren)
4. Die Kopie editieren, sodass an jeder Stelle, wo *DRV\_Slave2* aufgeführt ist nun *DRV\_Slave1* steht
- 5.

```

4 (* End of system generated code, could be modified for your application *)
fbSlave1Module
FB_AxisModuleExtIf
astSubModuleInterface[c_udiSlave1].i_xEnable -- i_xEnable
stSlave1Interface.stManual.i_xJogForw -- i_xJogForw
stSlave1Interface.stManual.i_xJogBack -- i_xJogBack
stSlave1Interface.stMain.i_xHwLimitPos -- i_xHwLimitPos
stSlave1Interface.stMain.i_xHwLimitNeg -- i_xHwLimitNeg
astSubModuleInterface[c_udiSlave1].iq_diCmd -- iq_diCmd
astSubModuleInterface[c_udiSlave1] -- iq_stStandardModuleItf
iq_stExceptionList -- iq_stExceptionList
iq_stLogDataList -- iq_stLogDataList
stSlave1Interface -- iq_stAxisModuleItf
q_xActive
q_xReady
q_etDiag
q_etDiagExt
q_smMsg
q_xCmdActive
q_xCmdDone
q_xDriveActive
q_xHomeOk
q_xSynchron
q_lrPosition
q_lrPositionX

5 (* End of system generated code, could be modified for your application *)
fbSlave2Module
FB_AxisModuleExtIf
astSubModuleInterface[c_udiSlave2].i_xEnable -- i_xEnable
stSlave2Interface.stManual.i_xJogForw -- i_xJogForw
stSlave2Interface.stManual.i_xJogBack -- i_xJogBack
stSlave2Interface.stMain.i_xHwLimitPos -- i_xHwLimitPos
stSlave2Interface.stMain.i_xHwLimitNeg -- i_xHwLimitNeg
astSubModuleInterface[c_udiSlave2].iq_diCmd -- iq_diCmd
astSubModuleInterface[c_udiSlave2] -- iq_stStandardModuleItf
iq_stExceptionList -- iq_stExceptionList
iq_stLogDataList -- iq_stLogDataList
stSlave2Interface -- iq_stAxisModuleItf
q_xActive
q_xReady
q_etDiag
q_etDiagExt
q_smMsg
q_xCmdActive
q_xCmdDone
q_xDriveActive
q_xHomeOk
q_xSynchron
q_lrPosition
q_lrPositionX

```

Abbildung 18: Einbinden der Endlagesensoren

Sowohl für den *DRV\_Slave1* als auch den *DRV\_Slave2* die globalen Variablen für die Endlagesensoren an den Stellen *i\_xHwLimitPos* bzw. *i\_xHwLimitNeg* einsetzen.

Im nächsten Schritt wird die **Parametrierung und Inbetriebnahme des Netzteils** durchgeführt. Dafür muss zunächst wieder eine Hardwareparametrierung nach folgenden Schritten vorgenommen werden:

1. Im Projektbaum öffnen der Gerätedatei *Sercos\_Master (Sercos Master) -> PSM\_PowerSupply*

Allgemeines und Status	DINT			
DiagClass	DINT			Diagnoseklasse (AD)[0x00CD]
DiagCode	DINT			Diagnosenummer (AD)[0x00001]
DiagSource				Diagnosequelle (AD)[0x00CE]
udiType	UDINT	0	0	Diagnosetext (AD)[0x0003]
udiInstance	UDINT	0	0	Erweiterte Diagnosemeldung (AD)
udiParameterId	UDINT	0	0	Status der Leistungsversorgung-I
DiagMsg	STRING(39)	=	=	Leistungsversorgung-Überwachu
DiagExtMsg	STRING(14)	=	=	Phasenüberwachungsmodus (EP)
PowerSupplyCheckSet	Enumeration of BOOL	aus / 0	aus / 0	Netzspannung (ES)[0x0000]
PowerSupplyCheck	Enumeration of BOOL			Bremswiderstand Modus (EP)[0]
PhaseCheckMode	Enumeration of DINT	Dreiphasige Überwachung / 2	Dreiphasige Überwachung / 2	Betriebsbereit (AD)[0x0008]
MainsVoltageMode	Enumeration of DINT	400V / 2	400V / 2	Betriebszustand (Netzteil und Anl)
BrakingResistorMode	Enumeration of DINT	400V Modus / 1	400V Modus / 1	Bereit fuer Auftragserarbeitung
Ready	Enumeration of BOOL			Gerätezustand (AS)[0x0010]
GroupState	Enumeration of DINT	Initialisierung / 0	Initialisierung / 0	Überlast des Netzteils (AS)[0x0011]
GroupReady	Enumeration of BOOL			Temperatur der Steuerplatine (A!
InternalDeviceState	Enumeration of DINT	Initialisierung / 0	Initialisierung / 0	Temperatur der Gleichrichters (A!
PowerSupplyOverload	INT		%	
AutoDischarge	Enumeration of BOOL	Aktiviere automatische Entladung / 1	Aktiviere automatische Entladung / 1	
ControlBoardTemp	INT(-10..150)	0	0   °C	
RectifierTemp	INT(-10..150)	0	0   °C	

