Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Howard Düsing

Entwicklung eines Servoantriebes mit verteilten Komponenten

Fakultät Technik und Informatik

Faculty of Engineering and Computer Science

Howard Düsing

Entwicklung eines Servoantriebes mit verteilten Komponenten

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Mechatronik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing Florian Wenck
Zweiter Prüfer : Dr. Johann Zitzelsberger

Abgegeben am 28. Februar 2013

Howard Düsing

Thema der Bachelorthesis
Entwicklung eines Servoantriebes mit verteilten Komponenten

Stichworte
Servoantrieb, Regelung Synchronmaschine, dezentrale Regelung, EtherCAT, EtherCAT IP Core, EtherCAT Master, EtherCAT Slave

Kurzzusammenfassung
Die nachfolgende Bachelorthesis beschreibt die Entwicklung und Realisierung eines Servoantriebes mit verteilten Komponenten unter Anwendung eines modularen Denkansatzes zur Dezentralisierung der Mess- und Regeleinrichtung. Der Schwerpunkt wird hierbei auf die echtzeitfähige Datenübertragung der für die Regelung relevanten Informationen gelegt. Zudem werden die Anforderungen an die Datenübertragung erarbeitet und ein Lösungskonzept auf Basis von EtherCAT in die vorhandene Hardware implementiert. Der erstellte Versuchsaufbau wird abschließend in Betrieb genommen und evaluiert.

Howard Düsing

Title of the paper
Development of a servo drive with distributed components

Keywords
servo drive, synchronous machine control, decentralized control, EtherCAT, EtherCAT IP Core, EtherCAT Master, EtherCAT Slave

Abstract
The following bachelor thesis describes the development and implementation of a servo drive with distributed components using a modular approach to decentralize the measurement and control equipment. The focus here is put on the real-time data transmission of the relevant control information. Moreover, the requirements are developed on the data transfer and a solution on the basis of EtherCAT is implemented into the existing hardware. To complete the test setup, it´s finally put into operation and evaluated.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
AC	Alternating Current
ASIC	Application Specific Integrated Circuit
ASM	Asynchronmaschine
AXI	Advanced eXtensible Interface Bus
bspw.	beispielsweise
CEI	Customize Exchange Interface
DC	Direct Current
DC	Distributed Clocks
DDR3-RAM	Double Data Rate – Random Access Memory Type 3
DO	Digital Output
DPRAM	Dual Port Random Access Memory
ENI	EtherCAT Network Information File
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information File
ETG	EtherCAT Technology Group
FMMU	Fieldbus Memory Management Unit
FOR	Feldorientierte Regelung
FPGA	Field Programmable Gate Array
ggf.	gegebenenfalls

GNM	Gleichstrom-Nebenschlussmaschine
GSM	Gleichstrommaschine
IGBT	Insulated-Gate Bipolar Transistor
IP Core	Intellectual Proberty Core
PDI	Process Data Interface
PLB	Processor Local Bus
PM	Permanent Magnet
PWM	Pulsweitenmodulation
RTE	Real-time Ethernet
SDB	Smart Driver Board
SM	Synchronmaschine
SM	SyncManager
SoC	System-on-a-Chip
sog.	sogenannt
SSB	Smart Sensor Board
TG	Tachogenerator
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XML	Extensible Markup Language
z.B.	zum Beispiel

Symbolverzeichnis

Symbol	Bezeichnung	Einheit
f	Frequenz	Hz
f_{Schalt}	Schaltfrequenz	Hz
i_A	Ankerkreis Strom	A
i_a	Strom Phase A	A
i_b	Strom Phase B	A
i_c	Strom Phase c	A
$i_{d\ Ist}$	q-Komponente Ist-Strom	A
$i_{d\ Soll}$	d-Komponente Soll-Strom	A
i_{Ist}	Ist-Strom	A
I_N	Nennstrom	A
$i_{q\ Ist}$	q-Komponente Ist-Strom	A
$i_{q\ Soll}$	q-Komponente Soll-Strom	A
i_{Soll}	Soll-Strom	A
K_M	Motorkonstante	o.E
L_A	Ankerkreis Induktivität	H
m/m_M	Drehmoment Motor	Nm
M_N	Nenndrehmoment	Nm
m_{Soll}	Soll-Drehmoment	Nm
n_M	Drehzahl Motor	1/s
n_N	Nenndrehzahl	1/s
P_N	Nennleistung	W
R_A	Ankerkreis Widerstand	Ω
t_{ADC1}	Konvertierungszeit 1	µs
t_{ADC2}	Konvertierungszeit 2	µs
t_{buffer}	Zeitpuffer	µs
t_{calc}	Berechnungszeit	µs
t_{cycle}	Zykluszeit	µs
$t_{data\ delay}$	Zeit für Datenverzug	µs
$t_{DB\ copy}$	Zeit für Driver-Board Umkopiervorgang	µs
$t_{Jitter\ max}$	Zeit des maximalen Jitter	µs
$t_{pre\text{-}trig}$	Vorhaltezeit	µs
$t_{SB\ copy}$	Zeit für Sensor-Board Umkopiervorgang	µs
$t_{travel\ time}$	Datenlaufzeit	µs
u	Spannung	V

Symbol	Bezeichnung	Einheit
u_A	Ankerkreis Spannung	V
u_a	Spannung Phase A	V
u_b	Spannung Phase B	V
u_c	Spannung Phase C	V
u_d	d-Komponente Spannung	V
u_i	Spannung Stromsensor	V
$u_{i\ Ist}$	Spannung Stromsensor Ist-Strom	V
U_N	Nennspannung	V
u_q	q-Komponente Spannung	V
u_q	Induzierte Spannung	V
U_R	Spannung Resolver Rotor	V
U_{S1}	Spannung Resolver Spule 1	V
U_{S2}	Spannung Resolver Spule 2	V
u_{St}	Strangspannung	V
u_{ZK}	Zwischenkreisspannung	V
u_ω	Spannung Drehzahlsensor	V
$u_{\omega\ Ist}$	Spannung Drehzahlsensor Ist-Drehzahl	V
$u_{\omega\ Soll}$	Spannung Drehzahlsensor Soll-Drehzahl	V
v	Meldegröße	o.E.
w	Führungsgröße	o.E.
x	Ist-Wert	o.E.
x_a	Regelgröße	o.E.
z	Störgröße	o.E.
α	Winkel	°
η	Wirkungsgrad	o.E.
$\vartheta_{s\ Ist}$	Rotorlage Ist-Wert	°
φ	Rotorlage	°
Φ_M	Magnetischer Fluss Motor	Wb
ω/ω_M	Winkelgeschwindigkeit Motor	1/s

1. Einleitung

In hoch entwickelten Industrieländern werden gegenwärtig etwa 60 % der erzeugten elektrischen Energie mittels elektrischer Antriebe in unterschiedliche mechanische Energieformen gewandelt[1]. Dieser Prozess ist allgegenwärtig und beschränkt sich schon lange nicht mehr auf den industriellen Einsatz. Nahezu alle Haushaltsgeräte nutzen elektrische Energie in Kombination mit der Antriebstechnik zur Bewältigung unterschiedlichster Aufgaben. Zudem erlangen elektrische Antriebe in der Personen- und Güterbeförderung eine stetig wachsende Bedeutung[2]. Die elektrische Servoantriebstechnik bildet hierbei immer häufiger die Schnittstelle zwischen den elektronischen Steuerungen und den mechanischen Komponenten eines Antriebssystems. Sie löst die auf Mechanik basierenden Bewegungsabläufe ab und ist eine Art Triebfeder zur Bewegungssteuerung automatisierter Prozesse[3]. Nicht zuletzt wird dieser Vorgang durch die sich ständig erhöhende Leistungsfähigkeit von Mikroelektronik, Sensorik und Leistungselektronik getragen.

In Zusammenarbeit mit der ESW GmbH am Standort Wedel ist im Rahmen dieser Ausarbeitung das bestehende Konzept der In-House gefertigten Servoantriebe überarbeitet worden. Diese Erweiterung ist nötig, um die Technologie des Servoantriebes für zukünftige Produkte noch flexibler zu gestalten und sich durch Innovation auch weiterhin vom Markt abzuheben. Die ESW GmbH ist ein Tochterunternehmen der Jenoptik AG und Anbieter technologisch komplexer, innovativer Produkte und Leistungen in der Zivil- und Verteidigungstechnik. Die Kernkompetenzen der ESW liegen in den Bereichen der Fahrzeug- und Flugzeugausrüstung, Antriebs- und Stabilisierungstechnik sowie Energiesystemen.

Die nachfolgende Bachelor Thesis beschreibt zunächst die Servoantriebstechnik mit den dazugehörigen Regelungskonzepten und den unterschiedlichen Bauarten der Antriebsmotoren. Anschließend wird das Konzept eines Servoantriebes mit verteilten Komponenten erläutert und die sich hieraus ableitenden Konsequenzen erarbeitet.

[1] Vgl. (Schulze, 2008, S. 11).

[2] Anstieg des Pkw Bestands mit alternativen Antrieben um 6,7% im Vergleich zum Vorjahr (Stand 01.01.2012) | Quelle: (Kraftfahrt-Bundesamt, 2013).

[3] Vgl. (Griesenbruch, 2013).

Anhand der Rahmenbedingungen seitens der ESW GmbH sowie weiterer technischer Anforderungen wird ein Lösungskonzept erarbeitet und in Form eines Demonstrators, sowohl mit industriellen Kaufkomponenten als auch mit internen proprietären Baugruppen praktisch erprobt. Der Schwerpunkt liegt hierbei auf der echtzeitfähigen Datenübertragung der verteilten Komponenten. Abschließend wird das Projekt anhand der zuvor erarbeiteten Rahmenbedingungen evaluiert und getestet. Darüber hinaus wird ein kurzer Überblick über zukünftige Modifikationen und Erweiterungen gegeben.

2. Servoantriebe

2.1. Grundstruktur des Servoantriebes

Auf Grund der Vielzahl unterschiedlicher Ausführungsformen im Bereich der Servoantriebe ist es sinnvoll, zunächst die allgemeingültige Grundstruktur des Antriebes zu beschreiben. Diese Grundstruktur eines Servoantriebes lässt sich in die aus der Abbildung 2.1 ersichtlichen Baugruppen unterteilen. Zu ihr gehören nicht nur die Regelstrecke, sondern auch die Netzeinspeisung sowie die Mess- und Regeleinrichtung. Der Leistungsfluss von der Energiequelle bis zur Antriebsmaschine und zurück erfolgt über die Teilelemente der Regelstrecke. Die elektrische Energie ist in der Abbildung 2.1 dunkel blau eingefärbt.

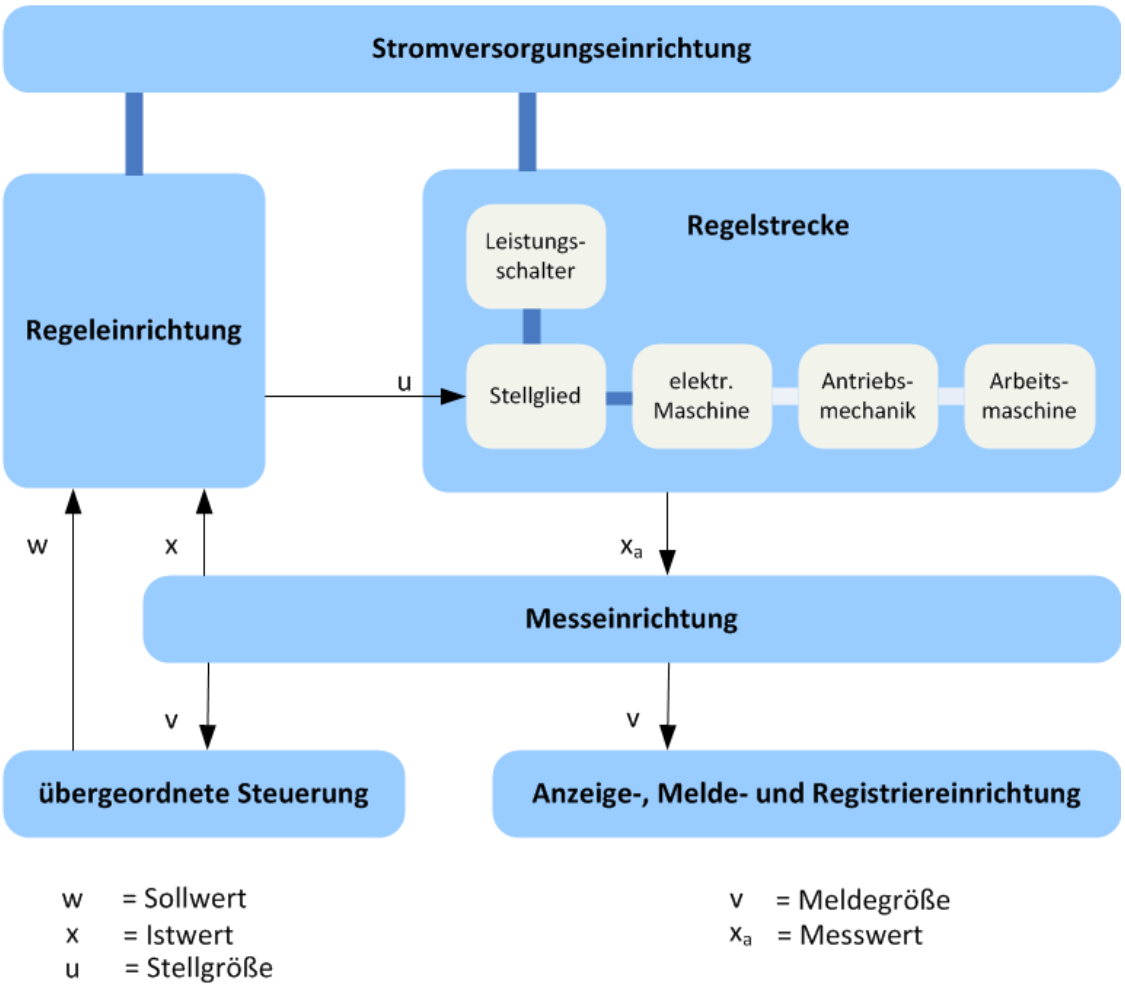


Abbildung 2.1: Grundstruktur eines Servoantriebes | Quelle: Eigene Darstellung in Anlehnung an (Schulze, 2008, S. 13)

Stromversorgungseinrichtung

Die Stromversorgungseinrichtung dient dem Bereitstellen von elektrischer Energie, sowie ggf. zur Aufnahme dieser, sofern der Antrieb auch als Generator Verwendung findet. Typisch hierfür sind sowohl Strom- als auch Spannungsquellen mit variabel einstellbaren Strömen und Spannungen.

Messeinrichtung

Bei der Messeinrichtung handelt es sich um eine Vorrichtung zur Erfassung der, zur Steuerung des technischen Prozesses benötigten, Zustandsinformationen. Bei einem Servoantrieb sind dies in der Regel Messeinrichtungen zum Ermitteln der Motorströme und –spannungen sowie eine Messeinrichtung zur Detektion des Weges bzw. Winkels des Rotors.

Übergeordnete Steuerung

Die Aufgabe der Sollwert-Vorgabe (auch Führungsgröße) wird durch die übergeordnete Steuerung wahrgenommen. Hierbei handelt es sich um den angestrebten Ist-Wert des Systems, der in Abhängigkeit der verwendeten Steuerung in Form von Weg, Geschwindigkeit oder Drehmoment vorgegeben werden kann. Die Vorgabe des Sollwertes erfolgt entweder manuell oder automatisiert, wobei die übergeordnete Steuerung sowohl als PC aber auch als Embedded-Lösung umgesetzt werden kann.

Anzeige-, Melde- und Registriereinrichtung

Die Anzeige-, Melde- und Registriereinrichtung dient zur Anzeige von Meldegrößen aus der Messeinrichtung. Ferner wird sie als Schnittstelle zur Einbindung des Einzelantriebes in eine Einzel- oder Mehr-Komponenten-Prozesssteuerung verstanden.

Regelstrecke

Die Leistungsschalter und das Stellglied sowie die elektrische Maschine (elektromechanischer Wandler) inklusive der Antriebsmechanik und ggf. einer Arbeitsmaschi-

ne bilden die Grundelemente der Regelstrecke[4]. Die Regelstrecke beinhaltet somit die zu manipulierende physikalische Größe und wird oft als mathematisches Modell eines dynamischen Systems mit einer Verkettung von Einzelsystemen abgebildet.

Regeleinrichtung

Bei der Regeleinrichtung handelt es sich um jenes Modul, welches die, durch den elektro-mechanischen Wandler bereit gestellte mechanische Energie, effektiv beeinflusst. Hierfür vergleicht die Regeleinrichtung den Istwert mit der Führungsgröße und manipuliert diese anschließend gezielt. Die Module Regelstrecke, Messeinrichtung und Regeleinrichtung bilden hierbei einen Regelkreis. Typische Umsetzungen von Regelkreisen gibt es für die Lage (Weg), Drehzahl (Geschwindigkeit) und das Drehmoment (Strom).

2.2. Regelungstechnische Struktur eines Servoantriebes

Die komplette Regelung eines Servoantriebes fasst die Regeleinrichtung, die Regelstrecke und die Messeinrichtung in einem geschlossenen Regelkreis zusammen. Daraus ergibt sich eine Grundstruktur, die in Form eines allgemein gültigen Blockschaltbilds in der Abbildung 2.2 dargestellt ist.