Abbildung 19: Konfiguration des Netzteils

Der *PhaseCheckerMode* muss auf *Dreiphasige Überwachung / 2* eingestellt werden, da das Netzteil dreiphasig angeschlossen und genutzt wird. Der *MainsVoltageMode* wird gesetzt auf *400V / 2* und der *BrakingResistorMode* wird ebenfalls gesetzt auf *400V Modus / 1*. Diese Werte ergeben sich ebenfalls aus dem dreiphasigen Anschluss des Netzteils.

Je nachdem, ob bei der Verdrahtung ein Netzschütz verbaut oder diese weggelassen wurde, müssen nun weitere Programmänderungen am *MotionTemplateFull* vorgenommen werden. An erster Stelle wird erklärt, wie ohne ein Netzschütz vorgegangen werden muss:

1. Im Projektbaum öffnen der Datei *Application -> TemplateFullProgrammingFramework -> TaskCalls -> SR\_MainMachine (PRG) -> Input\_Action*

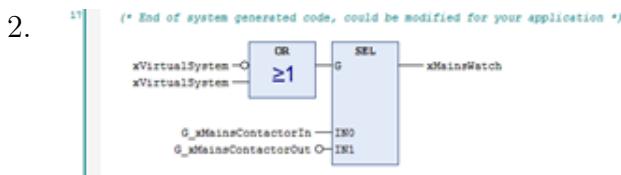


Abbildung 20: Deaktivierung der Netzüberwachung

Das Netzwerk in Abschnitt 17 muss erweitert werden um einen *OR-Funktionsbaustein*, zu dem die Eingangsvariable *xVirtualSystem* einmal negiert und einmal nicht-negiert zugewiesen wird. Diese Änderung deaktiviert die Netzüberwachung. Grundsätzlich sollte dieser Weg nicht gewählt und ein Netzschatz verbaut werden.

Bei der Verwendung eines Netzschatzes sollte wie folgt vorgegangen werden:

1. Mappen der Eingangsvariablen am LMC für die Netzschatzrückmeldung
2. Im Projektbaum öffnen der Datei *Application -> TemplateFullProgrammingFramework -> TaskCalls -> SR\_MainMachine (PRG) -> Input\_Action*
3. Zuweisen der globalen Variablen für *G\_xMainsContactorIn* und *G\_xMainsContactorOut*

Im nächsten Schritt der Software-Implementation findet die **Implementation der funktionalen Sicherheit** statt. Dazu wird zunächst der Safety Logic Controller (**slc!**) konfiguriert:

1. Im Projektbaum öffnen der Gerätedatei *Sercos\_Master (Sercos Master) -> SLC\_TM5CSLCx00FS (TM5CSLCx00FS)*

2.	
----	--

Abbildung 21: I/O-Konfiguration des **slc!**

Der Wert *SLC2LMC\_NumberOfBOOLs* muss auf *8 Bool / 1* gesetzt werden, sodass bis zu 8 Bool-Werte vom Safety Logic Controller an den **lmc!** übertragen werden können. Analog muss der Wert *LMC2SLC\_NumberOfBOOLs* auf *8 Bool / 1* gesetzt werden, dass bis zu 8 Bool Werte vom **LMC!** an den **SLC!** gesendet werden können.