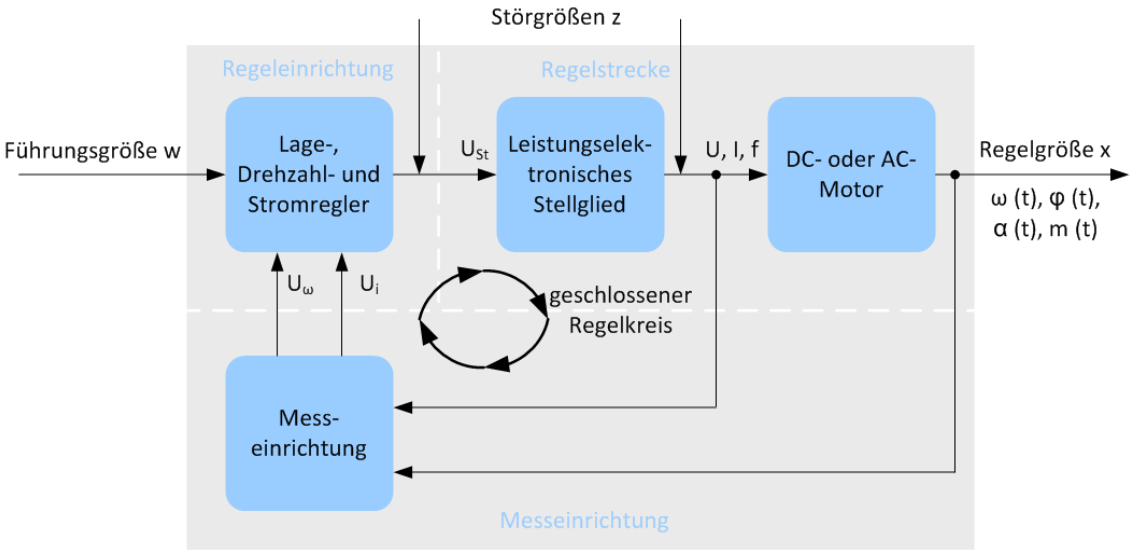


Abbildung 2.2: Regelungstechnische Struktur eines Servoantriebes | Quelle: Eigene Darstellung in Anlehnung an (Schulze, 2008, S. 16)

[4] Vgl. (Schulze, 2008, S. 16).

Anhand des Blockschaltbildes wird deutlich, dass die einzelnen Komponenten einen geschlossenen Kreislauf, den sog. Regelkreis, ergeben. Dieser geschlossene Kreislauf ist eine Grundvoraussetzung, um ein System als Regelung zu deklarieren. Besitz ein vergleichbares System keine Rückführung der gemessenen Istwerte, so wird dieses als Steuerung bezeichnet. Aus diesem Sachverhalt lässt sich der grundlegende Unterschied beider Systeme ableiten. Bei einer Steuerung wird lediglich ein Sollwert in Form der Führungsgröße auf die Regelstrecke gegeben. In Abhängigkeit physikalischer Zusammenhänge reagiert das Gesamtsystem und es stellt sich eine berechenbare Ausgangsgröße ein. Äußere Einflüsse, die sog. Störgrößen, beeinflussen die Regelstrecke jedoch mit einem zeitlich unvorhersehbaren Verlauf, sodass das vorhergesagte Verhalten der Ausgangsgröße nicht mehr dem tatsächlichen Verhalten entspricht.
Bei einer Regelung hingegen wird regelmäßig die Ausgangsgröße (Regelgröße x) messtechnisch erfasst und an die Regeleinrichtung zurückgeführt. Hierbei ermittelt die Messeinrichtung auch den Einfluss von Störgrößen auf das System, sodass in der Regeleinrichtung ein Abgleich von gewünschter Führungsgröße und messtechnisch erfasster Istgröße stattfinden kann. Der vorgegebene Sollwert kann hierdurch nicht nur angefahren, sondern auch über einen beliebigen Zeitraum nahezu konstant gehalten werden[5].

2.3. Bauarten von Servoantrieben

2.3.1. Gleichstromservoantriebe

Die Technologie der Servoantriebe lässt sich grundsätzlich in die Gruppen der Gleichstromservoantriebe und der Drehstromservoantriebe unterteilen. Bei den Gleichstromservoantrieben werden überwiegend permanenterregte Gleichstrommotoren (GSM oder GM)[6] oder Gleichstromnebenschlussmotoren (GNM)[7] als elektromechanische Wandler verwendet. Gespeist werden diese über einen Stromrichter, der wiederum seine elektrische Energie aus einem ein- bzw. dreiphasigen Eingangsnetz bezieht. Vervollständigt wird diese Regelstrecke durch die Stromrichterelektronik, die anhand ihrer Eingangsgröße U_{ST} (Steuerspannung) die benötigten Zündimpulse des netzgeführ-

[5] Vgl. (Schröder, 2009, S. 2).
[6] Vgl. (Schulze, 2008, S. 51).
[7] Vgl. (Schröder, 2009, S. 223).

ten Thyristorstromumrichters bzw. die PWM-Ansteuersignale für den selbstgeführten Pulssteller generiert[8]. Die Regeleinrichtung für Strom und Drehzahl wird meist zusammen mit dem leistungselektronischen Stellglied implementiert. Als Messeinrichtung werden Drehzahl-, Strom- und/oder Spannungssensoren verwendet, die meist einen zu ihrem Messwert proportionalen Spannungswert (z.B. Ankerstrom $U_{i\ Ist}$, Drehzahl $U_{\omega\ Ist}$) an die Regeleinrichtung zurückführen. Abbildung 2.3 fügt die vorgestellten Teilsysteme zur Struktur des Gleichstromservoantriebes zusammen.

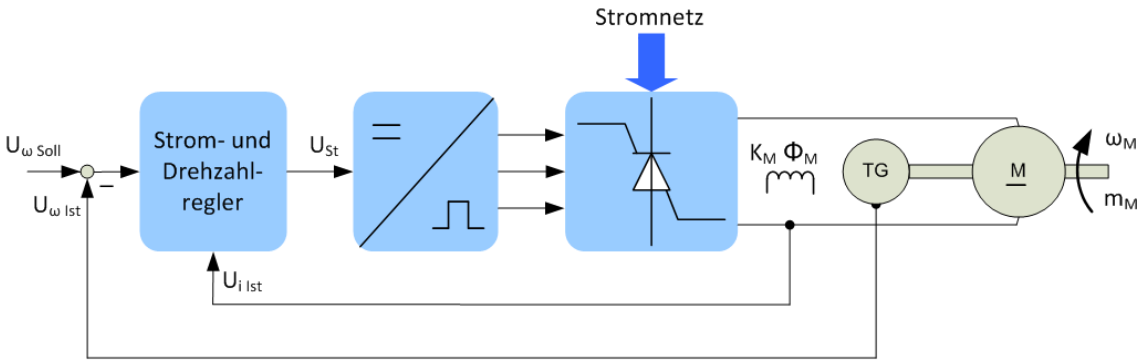


Abbildung 2.3: Aufbau eines Gleichstromservoantriebes | Quelle: Eigene Darstellung in Anlehnung an (Schulze, 2008, S. 51)

Um das grundlegende Regelungskonzept der GSM zu erläutern, ist es zunächst sinnvoll, den Strom- und Drehzahlregler sowie das leistungselektronische Stellglied als Black Box zu betrachten und die GSM durch das in Abbildung 2.4 eingeführte Ersatzschaltbild zu ersetzen.

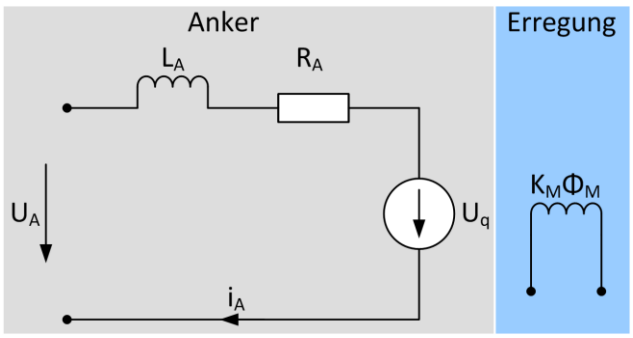


Abbildung 2.4: Elektrisches Ersatzschaltbild der GSM | Quelle: Eigene Darstellung in Anlehnung an (Leonhard, 2000, S. 53)

Ferner wird angenommen, dass diese Black Box mit Hilfe der Eingangsgrößen U_{Soll} (Führungsgröße), $U_{\omega\ Ist}$ (Drehzahl-Regelgröße) und $U_{I\ Ist}$ (Strom-Regelgröße) die Aus-

[8] Vgl. (Schulze, 2008, S. 51).

gangsgrößen U_A (Ankerspannung) und i_A (Ankerstrom) zur Verfügung stellt. In Kombination mit dem in Abbildung 2.4 eingeführten elektrischen Ersatzschaltbild der GSM, können anschließend folgende physikalische Zusammenhänge zur gezielten Beeinflussung von Drehzahl und Drehmoment abgeleitet werden:

$$U_A = i_A \cdot R_A + L_A \cdot \frac{di_A}{dt} + U_q \rightarrow Kirchhoff´sche\ Maschenregel \tag{2.1}$$

Wird zudem von einem nahezu konstanten Ankerstrom ausgegangen, so folgt:

$$L_A \cdot \frac{di_A}{dt} = 0 \tag{2.2}$$

Bei ausreichend kleinem Ankerwiderstand R_A bzw. bei ausreichend großer Ankerspannung U_A wird zudem der Einfluss des Terms $i_A \cdot R_A$ auf das Gleichungssystem sehr gering, sodass sich vereinfacht $U_A = U_q$ ergibt.
Unter Berücksichtigung des Induktionsgesetzes und der oben genannten Vereinfachungen folgt:

$$U_A = U_q = K_M \cdot \Phi_M \cdot \omega_M = K_M \cdot \Phi_M \cdot 2 \cdot \pi \cdot n_M \tag{2.3}$$

Durch die Beziehung vom magnetischem Fluss zum Motordrehmoment $\Phi_M \sim M_M$ ergibt sich bei konstantem Drehmoment ein proportionaler Zusammenhang zwischen der aus der Black Box stammenden Ankerspannung U_A und der Drehzahl der GSM n_M.
Der zuvor erwähnte Zusammenhang von Drehmoment M_M und magnetischem Fluss Φ_M stützt sich auf die folgende physikalische Grundlage:

$$M_M = K_M \cdot \Phi_M \cdot i_A \tag{2.4}$$

Hierdurch erklärt sich auch die regelungstechnische Abhängigkeit von Drehmoment M_M und Ankerstrom i_A, sofern von einem konstanten magnetischen Fluss ausgegangen wird. Somit gilt vereinfacht $M_M \sim i_A$[9].
Die eigentliche Aufgabe der Regelung besteht folglich darin, durch einen fortwährenden Abgleich der Führungsgröße mit den Regelgrößen den angestrebten Maschinenzustand auf Dauer aufrecht zu erhalten. Je nach einwirkender Störung kann die Regelung auf ein verändertes Drehmoment oder einer abweichenden Drehzahl mit einem ange-

[9] Vgl. (Giersch, Harthus, & Vogelsang, 2003, S. 127 u. ff.).

passten Strom i_A bzw. einer korrigierten Spannung U_A reagieren und infolgedessen den Ist-Zustand nachregeln.

Anhand der Abbildung 2.3 wird zudem deutlich, dass es sich bei der Regelung der GSM genau genommen um zwei verschiedene Regelkreise handelt. Der innere Regelkreis des Stromreglers ist dem äußeren Spannungs- (Drehzahl-) regelkreis unterlagert. Diese Art der Regelung wird auch als kaskadiert, bzw. als Kaskadenregelung bezeichnet[10].

2.3.2. Drehstromservoantriebe

Bei der zweiten Gruppe der Servoantriebe handelt es sich um die sog. Drehstromservoantriebe, die sowohl Synchronmotoren (SM), als auch Asynchronmotoren (ASM) als elektromechanische Wandler verwenden. Unabhängig von der Ausführung stützen sich die Regelungskonzepte häufig auf das Prinzip der feldorientierten Regelung (FOR). Hierbei wird eine Komponente des Ständerstroms mit Hilfe mathematischer Transformationen und der Überführung von Fluss, Strom und Spannung in die sog. Raumvektordarstellung direkt proportional zum Drehmoment[11]. Dem mathematischen Aufwand wird folglich Rechnung getragen, indem im Anschluss ein der GSM ähnliches Regelungskonzept Anwendung finden kann. Die Abbildung 2.5 zeigt die Grundstruktur eines solchen Drehstromservoantriebes auf Grundlage einer ASM.

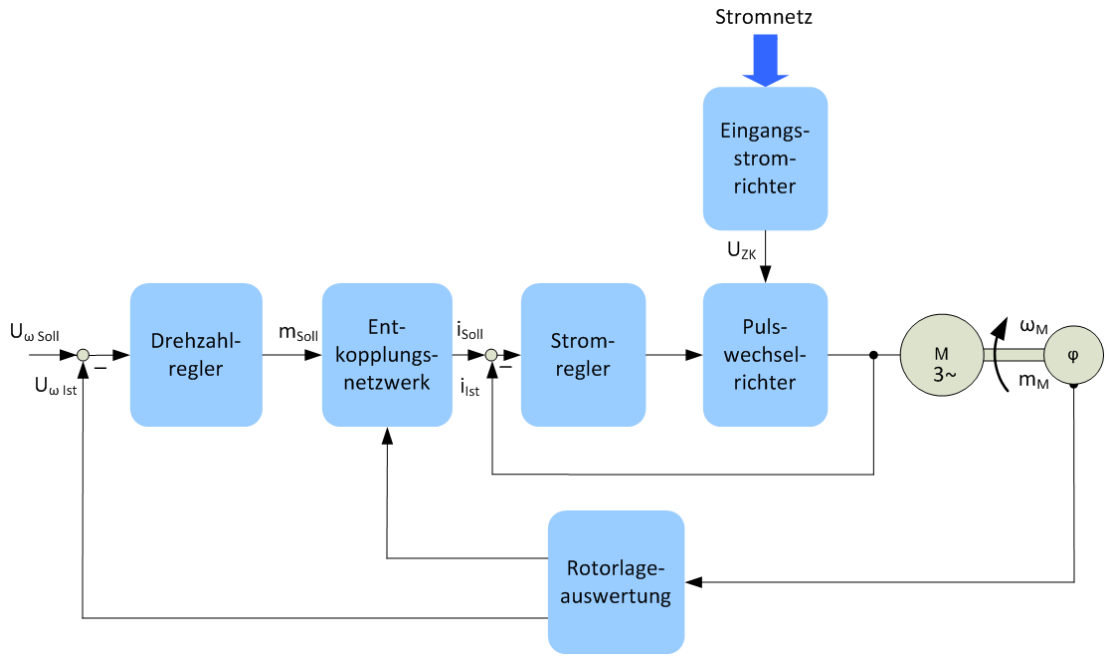


Abbildung 2.5: Aufbau eines Drehstromservoantriebes | Quelle: Eigene Darstellung in Anlehnung an (Schulze, 2008, S. 88)

[10] Vgl. (Schulze, 2008, S. 17).
[11] Vgl. (Schulze, 2008, S. 63).

Der innere Stromregelkreis beherbergt sowohl den eigentlichen Stromregler, als auch den Pulswechselrichter. Dieser sog. indirekte Umrichter mit Gleichspannungszwischenkreis U_{ZK} moduliert anhand seiner Eingangsgröße (= Ausgangsgröße Stromregler) die Ständerspannungen zur Einprägung der sinusförmigen Ständerströme in die Drehstrommaschine.

Dem Stromregelkreis überlagert findet wiederum der Drehzahlregler Anwendung. Im Vergleich zur GSM wird lediglich ein Entkopplungsnetzwerk eingefügt, welches dem stark nichtlinearen Klemmen- und Momentenübertragungsverhalten der Drehstrommaschinen entgegenwirkt[12].

2.4. Komponenten des verwendeten Servoantriebes

2.4.1. Netzteil

Die Auswahl der verwendeten Bauteile wurde im Vorfeld dieser Ausarbeitung seitens der Projektplanung getroffen, sodass diesbezüglich keine ingenieurstechnische Entscheidungsfindung angewendet wird. Zudem kann bei den Komponenten des Servoantriebes an dieser Stelle nur eine kurze und wenig detailreiche Beschreibung vorgenommen werden. Sowohl der Motor, als auch der Umrichter sind Bestandteile unterschiedlichster Einheiten militärischer Zielapplikationen und unterliegen folglich strengsten Geheimhaltungsanforderungen.

Beim Schaltnetzteil hingegen handelt es sich um ein typisches Labornetzgerät der Serie EA-PS 8000 welches in der abb abgebildet ist. Dieses ist mikroprozessorgesteuert und unterstützt die Funktion des Auto-Ranging, wobei ein hoher Ausgangsstrom oder eine hohe Ausgangsspannung bei konstanter Ausgangsleistung zur Verfügung gestellt wird. Tabelle 2.1 gibt einen Überblick über die technischen Daten.

Bei der Auswahl des Netzgerätes galt es lediglich, die Rahmenbedingungen zur Energieversorgung des verwendeten Umrichters einzuhalten. Hierfür muss das Netzgerät einen Ausgangsspannungsbereich von 0 bis 220 V DC abdecken. Der Ausgangsstrom ist hierbei von untergeordnetem Interesse, da der geplante Versuchsaufbau nur im Leerlauf und nicht unter Last betrieben wird. Um eine erhöhte Stromaufnahme seitens des

[12] Vgl. (Schulze, 2008, S. 88).

Umrichters beim Beschleunigen des Motors zu vermeiden, wird zudem bei der Umsetzung ein Anfahren der Soll-Drehzahl durch eine Rampen-Funktion implementiert.