3. Mapping der neu verfügbaren **ea!**-Variablen von bzw. zu dem **slc!** (siehe Abbildung 13 und Abbildung 14)

Anschließend kann das Programm für die Software-Implementation der funktionalen Sicherheit geschrieben werden. Dazu wird wie folgt vorgegangen:

1. Im Projektbaum Rechtsklick auf *Sercos\_Master (Sercos Master) -> SLC\_TM5CSLCx00FS (TM5CSLCx00FS)*

2. Im sich geöffneten Menü den Punkt *SoSafe -> SoSafe Programmable starten* auswählen
3. Zunächst muss sowohl ein Passwort für die Entwicklung und die Inbetriebnahme des Sicherheitsprogramms festgelegt werden. In dem der Arbeit beigefügten Programm ist das Passwort admin1 für sowohl die Entwicklung als auch die Inbetriebnahme festgelegt worden.
4. Es müssen im sich nun geöffneten Dialogfenster alle Module (drei Module) per Checkbox ausgewählt werden
5. Nun kann mit der Programmierung der Sicherheitsfunktionalitäten begonnen werden in der Funktionsbausteinsprache (FUP) nach IEC 61131-3

6.

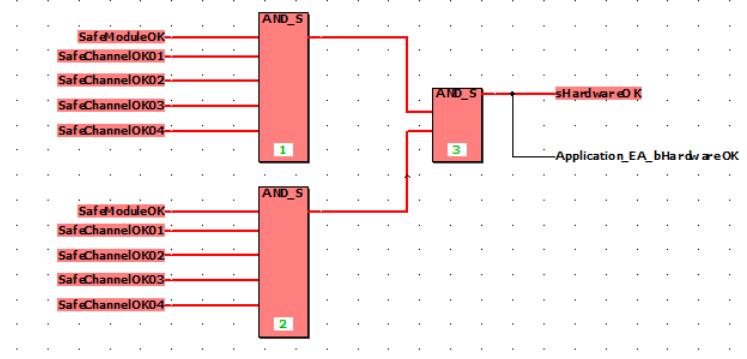


Abbildung 22: Hardwareprüfung der sicheren **ea!**-Module

Die sichere Variable *sHardwareOK* wird auf den Wert **TRUE** gesetzt, wenn sowohl die beiden **ea!**-Module keine Hardwarefehler aufweisen und die Verdrahtung der Eingänge und Ausgänge korrekt ist. Weiterhin wird der Wert über die Variable *Application\_EA\_bHardwareOK* auch an den **lmc!** weitergeleitet, um im Hauptprogramm genutzt zu werden (z. B. für die Ausgabe über OPC UA).

7.

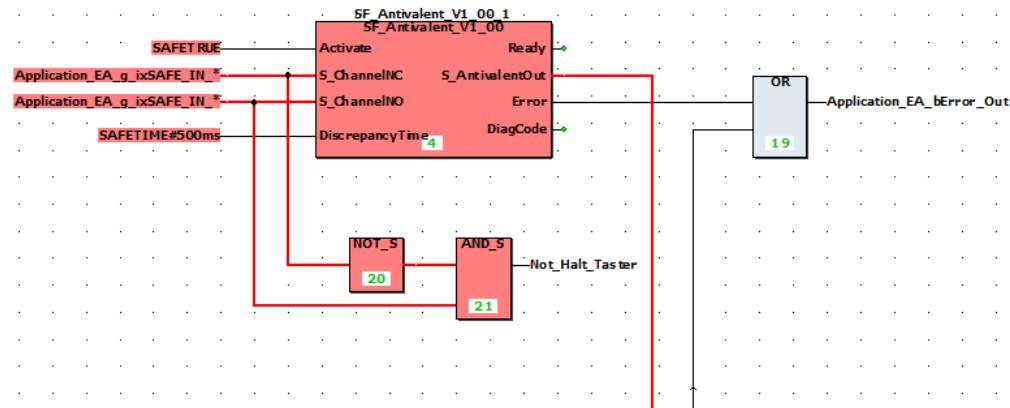


Abbildung 23: Sicherer SF\_Antivalent Funktionsbaustein für die Verarbeitung der Not-Halt-Auslösung

Bei dem sicheren Funktionsbaustein **SF\_Antivalent\_V1\_00\_1** handelt es sich um einen Baustein, der die beiden antivalenten Eingänge für die Not-Halt-Taster Auslösung verarbeitet. Ist die Variable **Application\_EA\_g\_ixSafe\_IN\_01 FALSE** und die Variable **Application\_EA\_g\_ixSafe\_IN\_00 TRUE**, so schaltet der Ausgang **S\_AntivalentOut** auf **TRUE**. Der Wert des Ausgangs wird im nächsten Bild weiterverarbeitet. Für die Weitergabe des gleichen Wertes an den **lmc!** wird der Baustein nicht benötigt und die beiden Eingänge können per **AND\_S**-Baustein verundet werden (unter Berücksichtigung, dass der Eingang **Application\_EA\_g\_ixSafe\_IN\_01** negiert werden muss). Der **Error** Ausgang kann auch verbunden und die Information an den **lmc!** weitergegeben werden in der Variable **Application\_EA\_bError\_Out**.