Abbildung 2.6: Netzteil EA-PS 8360-15 T |Quelle: (Elektro-Automatik, S. 1)

Tabelle 2.1: Technische Daten Labornetzteil | Quelle: (Elektro-Automatik, S. 15)

Typ:	EA-PS 8360-15 T
Abmessung:	90 x 240 x 400 mm (L x B x H)
Anzahl Ausgänge:	1
Ausgangsstrom:	0-15 A
Ausgangsspannung:	0-360 V/DC
Leistung:	1500 W
Anschlüsse:	Lastklemmen
Eingangsspannung:	90-264 V/AC

Eine weitere Einschränkung des Netzteils liegt aufgrund der fehlenden Fähigkeit zur Rückeinspeisung elektrischer Energie vor. Geht der verwendete Motor in den generatorischen Betrieb, so versucht der Umrichter die generierte elektrische Energie ins Netz zurückzuspeisen. Auch auf diesen Punkt muss folglich bei der Umsetzung Rücksicht genommen werden.

2.4.2. Motor

Bei dem verwendeten Motor handelt es sich um einen Drehstromsynchronmotor in Schlankläufer-Ausführung mit Rechteckständer, dessen prinzipieller Aufbau der Abbildung 2.7 entnommen werden kann. Der Läufer ist hierbei mithilfe von Magneten permanenterregt. Die Drehstromwicklungen hingegen sind in Nuten um den Läufer herum

angeordnet. Die technischen Daten des Motors sind in Tabelle 2.2 zusammengefasst. Im Motor integriert befindet sich der Rotorpositionsgeber in Form eines Resolvers.

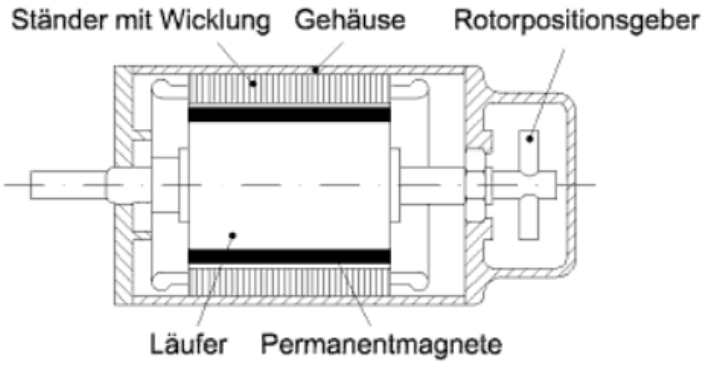


Abbildung 2.7: Aufbau Schlankläufer-Synchronmotor mit Rechteckständer | Quelle: (Schulze, 2008, S. 69)

Tabelle 2.2: Technische Daten Motor | Quelle: (ESW GmbH)

Typ	Synchronmotor permanenterregt
Nennspannung U_N	67 V_{EFF}
Nennstrom I_N	40 A_{EFF}
Nennleistung P_N	8,2 kW
Nennmoment M_N	23 Nm
Nenndrehzahl n_N	3400 1/min

Der Resolver ermittelt für die Messeinrichtung die absolute Lage des Rotors mit einer Genauigkeit von 0,1°. Hierfür werden an die, anhand der Abbildung 2.8 ersichtlichen, Statorspulen je eine Sinus- und eine Cosinus-Spannung (U_{S1} und U_{S2}) angelegt. Die in dem Rotor induzierte Spannung U_R wird induktiv von einer Elektronik erfasst und an die Messeinrichtung übermittelt. Liegt bspw. an der Ständerspule U_{S2} die Sinusspannung an, ermittelt die Elektronik für die Rotorposition φ = 0° keine Phasendifferenz. Dreht sich der Rotor anschließend im Uhrzeigersinn weiter, vergrößert sich die Phasenverschiebung der im Rotor induzierten Spannung U_R. Erreicht der Rotor die Position gegenüber der Ständerspule U_{S1}, beträgt die Phasenverschiebung 90° (Cosinus.). Für alle weiteren Zwischenpositionen stellen sich Phasenlagen von 0 bis 360° ein[13].

[13] Vgl. (Roddeck, 2012, S. 186)

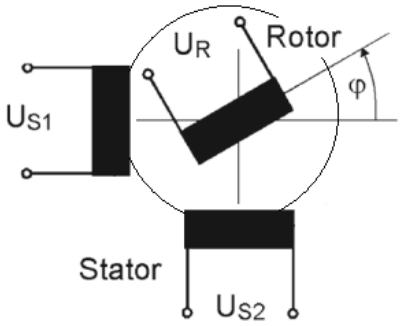


Abbildung 2.8: Aufbau Resolver | Quelle: Eigene Darstellung in Anlehnung an (Roddeck, 2012, S. 186)

2.4.3. Umrichter

Als leistungselektronisches Stellglied steht für den Testaufbau ein sog. Pulswechselrichter zur Verfügung. Anhand der Eingangsgröße U_{ZK} (Zwischenkreisspannung) und dem hochfrequenten Anschalten der, zur B6-Brücke verschalteten, IGBTs wird am Ausgang eine sinusförmige Spannung moduliert. Diese dient wiederum zum Einprägen der sinusförmigen Ständerströme in die am Umrichter angeschlossene Synchronmaschine. Abbildung 2.9 zeigt den prinzipiellen Aufbau eines zwischenkreisgespeisten Pulswechselrichters.

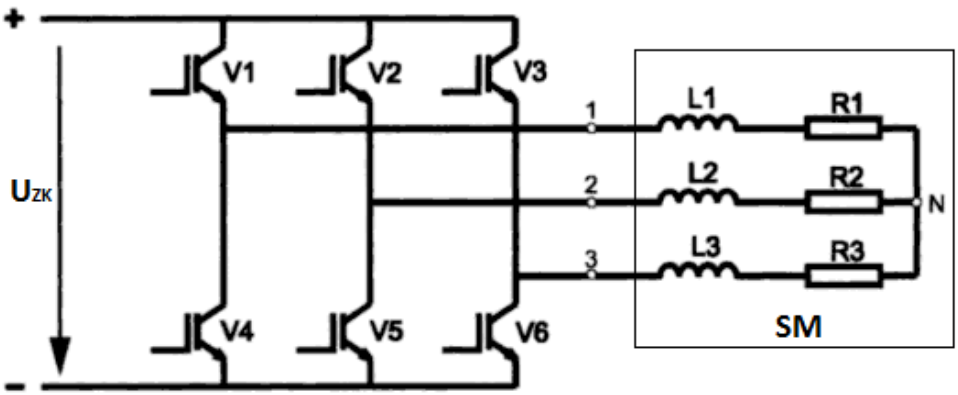


Abbildung 2.9: Zwischenkreisgespeister Pulswechselrichter | Quelle: (Fuest & Döring, 2007, S. 180)

Durch das Verschalten der Ventile V1 bis V6 entstehen am Ausgang des Umrichters sieben unterschiedliche Spannungsniveaus ($-U_{ZK}$, $-2/3\ U_{ZK}$, $-1/3\ U_{ZK}$, 0, $1/3\ U_{ZK}$, $2/3\ U_{ZK}$ und U_{ZK}). Um diese treppenförmige Ausgangsspannung einem sinusförmigen Spannungsverlauf anzunähern, wird beim Umrichter der ESW GmbH eine weitere B6-Brücke mit der halben Zwischenkreisspannung in die Schaltung integriert. Durch diese sog. 3-Level-Umrichter Technologie entstehen folglich sechs weitere Spannungspegel.

In Kombination mit einem Ausgangsfilter, der die noch immer treppenförmige Sinusspannung ein weiteres Mal glättet, entsteht eine sinusförmige Ausgangsspannung von $80V_{Eff}$. Die Tabelle 2.3 fasst die wichtigsten technischen Daten des Umrichters zusammen.

Tabelle 2.3: Technische Daten Umrichter | Quelle: (ESW GmbH)

Typ	Pulswechselrichter
Nennspannung Eingang $U_{N\,in}$	220 V_{DC}
Nennstrom Eingang $I_{N\,in}$	160 A
Nennspannung Ausgang $U_{N\,out}$	80 V_{Eff}
Nennstrom Ausgang $I_{N\,out}$	180 A_{Eff}
Schaltfrequenz f_{Schalt}	30 kHz
Wirkungsgrad η	0,95

2.4.4. Mess- und Regeleinrichtung

Die Mess- und Regeleinrichtung des verwendeten Antriebes ist fester, integraler Bestandteil des beschriebenen Umrichters. Sie besteht aus einem Kartenstapel, der sich aus drei unterschiedlichen Boards zusammensetzt. Das in der Abbildung 2.10 schematisch dargestellte Board an oberster Position spiegelt die eigentliche Messeinrichtung wieder. Die AD-Wandler des Boards konvertieren die Messwerte des Resolvers, sowie der Strom und Spannungssensoren in digitale Werte, bevor ein FPGA[14] die Informationen mittels einer Steckverbindung zum Regler-Board weiterleitet. Der FPGA des Regler-Boards berechnet die neue Stellgröße und schaltet mithilfe des unteren Boards (CEI – Customize Exchange Interface) die Ventile der Leistungselektronik.

[14] FPGA: Field Programmable Gate Array ist ein integrierter Schaltkreis aus der Digitaltechnik, der frei programmierbare Bereiche zum Implementieren von logischen Schaltungen besitzt. Vgl. (Biere, Kroening, Weissenbacher, & Wintersteiger, 2008, S. 73).

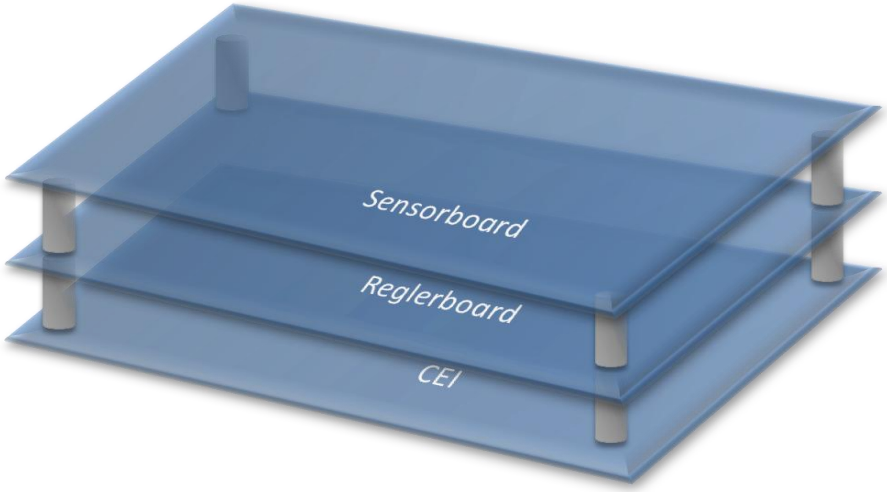


Abbildung 2.10: Prinzipskizze des Kartenstapels der im Umrichter integrierten Mess- und Regeleinrichtung | Quelle: Eigene Darstellung

Die Mess- und Regeleinrichtung bildet folglich, zusammen mit dem Umrichter, ein geschlossenes System. Anhand der Abbildung 2.11 wird deutlich, dass die einzelnen Teilsysteme des Regelkreises zu einem intelligenten Gesamtsystem verschmolzen sind. Die einzelnen Systemgrenzen sind somit kaum noch nachvollziehbar.

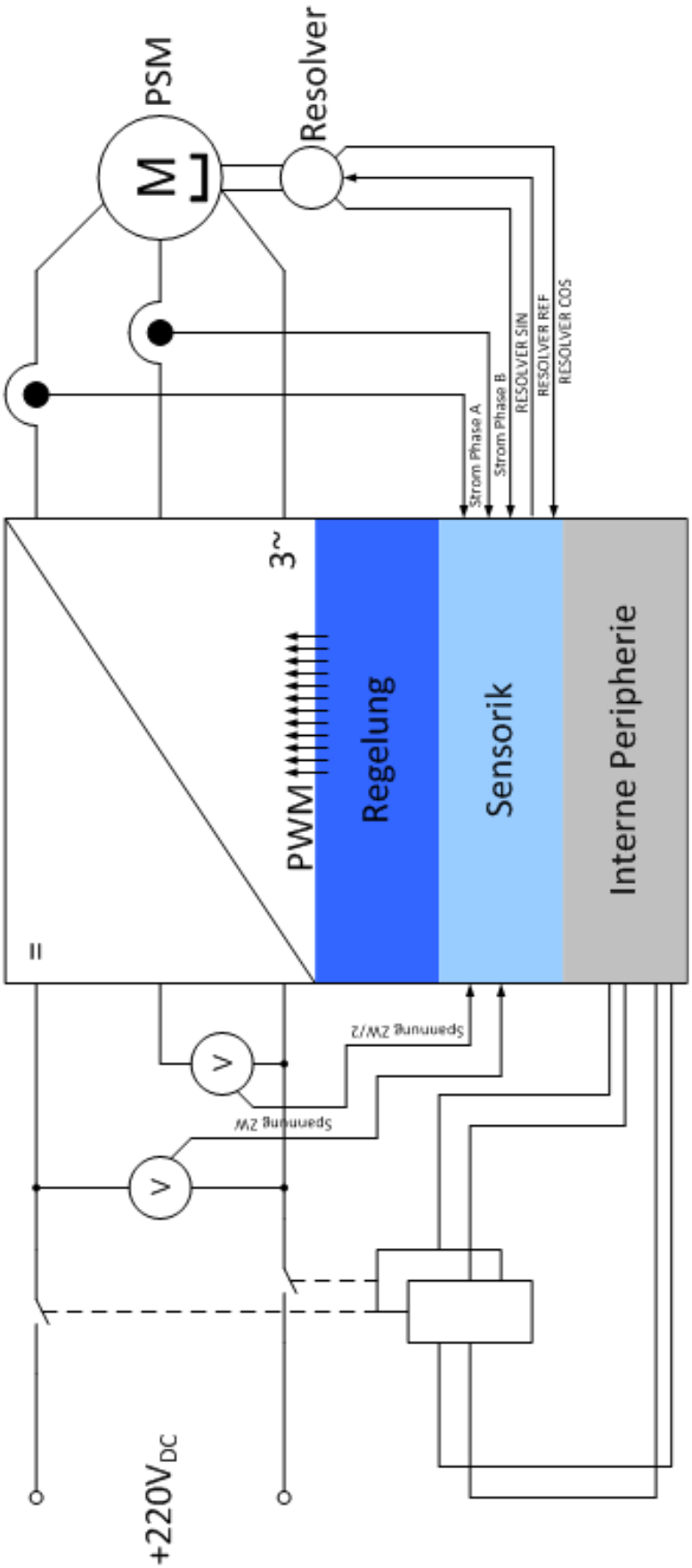


Abbildung 2.11: Ist-Situation Servoantriebstechnik der ESW GmbH | Quelle: Eigene Darstellung

3. Konzeption eines Servoantriebes mit verteilten Komponenten

3.1. Aufbau des Antriebssystems

Anhand der zuvor beschriebenen Ausgangssituation wird an dieser Stelle das bestehende Konzept des Servoantriebes aufgebrochen und in einzelne Teilsysteme zerlegt. Diese Form der Modularisierung führt zu zweierlei Vorteilen. Zum ersten wird durch einen derartigen Aufbruch das Komplettsystem des Servoantriebes, wieder der in Kapitel 2 erörterten Grundstruktur angenähert. Die über lange Zeit entstandenen Unschärfen und Überschneidungen von Teilsystemgrenzen verschwinden, wodurch das Gesamtsystem nachvollziehbarer und einzelne Komponenten leichter austauschbar werden. Hierdurch können bspw. bei Produktaktualisierungen, aber auch bei Neuentwicklungen einzelne Module kostengünstig ausgetauscht oder erneuert werden. Der zweite Vorteil liegt im Vorhaben der Realisierung einer industriellen Standardschnittstelle auf Basis von Echtzeit-Ethernet, welche die Kommunikationsgrundlage der einzelnen Module bildet. Hierdurch kann der aufwändige In-House Standard abgelöst werden, wodurch die Möglichkeit besteht, sowohl produktinterne Schnittstellen, als auch die Schnittstellen zum Kunden über einen gemeinsamen Standard zu bedienen.

Der aus diesem Konzept entstehende Testaufbau soll zunächst mit Hilfe eines Standard-PCs (Master) gesteuert werden. Dieser nimmt die Funktion der übergeordneten Steuerung wahr und stellt die Führungsgröße zur Verfügung. Zusätzlich können an dieser Stelle sämtliche Messdaten eingesehen werden.

Der gesamte Regelungsprozess wird hingegen auf einer separaten Karte (SDB – Smart Driver Board) bearbeitet. Diese bezieht die Regler-Parameter sowie die Messdaten in Echtzeit über das Netzwerk. Gleichzeitig dient das SDB als Schnittstelle zur Leistungselektronik, indem die vom Regler modulierten PWM-Signale via Flachbandleitung an den Umrichter übermittelt werden.

Für die Zustandsüberwachung werden sämtliche Messdaten zyklisch auf dem Sensor-Board (SSB – Smart Sensor Board) analog-digital gewandelt und anschließend dem Netzwerk als konsistentes Systemabbild zur Verfügung gestellt. Für die Aufnahme der

Messwerte werden je zwei Strom- und Spannungssensoren sowie ein Resolver verwendet.

Um ein berührungsloses Einschalten der Anlage zu ermöglichen, werden die im Umrichter verbauten Schütze über eine industrielle, digitale Output-Klemme angesteuert. Dies erhöht die Sicherheit im Umgang mit berührungsgefährlichen Spannungen und erlaubt netzwerkgesteuerte Notabschaltungen im Fehlerfall. Abbildung 3.1 visualisiert den Aufbau des Antriebssystems anhand der zur Verfügung stehenden Komponenten, unter Anwendung des modularen Denkansatzes.

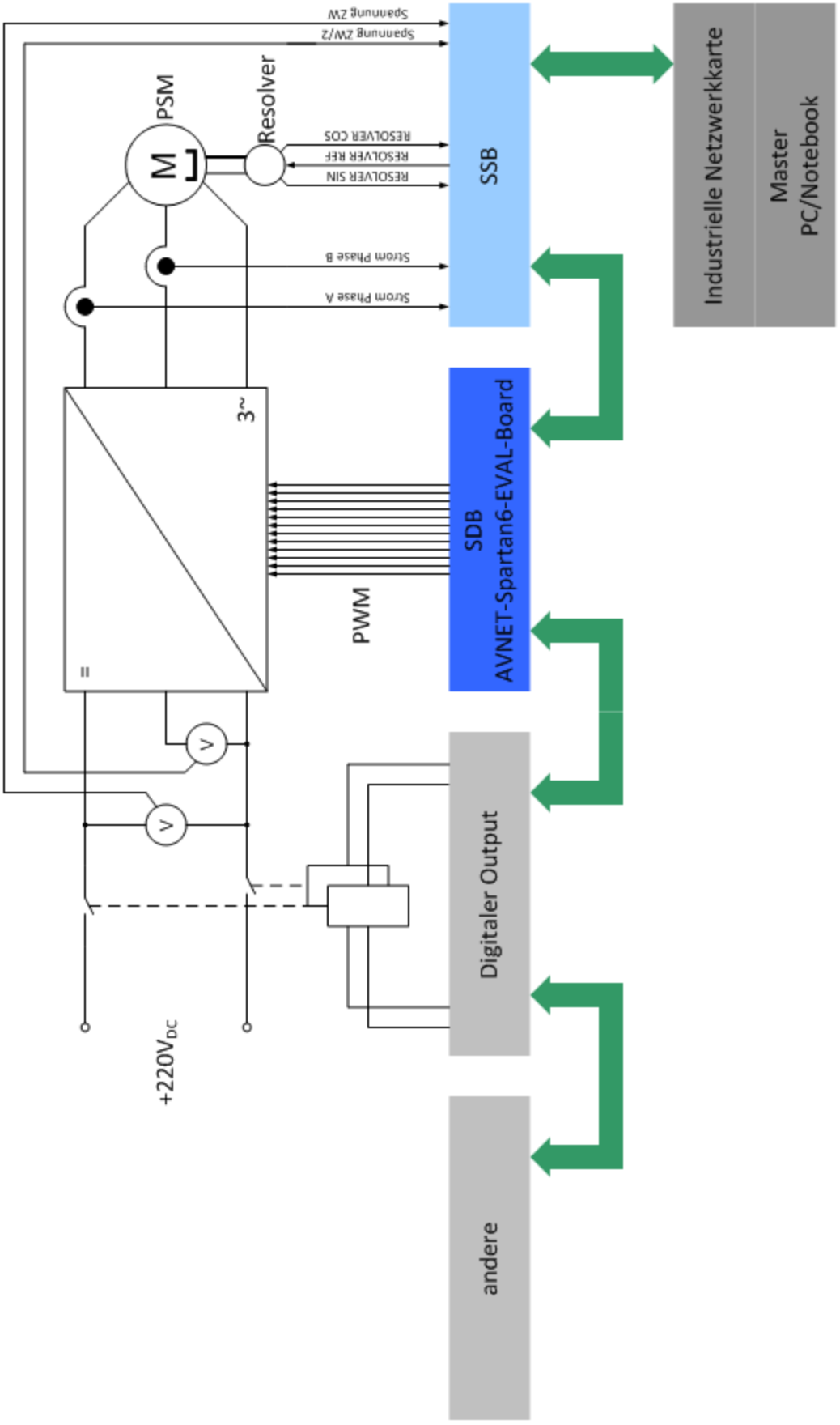


Abbildung 3.1: Basiskonzept des Servoantriebes auf Grundlage verteilter Komponenten | Quelle: Eigene Darstellung

3.2. Ableitung der speziellen Anforderungen

3.2.1. Anforderungen an den Testaufbau

Aufgrund des zuvor beschriebenen Aufbaus ergeben sich zwangsläufig eine Reihe unterschiedlicher Anforderungen, die sich auf zwei übergeordnete Baugruppen aufteilen lassen. Die erste Baugruppe umfasst aus regelungstechnischer Sicht die Regelstrecke in Form des leistungselektronischen Stellglieds (Umrichter) und des elektromechanischen Wandlers (Motor) sowie deren Energieversorgung (Netzteil). Diese sollen, perspektivisch betrachtet, unabhängig vom restlichen Testaufbau austauschbar sein, um die Mess- und Regeleinrichtung auch auf andere Antriebssysteme übertragen zu können. Demzufolge bilden diese Bauteile einen austauschbaren Block, der eine definierte Schnittstelle zur Mess- und Regeleinrichtung benötigt.

Die zweite Baugruppe bilden die vernetzten Teilnehmer des restlichen Regelkreises in Form der Messeinrichtung (Smart Sensor Board) und der Regeleinrichtung (AVNET Development Board), sowie einer digitalen Output-Klemme zum berührungslosen Einschalten der Umrichterschütze. Auch bei dieser Baugruppe ist eine definierte Schnittstelle vorgesehen, um sowohl die Sensorsignale aufzunehmen, als auch die separate Energieversorgung (24 V / DC bei ca. 1 A -> Standard Labornetzteil) der einzelnen Teilnehmer zu gewährleisten.

Um eine kurze Einarbeitung und leichte Wartung des Testaufbaus zu ermöglichen, ist zudem ein hohes Maß an Übersichtlichkeit im Bereich der Anordnung und Verdrahtung gefordert. In dieser Anforderung mit inbegriffen ist ein ansprechendes Design und ein hohes Maß an Bedienerfreundlichkeit, um den Testaufbau sowohl bei internen Vorstellungen, als auch bei Messeauftritten präsentieren zu können. Hierfür werden die vorhandenen Netzwerkteilnehmer auf einer massiven und transportablen Konstruktion befestigt.

3.2.2. Anforderungen an die Datenübertragung

Bei den Anforderungen an die Datenübertragung gilt es ebenfalls einer Vielzahl an unterschiedlichen Aufgabenstellungen gerecht zu werden. Hierbei ist jedoch die Einschränkung durch die Verwendung eines echtzeitfähigen, Ethernet-Basierenden (RTE) Industriestandards seitens der ESW GmbH gegeben.

Da es sich bei der Datenübertragung im Wesentlichen um das zyklische Rückführen der aktuellen Messwerte eines dynamischen Prozesses an die digitale Regelung handelt, ist eine sehr geringe Zykluszeit von übergeordnetem Interesse. Zusätzlich darf dieses zyklische Aktualisieren nur eine sehr geringe Varianz (Jitter) aufweisen, um den Regelungsprozess nicht nachteilig zu beeinflussen.
Um die Mess- und Regeleinrichtung zeitlich zu koordinieren, ist zudem die Möglichkeit der Synchronisation unterschiedlicher Teilnehmer wünschenswert. Diese soll analog zur Zykluszeit einen geringen Jitter aufweisen.
Um auch den hohen Sicherheitsanforderungen in militärischen Zielapplikationen gerecht zu werden, ist ein umfassendes Sicherheitspaket in Form von Redundanz, Lichtwellenleiter und ggf. speziellen Sicherungsprotokollen notwendig.
Ebenfalls durch die Zielapplikation vorgegeben, ist ein hohes Maß an Verfügbarkeit und Akzeptanz des auszuwählenden Standards. Diese gewährleisten eine gewisse Sicherheit in Bezug auf eine langjährige Liefermöglichkeit, die aufgrund der ebenso langjährigen Produktlebenszyklen notwendig ist.
Da der Testaufbau lediglich den Grundstein zur Technologieerschließung eines neuen Datenübertragungsstandards und zur angestrebten Modularisierung bildet, ist zudem die zukünftige Erweiterbarkeit in Form von Flexibilität und flächendeckender Topologie von relevantem Interesse.
Aufgrund des hausinternen Know-how der ESW GmbH im Bereich FPGA-Basierender Lösungskonzepte und deren Vorteil im Bezug auf Flexibilität gegenüber ASICs, wird bei der Lösungsfindung zudem die vom Markt angebotene Hardware bewertet.
Sämtliche Anforderungen sind demzufolge in der Lösungsfindung zu berücksichtigen und sinnvoll zu gewichten.

3.2.3. Timing-Anforderungen

Das Kapitel der Timing-Anforderungen beschreibt den zeitlichen Signalverlauf, der für die Regelung relevanten Mess- und Steuersignale sowie deren Abhängigkeiten untereinander. Auch hierbei wird wiederum, unter Anwendung des modularen Denkansatzes, eine Unterscheidung zwischen Mess- und Regeleinrichtung vorgenommen.

Timing des Smart Sensor-Board

Den zeitlichen Verlauf der für das Sensor Board relevanten Signale und Bearbeitungsprozesse stellt die Abbildung 3.2 dar. Die Dependenz der Signale wird mithilfe von Pfeilen gekennzeichnet.

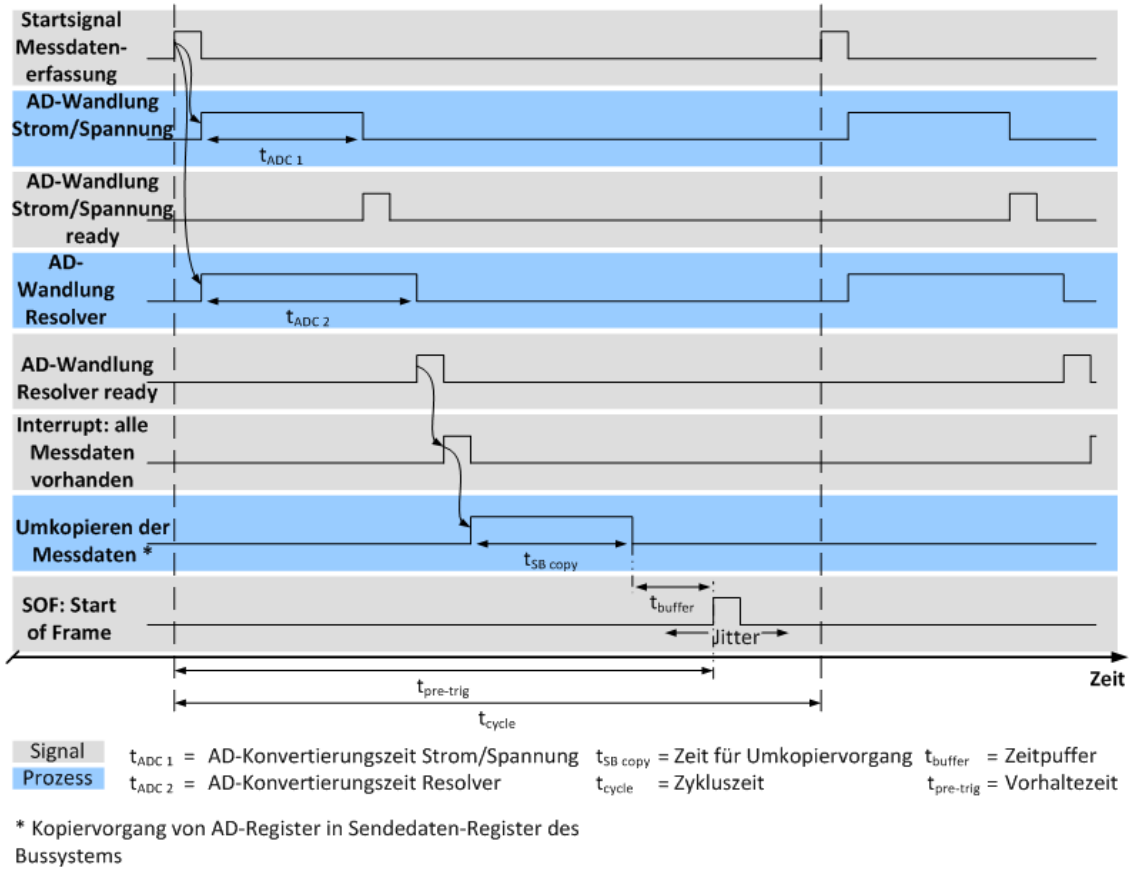


Abbildung 3.2: Timing Smart Sensor Board | Quelle: Eigene Darstellung

Ausgelöst wird die AD-Wandlung der Messsignale mithilfe eines speziellen Impulses (*Startsignal Messdatenerfassung*). Daraufhin startet die Wandlung der analogen Messwerte von Strom, Spannung und Winkel in digitalisierte Daten. Der Zeitraum der Wandlung wird in der Abbildung 3.2 durch t_{ADC1} und t_{ADC2} angegeben. Die unterschiedlichen Größenordnungen der Konvertierungszeiten resultieren aus den ungleichen Datengrößen. Die AD-Wandlung der Strom- und Spannungswerte liefert 12 Bit große Datenpakete, wohingegen die Resolverwerte in einen 16 Bit Digitalwert umgewandelt werden. Für die weitere Betrachtung ist folglich t_{ADC2} als die längere Zeit maßgebend. Mithilfe eines weiteren Signals (*AD-Wandlung Resolver ready*) quittieren die AD-Wandler den Konvertierungsvorgang, wodurch ein Interrupt-Signal (*alle Messdaten*

vorhanden) ausgelöst wird. Dieses Signal wird wiederum genutzt, um den Umkopiervorgang der Messdaten vom Registerbereich der AD-Wandler in den Registerbereich der Datenübertragung zu starten. Die hierfür benötigte Zeit ist in der Abbildung mit t_{copy} angegeben. Nach dem Kopiervorgang erreicht ein neuer Daten-Frame (*SOF – Start of Frame*) das Sensor-Board.

Die Abbildung verdeutlicht somit, dass das Starten der AD-Wandlung mit einer berechenbaren Vorhaltezeit ausgelöst werden muss, um zu gewährleisten, dass dem Netzwerk ein aktueller Datensatz zur Verfügung steht. Aufgrund des zu erwartenden Jitter des SOF ist zudem ein zusätzlicher Sicherheitszeitraum zwischen dem Ende des Kopiervorgangs und den SOF vorzusehen. Die messtechnische Ermittlung der beschriebenen Zeiten ist dem Anhang 1 zu entnehmen. Die Vorhaltezeit $t_{pre\text{-}trig}$ ergibt sich rechnerisch wie folgt:

$$t_{pre-trig} = t_{ADC2} + t_{copy} + t_{buffer} \tag{3.1}$$

Der Zeitanteil t_{buff} kann folgendermaßen abgeschätzt werden:

$$t_{buffer} = t_{Jitter\,max} \cdot S \tag{3.2}$$

wobei S einen ausreichend dimensionierten Sicherheitsfaktor repräsentiert.

Timing des Regler-Boards

Der zeitliche Verlauf der für das Regler-Board relevanten Signale und Prozesse wird in Abbildung 3.3 aufgezeigt. Anders als beim Sensor-Board gibt es keine besonderen Vorhaltezeiten zu berücksichtigen. Wünschenswert ist jedoch die automatische Generierung eines Interrupts nach Frame-Ende (*EOF – End of Frame*) oder dem Erhalten neuer Daten, um die Folgeprozesse zeitlich optimiert anzuregen.

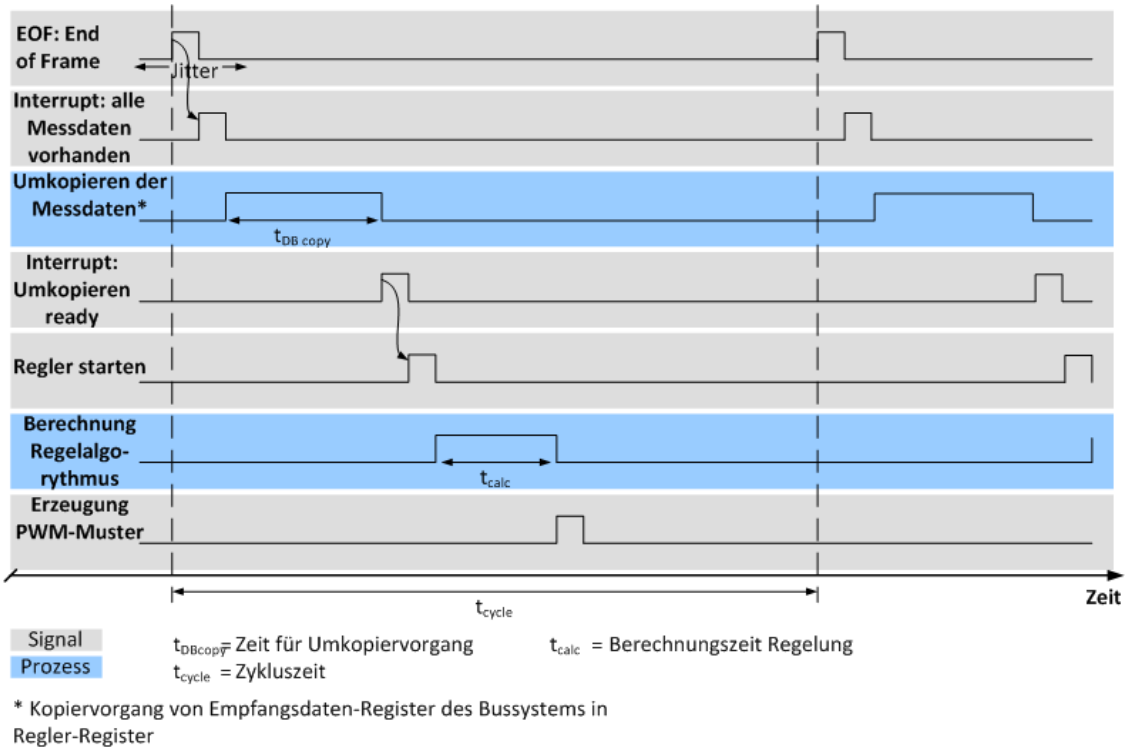


Abbildung 3.3: Timing Smart Driver Board | Quelle: Eigene Darstellung

Es ist darüber hinaus zu gewährleisten, dass in jedem Zyklus aktualisierte Messdaten für die Regelung zur Verfügung stehen. Ferner ermöglicht die Kombination von Abbildung 3.2 und Abbildung 3.3 das Ermitteln des Messdatenalters. Da es sich bei der Servoantriebsregelung um einen dynamischen Prozess handelt, ändert dieser permanent seinen Ist-Zustand. Die Messwerte sind folglich nicht mehr aktuell nachdem sie vom Sensor-Board erfasst und zum Regler-Board übermittelt wurden. Ist das Alter der Messdaten jedoch bekannt und zeitlich nahezu konstant, kann mithilfe der Regelung dieses entstandene Delay kompensiert werden.