8.

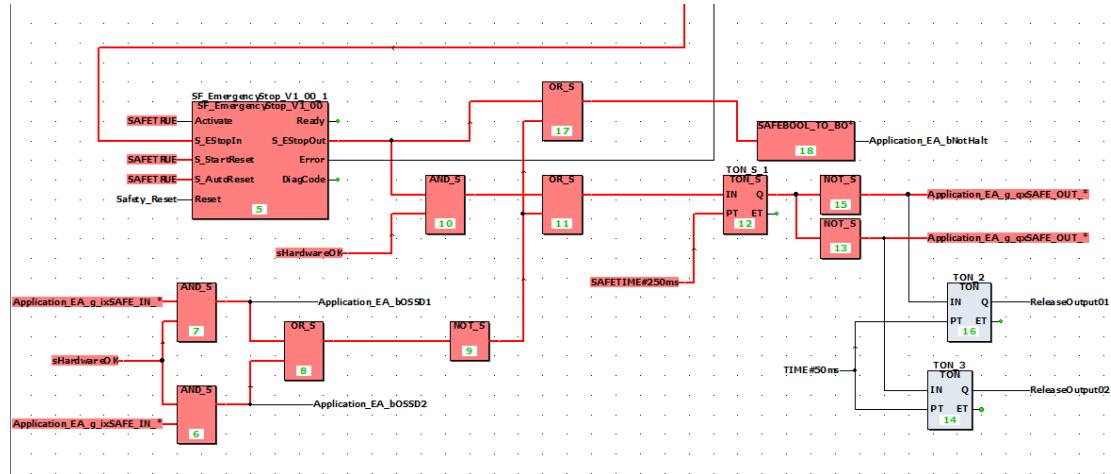


Abbildung 24: Verarbeitung des Not-Halt-Signals und der Lichtvorhangssignale

9. Zunächst wird der sichere Funktionsbaustein *SF\_EmergencyStop\_V1\_00\_1* hinzugefügt. In den Eingang *S\_EStopIn* wird das Ausgangssignal des Antivalent-Funktionsbausteins gelegt. Wird der Not-Halt aktiviert, schaltet der Ausgang *S\_EStopOut* des Bausteins auf **TRUE**. Über den *Reset*-Eingang muss der Baustein wieder freigegeben werden, sodass der Ausgang wieder auf **FALSE** umschaltet. Auf den Reset-Eingang wird die Variable *>Safety\_Reset* gelegt, die vom **lmc!** an den **slc!** weitergeleitet wird. Somit kann durch das Auslösen des Hardware Reset-Tasters am **lmc!** der Funktionsbaustein wieder freigegeben bzw. zurückgesetzt werden. Die Eingänge *S\_StartReset* und *S\_AutoReset* sollten auf **SAFEFALSE** gesetzt werden. Im Bild ist jedoch vorerst zu erkennen, dass die Werte auf **SAFETRUE** gesetzt sind, da zum Testen des Sicherheitsprogramms ein automatisches Zurücksetzen das Drücken des Reset-Tasters hinfällig macht und somit Zeit spart.
10. Anschließend werden die beiden sicheren Eingänge *Application\_EA\_g\_ixSafe\_IN\_02* und *Application\_EA\_g\_ixSafe\_IN\_03* verarbeitet. Dabei handelt es sich um den Lichtvorhang (OSSD1 und OSSD2). Die beiden Eingänge müssen jeweils verundet werden mit der Variable *bHardwareOK*. Ist eines der beiden Signale des Lichtvorhangs **FALSE**, so muss der Ausgang, dessen Signal später mit dem Not-Halt-Signal verordert wird **TRUE** sein (umgesetzt durch ein *S\_ODER*- und ein *S\_Not*-Baustein).
11. Wie bereits im vorherigen Schritt angedeutet wird nun das Ausgangssignal der LichtvorhangSchaltung mit dem Ausgang der Not-Halt-Schaltung verordnet. Wichtig ist, dass das Not-Halt-Signal auch mit dem Wert der Variablen *sHardwareOK* verundet wird, um Hardwarefehler auszuschließen.
12. Im nächsten Schritt wird das Not-Halt-Signal zu einem nicht-sicheren Bool-Wert umgewandelt, welcher über die Variable *Application\_EA\_bNotHalt* an den **lmc!** übergeben wird.
13. Der letzte Schritt in der Programmierung des Sicherheitsprogrammes ist das setzen der Ausgänge des **slc!**. Wichtig ist zu beachten, dass die Ausgänge mit dem *InverterEnable*-Eingang des Servoreglers (LXM 62D) verbunden sind. Nehmen die Ausgänge den Wert **FALSE** an, schaltet der Servoregler ab. Das soll dann passieren, wenn der Not-Halt ausgelöst bzw. der Lichtvorhang durchbrochen wurde. Jedoch ist es notwendig, dass die Motoren zunächst angehalten haben, bevor der Regler abgeschaltet wird, da sonst bei zu frühem Abschalten des Reglers der Bremsvorgang nicht fortgesetzt wird und die beiden Achsen des Positioniersystems austrudeln würden. Dies könnte dafür sorgen, dass die Schlitten auf den Achsen über die Endlagen hinausrutschen und Beschädigungen an der Anlage verursachen. Deshalb wird über ein *S\_TON*-Baustein die Abschaltung der Ausgänge um 250ms verzögert. In Abbildung 24 ist zu erkennen, dass zwei weitere nicht-sichere TONs genutzt wurden. Diese sind für das Lösen der Ausgänge nach einem Signalwechsel zuständig. Es handelt sich