Eine Abschätzung kann anhand der Abbildung 3.2 und Abbildung 3.3wie folgt vorgenommen werden:

$$t_{data\ delay} = t_{pre\ trig} + t_{travel\ time} + t_{DB\ copy} \tag{3.3}$$

Für die Durchlaufzeit einer Ethernet-Leitung kann $t_{travel\ time}$ = 550ns je 100m Leitungslänge angenommen werden[15].

[15] Vgl. (Beckhoff Automation GmbH, 2013).

3.3. Erarbeitung eines Lösungskonzeptes

Zur Erarbeitung eines geeigneten Lösungsansatzes werden die zuvor abgeleiteten Anforderungen bzgl. der Datenübertragung in Form einer Matrix gegenübergestellt. Zudem werden die unterschiedlichen Anforderungen je nach Priorität gewichtet und die vorhandenen Marktteilnehmer entsprechend analysiert. Anhand der Eignung des Systems werden 0, 1 oder 2 Punkte, multipliziert mit der Gewichtung, vergeben. Erfüllt ein Standard die Anforderung nicht, so führt dies zum Ausschluss. Die erhobenen Daten, sowie die Bewertung sind der Tabelle 3.1 zu entnehmen.

Tabelle 3.1: Bewertungsmatrix der Marktsituation | Quelle: (PI, 2012) | (ODVA, 2012) | (EtherCAT Technologie Group, 2012) | (EPSG, 2012) | (Modbus Organization, Inc., 2012) | (Quest Trend Magazine, 2012)

Anforderung	Gewichtung	PROFINET IO	EtherNet/IP	EtherCAT	POWERLINK	Open Modbus/ TCP
geringe Zykluszeiten (vgl. bei 1 Teilnehmer)	2	9,8µs 2 Punkte	9,8µs 2 Punkte	8,42µs 2 Punkte	24µs 1 Punkte	10ms 0 Punkte
Genauigkeit der Synchronisation	1	< 1µs 2 Punkte	10µs 0 Punkte	< 1µs 2 Punkte	< 1µs 2 Punkte	keine Angaben
Maximaler Jitter	2	1µs 1 Punkt	< 1µs 2 Punkte	1µs 1 Punkt	1µs 1 Punkt	keine Angaben
Lichtwellenleiter	1	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte
flächendeckende/flexible Topologie	1	Switched: Baum, Stern, kurze Linie 1 Punkt	aktiver Stern, Linie nur via Switch 0 Punkte	Linie, Ring, Baum, Stern 2 Punkte	Hub: Baum, Stern, kurze Linie 1 Punkt	Stern, Linie nur via Switch 0 Punkte
Hohe Verfügbarkeit und Akzeptanz	1	300 Mitglieder 42% Marktanteil 2 Punkte	400 Mitglieder 8% Marktanteil 1 Punkt	2000 Mitglieder 24% Marktanteil 2 Punkte	400 Mitglieder 10% Marktanteil 1 Punkt	78 Mitglieder 3% Marktanteil 0 Punkte
Verfügbare Hardware	2	ASIC 0 Punkte	FPGA, netX, MAC 2 Punkte	FPGA, ASIC, netX 2 Punkte	FPGA, netX, MAC 2 Punkte	FPGA, netX 1 Punkt
Redundante Auslegung	1	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte	möglich 2 Punkte
Sichere Datenübertragung/ Safety-Standard	1	PROFIsafe openSAFETY 2 Punkte	CIP Safety openSAFETY 2 Punkte	openSAFETY Safety over EtherCAT 2 Punkte	openSAFETY 1 Punkt	openSAFETY 1 Punkt
Gesamtpunktzahl:		17	19	**22**	17	7

Die Auswertung der Entscheidungsmatrix anhand der Gesamtpunktzahl verdeutlicht einen Performancevorteil vom EtherCAT Standard gegenüber den alternativen Produkten am Markt. Insbesondere bei den doppelt gewichteten Anforderungen erreichen sowohl EtherCAT, als auch Ethernet/IP nahezu alle Punkte. Im Gegensatz zu Ethernet/IP deckt der EtherCAT Standard jedoch alle Anforderungen zufriedenstellend ab. Der Ethernet/IP Standard disqualifiziert sich hingegen durch eine verhältnismäßig ungenaue Synchronisation der Teilnehmer, sowie durch den erhöhten Verdrahtungsaufwand aufgrund der, für den Testaufbau unkomfortablen, Sterntopologie.

Die Datenübertragung via PROFINET IO ist für eine Anwendung innerhalb der ESW GmbH ebenfalls nicht rentabel, da es hier lediglich die Möglichkeit gibt, Slave-Applikationen mittels eines ASICs zu implementieren. Das vorhandene Know-how der Entwicklungsabteilung, sowie die hohe Flexibilität eines FPGAs können hier nicht in Anspruch genommen werden.

Der Datenübertragungsstandard POWERLINK bedient, ähnlich wie EtherCAT, jede Anforderung ausreichend, verliert jedoch in Summe durch die geringere Gesamtpunktzahl. Ausschlaggebend ist zudem der deutliche Performancenachteil im Bezug auf die Zykluszeit, die im Vergleich zum EtherCAT dreimal langsamer ist.

Der Open Modbus/TCP disqualifiziert sich durch einen Mangel an verfügbaren Informationen. Aufgrund der wenig performanten Zykluszeit ist eine weiterführende Marktrecherche zudem nicht notwendig.

Um den Anforderungen an den Testaufbau gerecht zu werden, wird mithilfe des firmeninternen Musterbaus eine solide Grundplatte gefertigt. Auf dieser sind die Netzwerkteilnehmer mittels Schraubverbindung montiert. Als Schnittstelle zwischen Antriebssensorik und Messeinrichtung dienen industrielle Durchgangsklemmen aus der Schaltschranktechnik, die mithilfe einer Tragschiene (ugs. Hutschine) auf die Grundplatte montiert werden. Diese Schnittstelle dient darüber hinaus zum Einspeisen der elektrischen Energie ($24V_{DC}$, ca. 1A), sowohl für die Netzwerkteilnehmer, als auch für Teilkomponenten des Umrichters.

Als Schnittstelle zwischen Regeleinrichtung und Umrichter wird auf den firmeninternen Standard zurückgegriffen (96-poliger Steckverbinder). Um den Verkabelungsaufwand jedoch möglichst gering zu halten, wird hierfür durch die Lehrabteilung ein spezielles

Adapter-Board gefertigt, welches im Umrichter auf die Standardschnittstelle montiert wird. Das Adapter-Board bezieht das vom Regler-Board modulierte PWM-Muster über eine Flachbandleitung (16 – polig, 12 x PWM, 1x Fehlersignal, 2x Spannungsversorgung, 1 x frei). Zudem erhöht das Board den Spannungspegel des PWM-Musters von 3,3V (max. Ausgangsspannung FPGA) auf die notwendigen 5V (Eingangsspannung Leistungselektronik/Umrichter) mittels eines Pegelwandlers.

Realisiert wird das Regler-Board zunächst auf einem Entwicklungsboard der Firma AVNET Electronics[16]. Für die I/O-Schnittstelle zum Umrichter wird zudem die I/O-Erweiterungsplatine XM105 (Xilinx, 2013) verwendet. Die Ethernet-Schnittstelle wird ebenfalls durch eine separate Aufsteckkarte realisiert. Verwendung findet hierbei das ISM Networking FMC Modul[17] der Firma AVNET Electronic.

Als Messeinrichtung wird eine proprietäre Weiterentwicklung des bisherigen Sensor-Boards verwendet. Sie besteht im Wesentlichen aus einem FPGA und zwei Ethernet-Schnittstellen, sowie Steckverbindern zur Messsignalaufnahme und geeigneten AD-Wandlern.

Zum Schalten der Umrichter-Schütze findet eine digitale Ausgangsklemme[18] der Firma Beckhoff Automation GmbH Anwendung. Da der Ausgangsstrom der Ausgangsklemme jedoch zu gering ist, wird zwischen Schütz und Klemme ein weiteres Relais geschaltet.

Um eine bestmögliche Zykluszeit bei geringem Jitter zu erreichen, wird als Mastersystem ein handelsüblicher PC verwendet, der mit einer für EtherCAT optimierten Netzwerkkarte[19] der Firma Beckhoff ausgerüstet wird. Umgesetzt wird dieses Mastersystem mithilfe der Software TwinCAT[20], die ebenfalls von der Firma Beckhoff Automation stammt.

Sämtliche Verkabelung des Aufbaus wird mithilfe von Kabelbindern und geeigneten Fixierungs-Pads auf der Grundplatte befestigt und übersichtlich angeordnet. Die Abbildung 3.4 visualisiert die Umsetzung des Testaufbaus sowie der Verkabelung und gibt einen Überblick über die Pin-Belegung (siehe Anhang 2) der Durchgangsklemmen.

[16] Vgl. (Xilinx, 2013).
[17] Vgl. (AVNET Electronics, 2013).
[18] Vgl. (Beckhoff Automation, 2013).
[19] Vgl. (Beckhoff Automation, 2013).
[20] Vgl. (Beckhoff Automation, 2013).

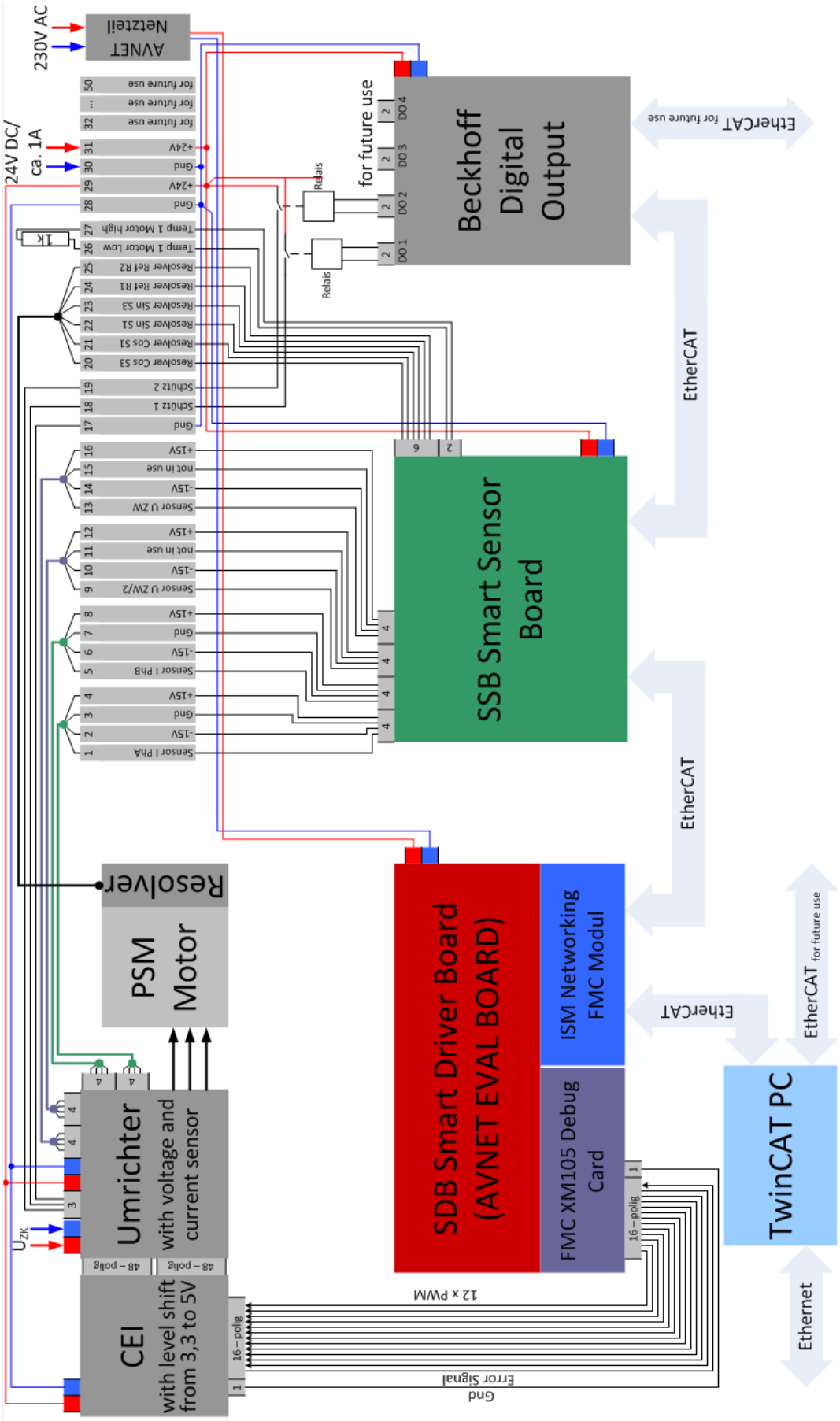


Abbildung 3.4 Aufbau und Signallaufplan des Testantriebes | Quelle: Eigene Darstellung

4. Implementierung des Lösungskonzeptes

4.1. Netzwerk der Verteilten Komponenten

4.1.1. Bussystem

Bei dem anhand von Kapitel 3.3 ausgewählten Datenübertragungsstandard EtherCAT handelt es sich um eines der aktuell performancestärksten Bussysteme im Bereich des Real Time Ethernet. Vertrieben und gefördert wird das Bussystem, welches in der IEC 61158 (Protokolle und Dienste) und IEC 61784-2 (Kommunikationsprofile) spezifiziert ist, durch die EtherCAT Technologie Group (ETG) und ihre ca. 2200 Mitglieder (Stand Februar 2013)[21].

Der grundlegende Unterschied zu vergleichbaren Systemen liegt in der besonderen Form der Datenübertragung und –Auswertung. Ein EtherCAT Master regt hierfür zyklisch die Datenübertragung in Form eines Daten-Frames an. Dieser Frame durchläuft sämtliche Teilnehmer des Netzwerkes nacheinander, wobei jeder Teilnehmer „on the fly“ – also beim Durchlaufen des Frames sowohl den Frame, als auch die eigenen Daten aktualisiert (lesen und schreiben der Teilnehmer). Durch diese Form des Zugriffs wird der Datenframe nur noch um wenige Nanosekunden verzögert, wodurch auch bei komplexen Anlagen innerhalb kürzester Zeit ein komplettes Systemabbild erzeugt wird. Grundlage des hochperformanten Lese- und Schreibzugriffes bildet ein DPRAM[22] Speicherbaustein und die von der Peripherie unabhängige Architektur des EtherCAT Slave Controllers (ESC)[23].

4.1.2. Topologie

Mit Ausnahme des Masters besitzen sämtliche EtherCAT-Teilnehmer in der Regel mindestens zwei Anschlüsse (Dateneingang und Datenausgang) zur Netzwerkintegration. Diese werden meist in Form von RJ45 Buchsen nach außen geführt und können mithilfe herkömmlicher Ethernet Leitungen (mindestens CAT 5 Standard) verbunden wer-

[21] Vgl. (Group EtherCAT Technologie, 2013).

[22] Dual-Port-Memory: Ist ein Speicher, bei dem von zwei Seiten sowohl Lese- als auch Schreibvorgänge möglich sind. Durch den beidseitigen Zugriff können zwei unterschiedliche Systeme gleichzeitig mit gemeinsamen Daten arbeiten, ohne dass sie sich in ihrer Zugriffsgeschwindigkeit beeinträchtigen. Vgl. (Valvana, 2011).

[23] Vgl. (Group EtherCAT Technologie, 2013).

den. Innerhalb eines Ethernet Kabels stehen den Teilnehmern wiederum getrennte Übertragungsleitungen (Rx und Tx) zur Verfügung. Werden mehrere Teilnehmer miteinander verbunden, entsteht oberflächlich betrachtet eine Linientopologie, die durch Verwendung spezieller Teilnehmer mit mehr als zwei Netzwerkanschlüssen zur Baum- oder Sterntopologie erweitert werden kann. Durch die im Ethernet-Kabel vorhandenen Hin- und Rückleitungen ist jedoch jeder Art von Topologie immer eine geschlossene Ringtopologie unterlagert.

Bei der Verwendung eines Masters mit zwei Netzwerkanschlüssen lässt sich zudem ein redundantes Netzwerk aufbauen, welches selbst im Fall einer Leitungsunterbrechung einen uneingeschränkten Restbetrieb ermöglicht. Abbildung 4.1 zeigt die für den Testaufbau verwendete Linientopologie sowie die Anordnung der Teilnehmer und verdeutlicht zudem die unterlagerte Ringtopologie.

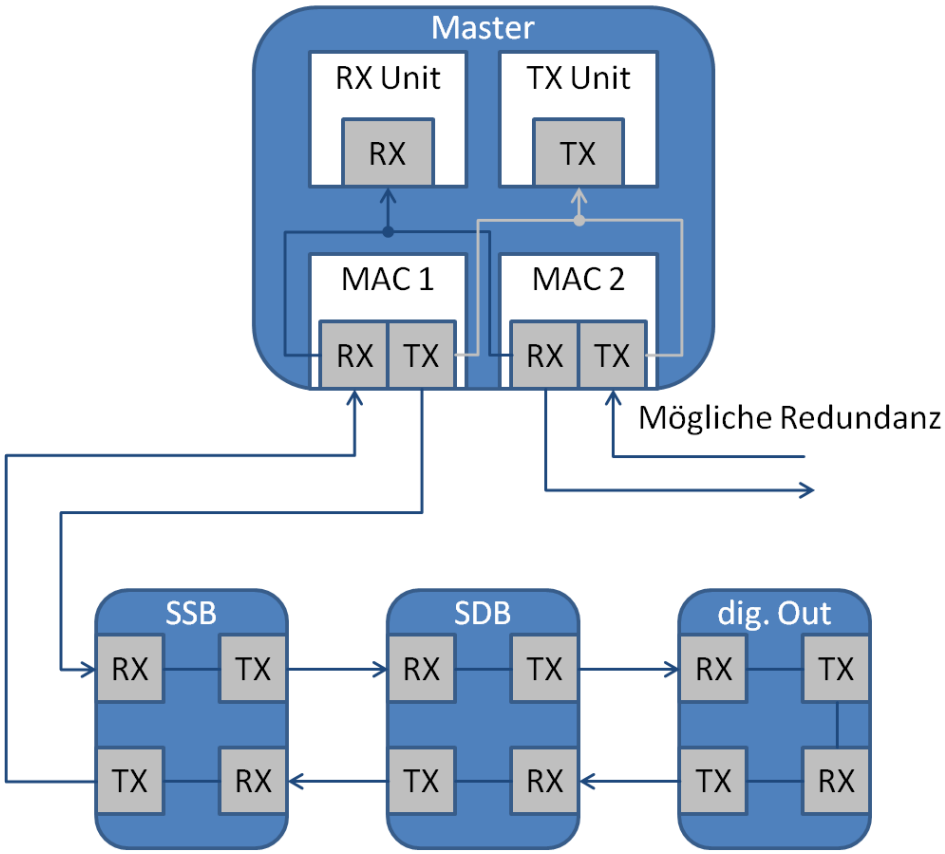


Abbildung 4.1: Topologie des Testaufbaus | Quelle: Eigene Darstellung

Die Anordnung der Teilnehmer ergibt sich aus den zuvor in Kapitel 3.2.3 erarbeiteten Timing-Anforderungen. Das Smart Sensor Board muss folglich zunächst die Messdaten

erfassen, bevor sie an das Smart Driver Board weitergeleitet werden können. Aus diesem Grund befindet sich das SSB logisch vor dem SDB. Die digitale Ausgangsklemme wird aus Sicherheitsgründen als logisch letzter Teilnehmer eingebunden. Bei einer Leitungsunterbrechnung sowie dem Ausfall eines Teilnehmers wird die Klemme somit vom Netzwerk getrennt und schaltet daraufhin in einen sicheren Zustand. Die digitalen Ausgänge öffnen, die angeschlossenen Relais werden spannungsfrei und die Schütze unterbrechen die Spannungsversorgung des Umrichters. Im Fehlerfall liegt folglich keine berührungsgefährliche Spannung am Umrichter an.
Die Abweichung zwischen logischer und physikalischer Anordnung entsteht durch die optimierte Platzausnutzung auf der Grundplatte und die unterschiedlichen Baugrößen der Teilnehmer.

4.1.3. Protokoll

Bei der Entwicklung des EtherCAT Standards wurde seitens der ETG berücksichtigt, dass neben Kaufkomponenten und Eigenentwicklungen auch jeder netzwerkfähige PC als EtherCAT Master fungieren kann. Als Folge dessen wird als Grundgerüst des EtherCAT Protokolls das in der Computerwelt etablierte Ethernet Frame als Träger verwendet. Integriert wird das EtherCAT Protokoll, wie anhand der Abbildung 4.2 ersichtlich, in den eigentlichen Datenbereich des Ethernet Frames. Bei der ausschließlichen Verwendung von EtherCAT Teilnehmern in einem Netzwerk kann jedoch auf den Ethernet Rahmen verzichtet werden. Die Master Software TwinCAT erkennt dies automatisch und verringert somit den zu übertragenden Overhead.
Die ersten zwei Byte nach dem Ethernet Header und den optionalen UDP oder IP Protokollen bilden den EtherCAT-Header und beinhalten Angaben über die Länge des Protokolls und des verwendeten EtherCAT-Typs. Die darauf folgenden 48 bis 1498 Byte bilden den eigentlichen Datenbereich des EtherCAT-Protokolls, der wiederum in n-verschiedene Datagramme aufgeteilt werden kann. Ein Datagramm beschreibt in der Regel den Kommunikationsabschnitt von genau einem Teilnehmer und untergliedert sich wiederum in einen eigenen Slave-Header, einen Datenbereich und den Working Counter.
Der Working Counter wird von jedem Slave in Abhängigkeit seiner ausgeführten Aktion um eins (Lesezugriff), zwei (Schreibzugriff) oder drei (Lese- und Schreibzugriff) inkre-

mentiert. Der Master vergleicht nach Empfang des Frames den erwarteten mit dem im Frame enthaltenen Working Counter und detektiert somit Fehlzugriffe[24].

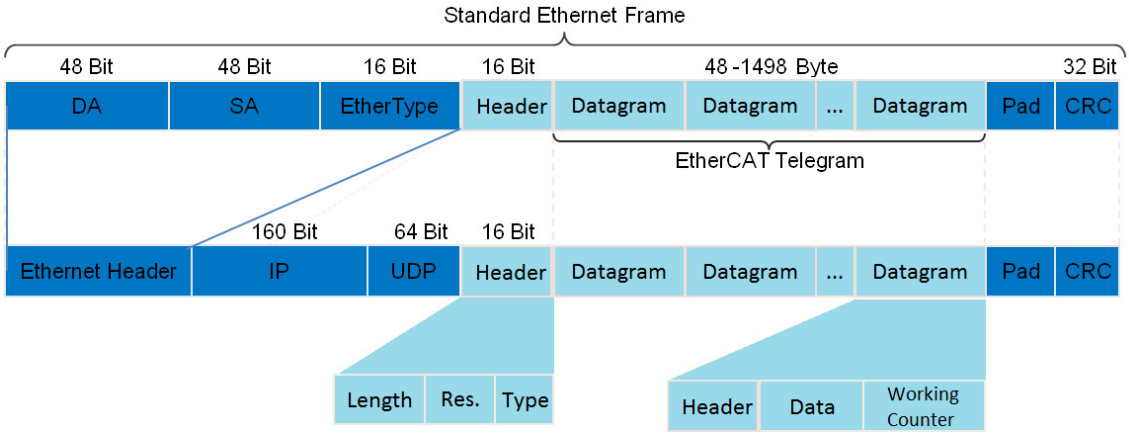


Abbildung 4.2: Allgemeiner Aufbau eines EtherCAT Protokolls | Quelle: Eigene Darstellung

Organisiert werden sämtliche Daten des Netzwerks in einem bis zu vier Gigabyte großen logischen Adressbereich (0x00000000 bis 0xFFFFFFFF), zu dessen gesamter Ausnutzung mehrere Frames benötigt werden. Jedem einzelnem Adressbereich (z.B. Ausgangsdaten eines Teilnehmers) ist zudem eines der in Tabelle 4.1 beschriebenen Attribute[25] zugeordnet, um bspw. die Zugriffsberechtigung zu definieren.

Tabelle 4.1: Ausgewählte EtherCAT Kommandos | Quelle: (EtherCAT Technologie Group, 2013)

BRD - Broadcast Read	lesen eines physikalischen Speicherbereichs für alle Slaves
BWR - Broadcast Write	schreiben eines physikalischen Speicherbereichs für alle Slaves
BRW - Broadcast Read and Write	lesen und schreiben eines physikalischen Speicherbereichs für alle Slaves
LRD - Logical Read	lesen eines logischen Speicherbereichs für einzelnen Slave
LWR - Logical Write	schreiben eines logischen Speicherbereichs für einzelnen Slave
LRW - Logical Read and Write	lesen und schreiben eines logischen Speicherbereichs für einzelnen Slave

[24] Vgl. (EtherCAT Technologie Group, 2013).
[25] Die Tabelle ist in Hinsicht des Testaufbaus auf die wichtigsten Kommandos reduziert. Alle weiteren Kommandos können der angegebenen Quelle entnommen werden.

4.1.4. Synchronisation

Zur Synchronisation von Teilnehmern stellt EtherCAT das System der verteilten Uhren (DC - Distributed Clocks) zur Verfügung. Hierfür wird zunächst ein BRD Kommando vom Master verschickt, indem jeder Teilnehmer vermerkt, wann der Frame den Slave auf dem Hin- und auf dem Rückweg durchläuft. Anhand dieser Zeitstempel errechnet der Master sämtliche Laufzeitverzögerungen des Netzwerkes. Anschließend wird allen an der Synchronisation beteiligten Slaves eine gemeinsame Systemzeit, abzüglich der errechneten Verzögerungszeit, mittels eines BWR Kommandos mitgeteilt. Hierfür dient meist die lokale Uhrzeit des ersten Slaves. Um anschließend ein Auseinanderdriften der Zeiten zu vermeiden, überwacht der Master die einzelnen Systemzeiten der Teilnehmer und korrigiert diese im laufenden Betrieb. Beim zeitweiligen Ausfall der Synchronisation durch eine Leitungsunterbrechung kann der Slave anhand seiner lokalen Uhrzeit einen sicheren Restbetrieb aufrecht erhalten, bis eine erneute Synchronisation nach Behebung der Unterbrechung möglich wird. Sämtliche Slaves, die das Verfahren der verteilten Uhren unterstützen, lassen sich somit unabhängig von ihrer Entfernung mit einer Genauigkeit kleiner 100ns synchronisieren.

Physikalisch genutzt werden können die verteilten Uhren mittels zweier spezieller Signale (SYNC 1 und SYNC 2), die sowohl bei FPGA- als auch bei ASIC-Ausführung zur Verfügung stehen. Die SYNC-Signale lassen sich zudem zeitlich zur Systemzeit in Schritten von einer Mikrosekunde verschieben. Zusätzlich kann die SYNC-Generierung durch einen Multiplikator/Divisor in Bezug auf die Zykluszeit hoch- bzw.- heruntergetaktet werden (Vorgehen zum Konfigurieren siehe Anhang 3). Soll bspw. ein SYNC-Ausgang lediglich jeden zweiten Zyklus ein Signal generieren, so wird mithilfe des Masters der Divisor 0,5 eingestellt.

4.1.5. EtherCAT Slave Controller

Sämtliche Funktionalität eines EtherCAT Teilnehmers wird mittels des EtherCAT Slave Controllers (ESC) zur Verfügung gestellt. Dieser ist in Form eines ASICs oder FPGAs fest im Teilnehmer integriert und bildet die Schnittstelle zwischen dem EtherCAT Netzwerk und den vom Teilnehmer organisierten Prozessdaten. Die Abbildung 4.3 veranschaulicht den Aufbau des ESCs.

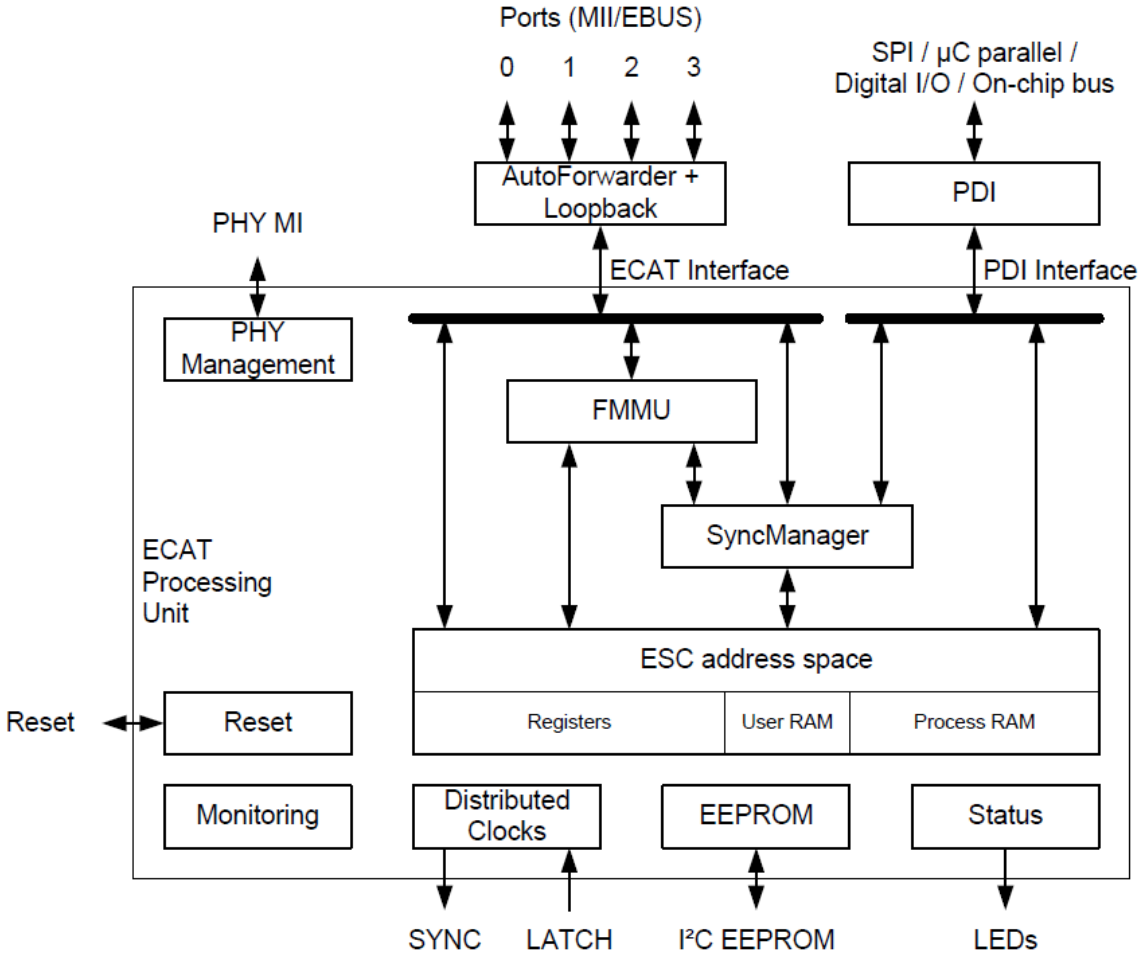


Abbildung 4.3: Aufbau und Organisation ESC | Quelle: (Beckhoff Automation GmbH, 2013, S. 1 in Section I)

Von besonderem Interesse, in Bezug auf die weitere Vorgehensweise, sind die Einheiten FMMU (Fieldbus Memory Management Unit) und SyncManager sowie der ESC Adressraum.

Der Adressraum besitzt eine Größe von 64kByte wobei die ersten 4kByte (0x0000 – 0x0FFF) von EtherCAT-spezifischen Registern und dem User RAM belegt sind. Die restlichen 60kByte stehen der angeschlossenen bzw. integrierten Peripherie (z.B. FPGA) in Form des PDI (Prozessdaten Interface) zur Verfügung[26].

Die SyncManager Unit beherbergt bis zu 16 verschiedene SyncManager, die für die Prozessdatenverwaltung verwendet werden. Sie verbinden erstellte Prozessvariablen oder -speicherbereiche mit physikalischen Speicherbereichen im ESC Adressraum. Zusätzlich organisieren sie die Zugriffe auf diesen physikalischen Speicher[27]. Unabhängig von ihrer Verwendung besitzt jeder EtherCAT Teilnehmer i.d.R. die SyncManager SM0

[26] Vgl. (Beckhoff Automation GmbH, 2013, S. 3 in Section I).
[27] Vgl. (Beckhoff Automation GmbH, 2013, S. 40 in Section I).

bis SM3, wobei SM2 für die Ausgangs- und SM3 für die Eingangsvariablen des PDI verwendet werden.

Die FMMU hingegen verknüpft den in Kapitel 4.1.3 beschriebenen logischen Adressraum des EtherCAT-Prozessabbildes mit dem im Teilnehmer real verfügbaren physikalischen Speicher des ESCs[28]. Die Abbildung 4.4 beschreibt den vollständigen Kommunikationspfad eines Datums vom Master über den ESC bis hin zum PDI.