dabei um eine Sicherheitsmaßnahme, so das Ausgänge nach Wertänderung immer gesteuert wieder freigegeben werden.

Nachdem nun das Sicherheitsprogramm komplettiert ist, kann die SoSafe Programmable Umgebung wieder verlassen werden. Abschließend muss eine letzte Änderung am *MotionTemplateFull* vorgenommen werden, um den Not-Halt auch auf **lmc!**-Seite zu implementieren. Dazu sind folgende Schritte nötig:

1. Im Projektbaum öffnen der Datei *Application -> TemplateFullProgrammingFramework -> TaskCalls -> SR\_HWCopyIO*

2.



Abbildung 25: Zuweisung der Not-Halt-Variable im **lmc!**

Der internen Variable *G\_xEmergencyIn* wird die Not-Halt-Ausgangsvariable aus dem **slc!** zugewiesen. Nun bekommt auch das *MotionTemplateFull* das reale Not-Halt-Signal zur Verfügung gestellt und kann dieses verarbeiten (z. B. für den Software-Not-Halt-Reset aus der Steuerungsvisualisierung).

Der letzte Schritt der Software Implementation ist die Einrichtung des **lmc!400c** sowie der Wago PFC200 Steuerung für die Nutzung als OPC UA Server. Dabei sind die Schritte für beide Steuerungen identisch. Es unterscheiden sich lediglich die Daten, die über OPC kommuniziert werden können. Der **lmc!** soll vor allem Positionsdaten und für die Positionierung relevante Daten bereitstellen können, als auch lesen. Die Wago-Steuerung ist verantwortlich für die Messung von Verbrauchswerten des Systems, wie der benötigte Strom oder die Leistungsaufnahme. Diese Daten werden über den Wago OPC Server bereitgestellt. Für die Implementierung der OPC Funktionalität werden auf beiden Steuerungen folgende Schritte durchgeführt:

1. Es wird eine weitere globale Variabelliste angelegt, die alle Variablen enthält, die per OPC versendet bzw. empfangen werden sollen.
2. Im Hauptprogramm werden die globalen Variablen mit den Werten der relevanten Variablen im Programm gefüllt.
3. Im nächsten Schritt wird eine Symbolkonfiguration angelegt unter Beachtung der Auswahl der Checkbox für die OPC Kompatibilität.

4. Zunächst wird die Symbolkonfiguration einmal gebaut per Button in der oberen Leiste.
5. Anschließend wird die neu erstellte Variablenliste per Haken selektiert.
6. Nach erneutem Bauen und der Übertragung des Programmes ist die OPC Kommunikation betriebsbereit

Mit Fertigstellung der Software-Implementation der funktionalen Sicherheit ist die Implementierung sämtlicher Modelle aus der Projektierungs- bzw. Modellierungsphase abgeschlossen. Anschließend gilt es die Anforderungen an das System mithilfe der Testkriterien zu diesen zu überprüfen, um sicherzustellen, dass die nun implementierten Funktionalitäten die Anforderungen erfüllen.