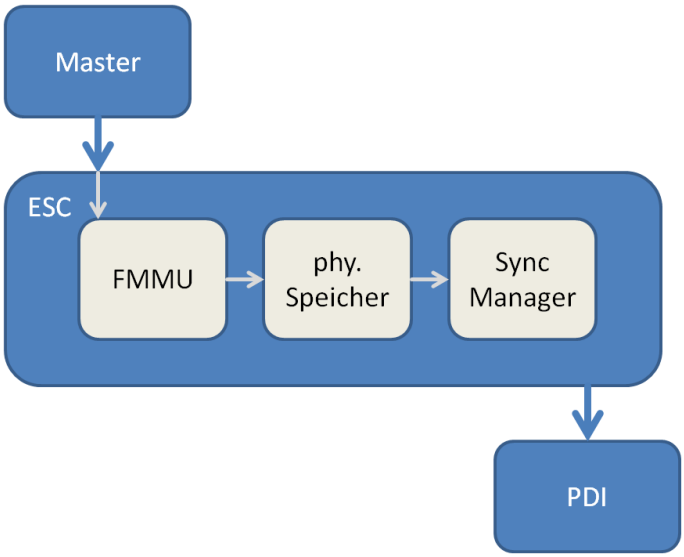


Abbildung 4.4: Kommunikationspfad EtherCAT | Quelle: Eigene Darstellung

Das vom Master im logischen Adressbereich organisierte Datum wird vom EtherCAT Teilnehmer empfangen, woraufhin die FMMU des ESCs das Datum in den physikalischen Speicher überträgt. Der SyncManager kopiert das Datum aus dem physikalischen Speicher in das PDI, sodass dieses bspw. einen Status aktualisieren kann. Bei einem Sendevorgang seitens des Teilnehmers wird dieser Prozess in umgekehrter Reihenfolge ausgeführt.

Der anhand der Abbildung 4.3 ersichtliche direkte Kommunikationspfad vom EtherCAT Netzwerk und PDI auf den physikalischen Speicher dient dem Beschreiben spezieller Konfigurationsregister (bspw. Freischalten von Interrupts).

4.1.6. Interrupts und spezielle Signale

Um einen EtherCAT Teilnehmer das Reagieren auf bestimmte Ereignisse zu ermöglichen, stellt der implementierte EtherCAT Slave Controller eine Reihe spezieller Signale

[28] Vgl. (Beckhoff Automation GmbH, 2013, S. 38 in Section I).

zur Verfügung. Ein Teil dieser Signale besitzt zudem die Fähigkeit als Interrupt zu fungieren, sodass der ansonsten zyklisch arbeitende Prozess eines Teilnehmers ereignisgesteuert unterbrochen werden kann. Die Abbildung 4.5 zeigt die bereitgestellten Interrupt-Quellen und deren Organisation.

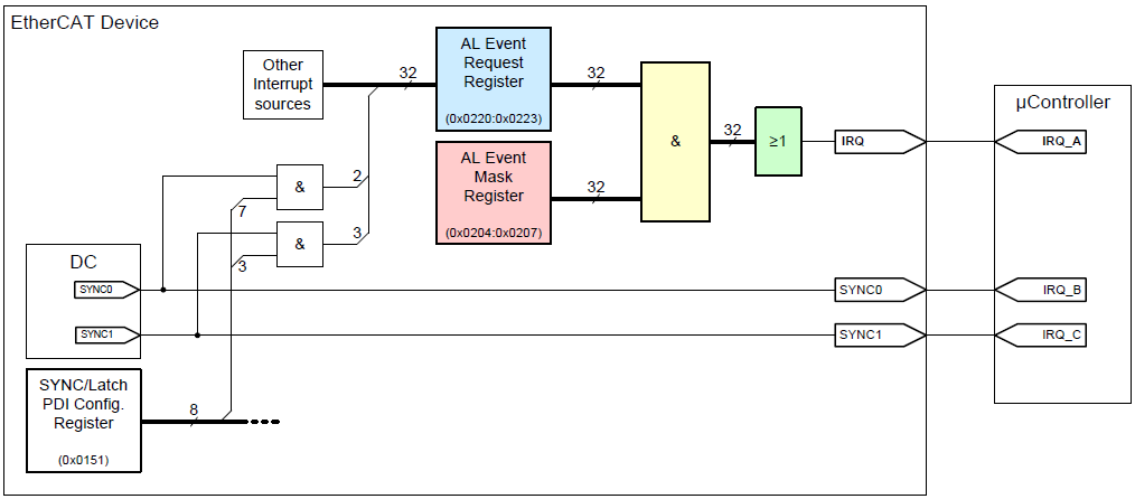


Abbildung 4.5: Interrupt Signale des ESC | Quelle: (Beckhoff Automation GmbH, 2013, S. 81 in Section I)

Zusätzlich zu den in Kapitel 4.1.4 beschriebenen SYNC-Signalen beinhaltet der Block „Other Interrupt Sources“ noch zwei LATCH Eingänge. Diese fungieren als interne Interrupt Controller, indem dort angeschlossene Signale einen Interrupt auslösen können. Die durch die LATCH Eingänge verursachten Interrupts werden zudem mit einem Zeitstempel versehen, deren Auswertung ebenfalls möglich ist.

Das Eintreffen neuer Daten kann ebenso mithilfe eines Interrupts detektiert werden. Ist der verantwortliche SyncManger entsprechend konfiguriert, löst der Speicherzugriff eines FMMUs auf den physikalischen Speicher des SyncManagers einen Interrupt aus.

4.2. Implementierung des EtherCAT IP Core

Bei einem IP Core (intellectual property core) handelt es sich um einen wiederverwendbaren Teil eines Elektronikdesigns, der meist in verschlüsselter Form käuflich erworben werden kann und somit das geistige Eigentum des Entwicklers schützt[29]. Im Fall des EtherCAT IP Cores der Firma Beckhoff Automation GmbH handelt es sich um den in Kapitel 4.1.5 beschriebenen EtherCAT Slave Controller, der als eine Art Black

[29] Vgl. (Gizopoulos, Paschalis, & Zorian, 2004, S. 10).

Box in den frei programmierbaren Teil eines FPGAs integriert werden kann. Die Abbildung 4.6 veranschaulicht dieses Prinzip und gibt einen Überblick über das grundsätzliche FPGA-Design des Regler Boards.

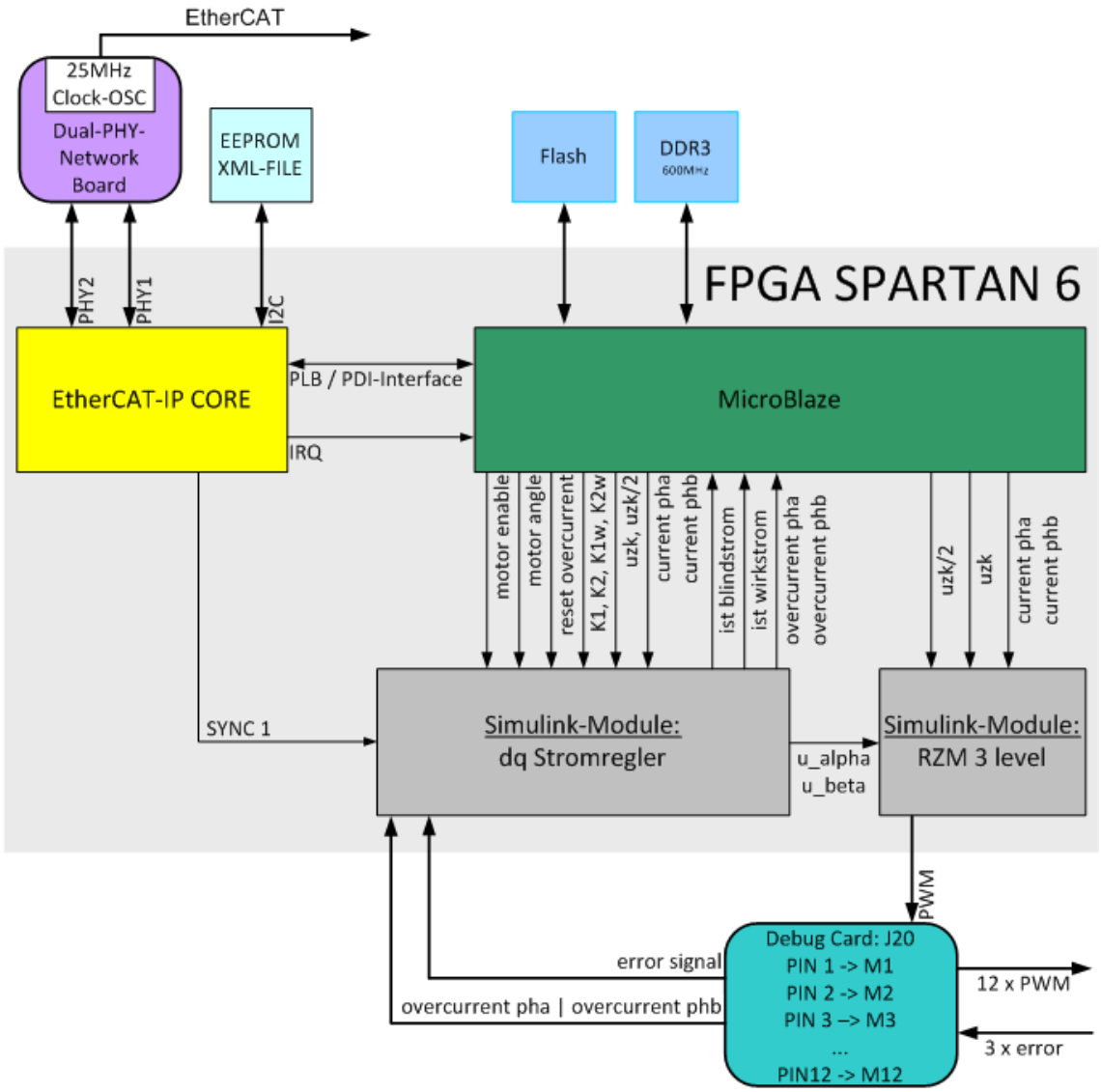


Abbildung 4.6: Prinzip-Skizze FPGA Design SDB | Quelle: Eigene Darstellung

Anhand der Abbildung 4.6 ist ebenfalls zu erkennen, dass zur Bearbeitung der Prozessdaten ein weiterer IP Core Anwendung findet. Hierbei handelt es sich um den MicroBlaze der Firma Xilinx, der mit einer Art Mikrocontroller verglichen werden kann. Zusammen mit dem FPGA-internen PLB Bus (Processor Local Bus) und einem externen Speicher (Flash/DDR3) bildet der MicroBlaze ein sog. „System-on-a-Chip“ (SoC)[30]. Der

[30] Vgl. (Xilinx, 2013).

MicroBlaze wird mithilfe der Programmiersprache C programmiert und dient als Prozessdaten Interface des ESCs.

Analog zum Regler Board wurde ein nahezu identisches FPGA-Design zur Realisierung des Sensor Boards verwendet. Der Prozessdatenaustausch findet hierbei jedoch nicht zwischen Simulink Modulen und MicroBlaze, sondern zwischen MicroBlaze und Registerbereich der AD-Wandler statt.

4.3. Implementierung des EtherCAT Slave Stacks

Da mittels des MicroBlaze eine in C programmierbare Plattform für die Verarbeitung der Prozessdaten zur Verfügung steht, wurde zum Ansprechen und Interagieren mit dem ESC eine umfangreiche Software Bibliothek erworben. Bei dieser Bibliothek handelt es sich um den EtherCAT Slave Stack der Firma PORT GmbH. Die Bibliothek stellt unter anderem Funktionen zur Konfiguration des ESCs bereit und ermöglicht sowohl das zyklische als auch das Interrupt-gesteuerte Aktualisieren der Prozessdaten.

Zur Konfiguration des ESCs greift die Bibliothek über das PDI auf den physikalischen Speicher zu. Der Austausch der Prozessdaten zwischen ESC und MicroBlaze erfolgt hingegen über die SyncManager. Erweitert wurde der Slave Stack um eigens programmierte, Interrupt-gesteuerte Routinen sowie der Main-Routine. Die Main-Routine initialisiert den ESC und bearbeitet zyklisch das Prozessdaten Interface.

Endlosschleife der Main-Routine für das Smart Sensor Board

```
/*SSB Part of Main*/
/*****************************************************************/

      while(1) {                          /* endless loop */

            if (ecInitState == LIB_TRUE) {
                  ec_flushMbox();         /* do the EtherCAT job */
                  }
            else {
                  ecInitState = ecInitEtherCAT();
            }

      }//end of while

// ---------------------------------------------------------------
```

Die Interrupt Routine des SSBs wird im Anschluss an die AD-Wandlung aufgerufen und kopiert die Messwerte der Sensorik in den Speicherbereich der SyncManager. Der ESC stellt diese anschließend mittels der FMMU dem EtherCAT Netzwerk zur Verfügung.

Interrupt Routine zum Umkopieren der Prozessdaten für das Smart Sensor Board

```
/*SSB Interrupt*/
/*****************************************************************/

void DeviceDriverHandler_ESW_Interrupt(void *CallbackRef)
{
//Kopieren der AD-Werte von AD-Registern in ECAT IP Core Register
//immer alle PDO´s aktualisieren!
      *IP_CORE_STROMSENSOR_PHA = *channel_V1;   //PHA -> ADC Ch. 1
      *IP_CORE_STROMSENSOR_PHB = *channel_V2;   //PHB -> ADC Ch. 2
      *IP_CORE_UZWK            = *channel_V4;   //Uzk -> ADC Ch. 5
      *IP_CORE_UZWK_HALF       = *channel_V3;   //Uzk/2 -> ADC Ch. 6
      *IP_CORE_WINKEL          = *motor_winkel;       //Winkel
      *IP_CORE_GESCHWINDIGKEIT = *motor_speed; //Geschwindigkeit

      *mb2peri ^= 0x8000;                      //toggle for debug
      *IP_CORE_TOGGLE_BIT = *mb2peri;

      // Toggle bit 1 from mb2peri register for end of interrupt
      *mb2peri |= 0x0002;
      *mb2peri &= 0xFFFD;
}

// ----------------------------------------------------------------
```

Beim Regler Board konnten die Prozessdaten in zeitkritische und nicht zeitkritische Daten aufgeteilt werden. Ein Teil der nicht zeitkritischen Daten (Input Daten) werden folglich in der Main-Routine umkopiert, um die Bearbeitungszeit der Interrupt Routine möglichst gering zu halten.

Endlosschleife der Main-Routine für das Smart Driver Board

```
/*SDB part of main*/
/*****************************************************************/

      while(1) {                          /* endless loop */

            if (ecInitState == LIB_TRUE) {
                  ec_flushMbox();         /* do the EtherCAT job */
                  }
            else {
                  ecInitState = ecInitEtherCAT();
            }

      //non time critical input data copy
      ESW_IST_BLINDSTROM      = ((unsigned) *p_ist_blindstrom);
      ESW_IST_WIRKSTROM       = ((unsigned) *p_ist_wirkstrom);
```

```
      ESW_OVERCURRENT_PHASE_A = ((char) *p_overcurrent_pha);
      ESW_OVERCURRENT_PHASE_B = ((char) *p_overcurrent_phb);
      ESW_ERROR_SIGNAL        = ((char) *p_error_signal);

      }//end of while

// ------------------------------------------------------------------
```

Der zweite Teil der nicht zeitkritischen Daten (Teile der Output Daten für Regler Parametrierung) muss jedoch in der Interrupt Routine umkopiert werden, da immer der gesamte Speicherbereich eines SyncManagers hintereinander beschrieben werden muss. Eine Optimierung durch Verwendung eines weiteren SyncManagers, der die nicht zeitkritischen Output Daten separat verwaltet ist jedoch möglich.

Interrupt Routine zum Umkopieren der Prozessdaten für das Smart Sensor Board

```
SDB interrupt
/*****************************************************************/

void DeviceDriverHandler_ESW_Interrupt(void *CallbackRef)
{
      //Umkopieren der Sensorwerte in ECAT Register des SyncMan
      *p_current_pha    = *IP_CORE_current_pha;
      *p_current_phb    = *IP_CORE_current_phb;
      *p_uzwk           = *IP_CORE_uzwk;
      *p_uzwk_half      = *IP_CORE_uzwk_half;
      *p_motor_winkel   = *IP_CORE_motor_winkel;
      *p_motor_speed    = *IP_CORE_motor_speed;
      *p_uBlaze2peri    = *IP_CORE_transmit_toggle_bit;
      //Umkopieren Reglerparameter usw. in ECAT Register des SyncMan
      *p_motor_off      = *IP_CORE_motor_off;
      *p_reset_overcurrent = *IP_CORE_reset_overcurrent;
      *p_sollwert       = *IP_CORE_sollwert;
      *p_k1             = *IP_CORE_k1;
      *p_k2             = *IP_CORE_k2;
      *frequenz_sollwert = *IP_CORE_frequenz_sollwert;
      *modulationsgrad  = *IP_CORE_modulationsgrad;
      *ramp_up_down     = *IP_CORE_ramp_up_down;
      *max_current      = *IP_CORE_max_current;
      *k1_w             = *IP_CORE_k1_w;
      *k2_w             = *IP_CORE_k2_w;
      *p_uBlaze2peri    |= *IP_CORE_mb2peri;

      // Toggle bit 1 from uBlaze2peri register for end of interrupt
      *p_uBlaze2peri    |= 0x0002;
      *p_uBlaze2peri    &= 0xFFFD;
}

//------------------------------------------------------------------
```

4.4. Entwicklung der EtherCAT Konfigurationsdatei

Um nach der Implementierung des EtherCAT IP Cores und des EtherCAT Slave Sacks die Entwicklung der EtherCAT Teilnehmer fertigzustellen, muss abschließend ein EtherCAT Slave Information File (ESI) für jeden Teilnehmer erstellt werden. Das ESI beinhaltet alle relevanten Informationen (z.B. Name, Inputs, Outputs, SyncManager, FMMUs) eines EtherCAT Teilnehmers und erfüllt zwei wesentliche Aufgaben.

Zum einen greift der ESC nach einem Neustart des Teilnehmers auf diese Konfiguration zurück. Erst nach dieser Initialisierung haben sowohl das PDI als auch der EtherCAT Master (über das Netzwerk) Zugriff auf den Teilnehmer (bzw. auf den physikalischen Speicher), um diesen entsprechend umzukonfigurieren, indem z.B. Interrupts freigeschaltet oder die DCs aktiviert werden. Zum anderen benötigt der EtherCAT Master sämtliche ESIs aller Teilnehmer, um aus diesen das Prozessabbild zu errechnen und den benötigten Frame zu ermitteln. Die Daten der Teilnehmer sowie deren Topologie werden anhand der ESI im sog. EtherCAT Network Information File (ENI) zusammen gefasst. Die Abbildung 4.7 veranschaulicht dieses Prinzip.

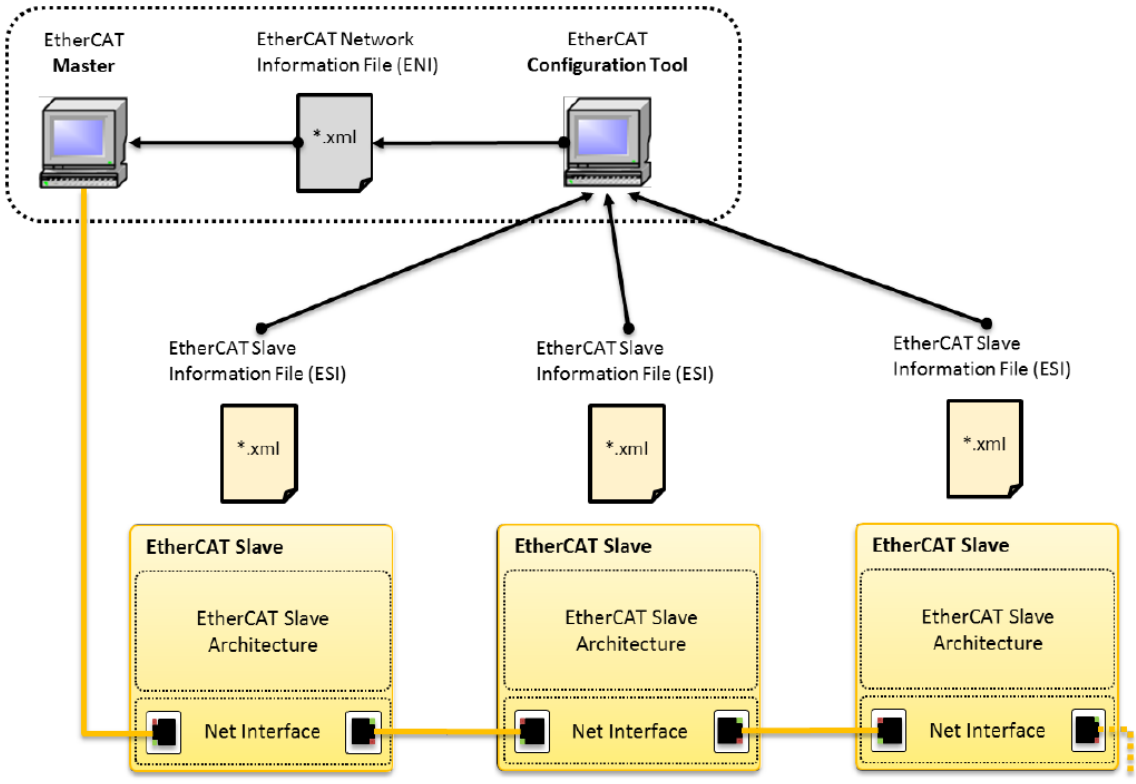


Abbildung 4.7: Basiskonfiguration EtherCAT | Quelle: (EtherCAT Technologie Group, 2013, S. 13 in Section I)

Zum Erstellen der ESIs wird das ebenfalls von der Firma PORT vertriebene EtherCAT Design Tool verwendet. Dieses ist zudem mit dem EtherCAT Slave Stack kompatibel und generiert neben dem benötigten ESI (in XML-Format) auch C-Initialisierungs-Routinen für den Slave Stack. Die wichtigsten Auszüge des ESI sind nachfolgend für das SSB erklärt.

Das ENI wird von der verwendeten Master Software TwinCAT (vgl. Kapitel 4.6 Implementierung des EtherCAT Masters) selbstständig erzeugt.

Beschreibung des Herstellers

Der nachfolgende Auszug bildet den ersten Teil des erstellten ESI ab. Es handelt sich hierbei um die Beschreibung des Vendors (= Entwickler, hier ESW GmbH mit Vendor ID 0x0000062C) und des Teilnehmers. Zudem besteht die Möglichkeit sowohl für den Vendor als auch für die Baugruppe ein hexadezimal codiertes Bild einzufügen, welches den Teilnehmer bspw. bei der Netzwerkkonfiguration mit TwinCAT abbildet. Im unteren Bereich, unter dem Punkt „Device Physics“, wird die physikalische EtherCAT Schnittstelle beschrieben. Der Wert „YY“ codiert hierbei zwei MII (Ethernet) Ports.

```
- <Vendor>
    <Id>#x0000062C</Id>
    <Name>ESW</Name>
    <Comment>ESW-TE1</Comment>
    <ImageData16x14>insert hex coded image here</ImageData16x14>
  </Vendor>
- <Descriptions>
  - <Groups>
    - <Group>
        <Type>SSB - Sensordatenerfassung</Type>
        <Name>IntraLink</Name>
          <ImageData16x14> insert hex coded image he-
          re</ImageData16x14>
      </Group>
    </Groups>
  - <Devices>
    - <Device Physics="YY">  <Type ProductCode="#x1"
        RevisionNo="#x1">SSB Version 2 -witch Slave Stack</Type>
        <Name>SSB</Name>
```

Beschreibung der Datentypen und Standardobjekte

Im Anschluss an die Entwicklerbeschreibung folgen die Beschreibungen der verwendbaren Datentypen, sowie deren Größe. Hier können ggf. eigene Datentypen für die spätere Verwendung erstellt werden.

```
- <DataTypes>
      - <DataType>
            <Name>BOOL</Name>
            <BitSize>1</BitSize>
        </DataType>
      - <DataType>
            <Name>BYTE</Name>
            <BitSize>8</BitSize>
      - <DataType>
```

Die nachfolgenden Standardobjekte gewährleisten, dass jeder beliebige Teilnehmer in ein EtherCAT Netzwerk integriert werden kann, indem in immer gleichen Speicherbereichen dieselben Informationen hinterlegt sind (z.B. erster Eintrag im Identity Objekt ist immer Vendor ID).

```
- <Object>
      <Index>#x1018</Index>
      <Name>Identity Object</Name>
      <Type>DT1018</Type>
      <BitSize>136</BitSize>
    - <Info>
        - <SubItem>
            <Name>number of entries</Name>
              - <Info>
                <DefaultValue>#x4</DefaultValue>
                </Info>
          </SubItem>
            - <SubItem>
              <Name>VendorId</Name>
             - <Info>
                <DefaultValue>#x0000062c</DefaultValue>
              </Info>
            </SubItem>
```

Das Standardobjekt „Identity Object“ liegt im physikalischen Speicherbereich mit der Adresse 0x1018. Es besteht aus 136 Bit und besitzt 4 Subelemente. Eines der Subelemente entspricht wiederum der Vendor ID mit dem Wert 0x0000062C.

Beschreibung der eigenen Objekte

Bei den eigenständig erstellten Objekten handelt es sich um die reservierten Speicherbereiche für die Messdaten des SSBs. Mithilfe des Slave Stacks werden diese Daten in den ESC kopiert. Im ESC wird hierfür folgendes Objekt angelegt.

```
- <Object>
        <Index>#x3000</Index>
        <Name>STROM_PHASE_A</Name>
        <Type>INT</Type>
        <BitSize>16</BitSize>
            - <Info>
                    <DefaultValue>#x0</DefaultValue>
              </Info>
            - <Flags>
                    <Access>rw</Access>
                    <PdoMapping>t</PdoMapping>
              </Flags>
  </Object>
```

Das Objekt zur Aufnahme der Stromwerte der Phase A liegt im Speicherbereich mit der Adresse 0x3000. Zudem besitzt das PDI die Zugriffsberechtigung „rw“, womit sowohl Lese- als auch Schreibzugriffe möglich sind. Da die Daten an das Netzwerk gesendet werden, ist zudem das Flag „t“ (transmit) gesetzt.

Beschreibung der SyncManager und FMMUs

Abschließend werden die SyncManager und FMMUs initialisiert. Da das SSB dem Netzwerk lediglich Messdaten zur Verfügung stellt (aus Sicht des Masters: Inputs) und selbst keine Daten empfängt, wird nur ein SyncManager benötigt. Die physikalische Startadresse beginnt bei 0x1100 und belegt 12 weitere Bytes. Im „Control Byte“ wird das 5. Bit gesetzt, wodurch beim Aktualisieren der Daten ein Interrupt generiert wird. Bezugnehmend auf den SyncManager „Inputs“ wird eine nicht weiter konfigurierbare FMMU eingerichtet.

```
</Profile>
        <Fmmu>Inputs</Fmmu>
          <Sm DefaultSize="12" StartAddress="#x1100"
            ControlByte="#x20" Enable="1">Inputs</Sm>
```

4.5. Implementierung des Reglers

Da bei dem verwendeten Regelungskonzept eine durchgängig digitale Signalverarbeitung vorliegt, ist es von Vorteil, die Regelung der Ständerstromvektoren und Drehzahl im rotorfesten Koordinatensystem auszuführen[31]. Die hierfür benötigten Transformationen sind reine Rechenroutinen. Die Abbildung 4.8 zeigt das angewandte feldorientierte Regelungskonzept des verwendeten Servoantriebes.

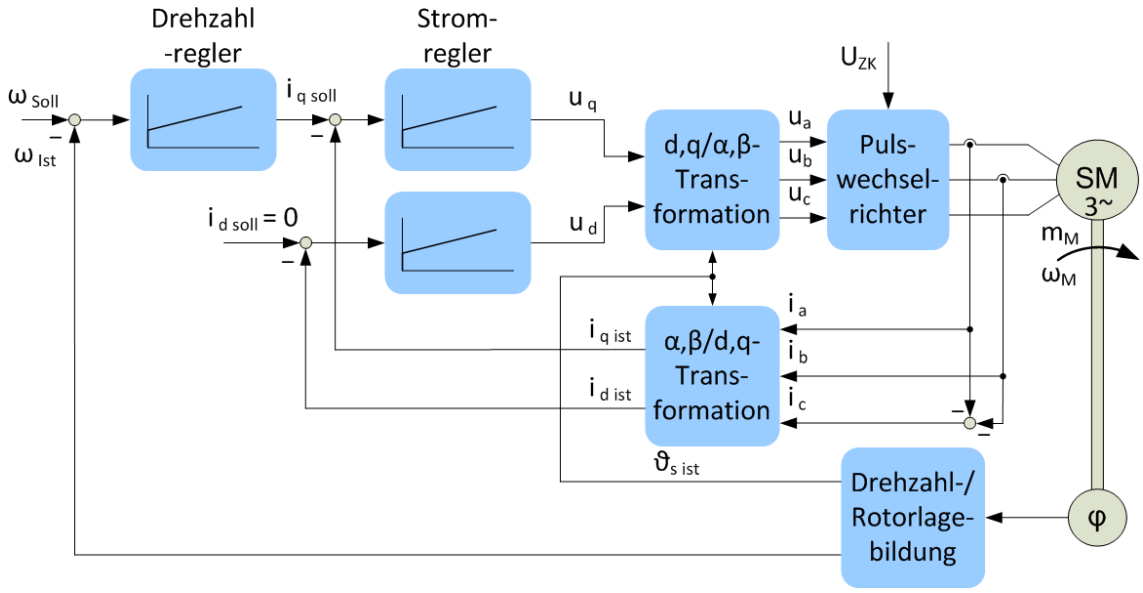


Abbildung 4.8: Regelungskonzept des Servoantriebes | Quelle: Eigene Darstellung in Anlehnung an (Schulze, 2008, S. 82)

Die Soll-Drehzahl ω_{Soll} wird mit der Ist-Drehzahl ω_{Ist} verglichen. Die vorherrschende Regeldifferenz wird an den Drehzahlregler (PI-Regler) weitergegeben. Dieser berechnet den Strom-Soll-Wert $i_{q\,Soll}$ und gibt im Grundstellbereich des Antriebes den Wert $i_{d\,Soll} = 0$ vor ($i_{d\,Soll} \neq 0$ entspricht Feldschwächbereich). Der Stromregler (PI Regler) errechnet hieraus die Komponenten u_q und u_d, die mithilfe der α,β-Transformation in die Wirk- und Blindstromkomponenten u_α und u_β transformiert und anschließend raumzeigermoduliert werden. Zusammen mit dem Pulswechselrichter, der die Aufgabe des mechanischen Kommutators einer Gleichstrommaschine übernimmt, entsteht ein dem Betriebsverhalten einer stromgeregelten Gleichstrommaschine ähnliches Verhalten[32].

[31] Vgl. (Schulze, 2008, S. 81).
[32] Vgl. (Schulze, 2008, S. 81).

Um den Regelkreis zu schließen, erfolgt die Rückführung der Strom-Ist-Werte mithilfe der d,q-Transformation. Neben der Rückführung der Ist-Drehzahl ω_{Ist} wird zudem für das rotorfeste Koordinatensystem die Rotorlage $\vartheta_{s\,ist}$ zurückgeführt.
Entwickelt wurde der Regelalgorithmus seitens der Entwicklungsabteilung der ESW mithilfe der Simulationssoftware Matlab Simulink. Diese ermöglicht einen Datei-Export in die VHDL-Ebene, sodass der Regler gemäß Abbildung 4.6 in den frei programmierbaren Bereich des FPGAs implementiert werden kann.

4.6. Implementierung des EtherCAT Masters

Die Funktionalität des EtherCAT Masters wird im Testaufbau mithilfe der Software TwinCAT (Version 2.11.2226) von der Firma Beckhoff Automation GmbH auf einem Desktop-PC (incl. Netzwerkkarte FC9022) implementiert. Der im TwinCAT integrierte System Manager ermöglicht das Konfigurieren von Netzwerk und Teilnehmern, indem er als Master über die Netzwerkverbindung auf die ESCs der Teilnehmer und deren physikalischen Speicher zugreift. Zusätzlich besteht die Möglichkeit, die im Teilnehmer gespeicherte Konfiguration auszulesen und ggf. zu manipulieren.
Vom Konfigurations-Modus lässt sich der TwinCAT Master in den Echtzeit-Modus schalten, wodurch ein vollwertiger, echtzeitfähiger EtherCAT Master das Initiieren des Frames übernimmt. Der Frame kann hierbei zyklisch, ohne ein lauffähiges SPS Programm oder SPS gesteuert mit einer minimalen Zykluszeit von 50µs angeregt werden. Zykluszeiten kleiner als 4ms werden jedoch nur mit spezieller Hardware (Netzwerkkarten oder Industrie PC) erreicht.
Die Abbildung 4.9 zeigt das Ergebnis eines Netzwerkscans des Testaufbaus. Da das Netzwerk keine Verzweigung aufweist, entspricht die Reihenfolge der Teilnehmer in der Abbildung der tatsächlichen Anordnung der Linientopologie. Unter dem Menüpunkt System Konfiguration werden Echtzeiteinstellungen getätigt, SPS Programme können unter dem Menüpunkt SPS Konfiguration angefügt und initialisiert werden. Eine umfangreiche Beschreibung zum Umgang mit TwinCAT ist dem Anhang 3 zu entnehmen.

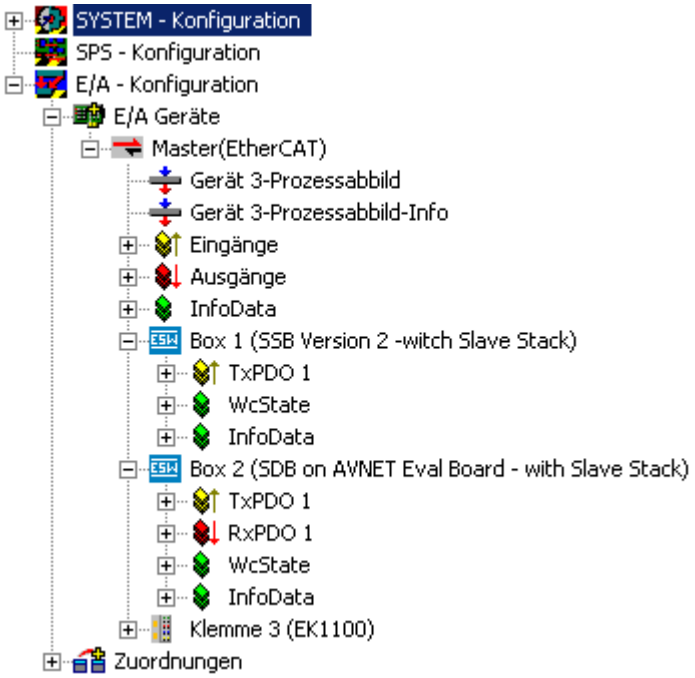


Abbildung 4.9: Screenshot TwinCAT nach System Scan

Nach dem erfolgreichen Netzwerkscan sind sämtliche Teilkomponenten des Antriebssystems vollständig implementiert. Der entwickelte Testaufbau ist betriebsbereit, eine Zykluszeit von 100µs ist eingestellt.

5. Inbetriebnahme des Testsystems

5.1. Inbetriebnahme des SSBs

Zur Inbetriebnahme des SSBs gilt es, den unter Kapitel 3.2.3 erarbeiteten Signalverlauf mithilfe der zur Verfügung stehenden Möglichkeiten zu initiieren. Das Startsignal zur Messdatenerfassung bzw. zum Starten der AD-Wandler wird durch das Aktivieren des SYNC 0 Signals realisiert. Die Abbildung 5.1 zeigt die hierfür vorgenommene Konfiguration.

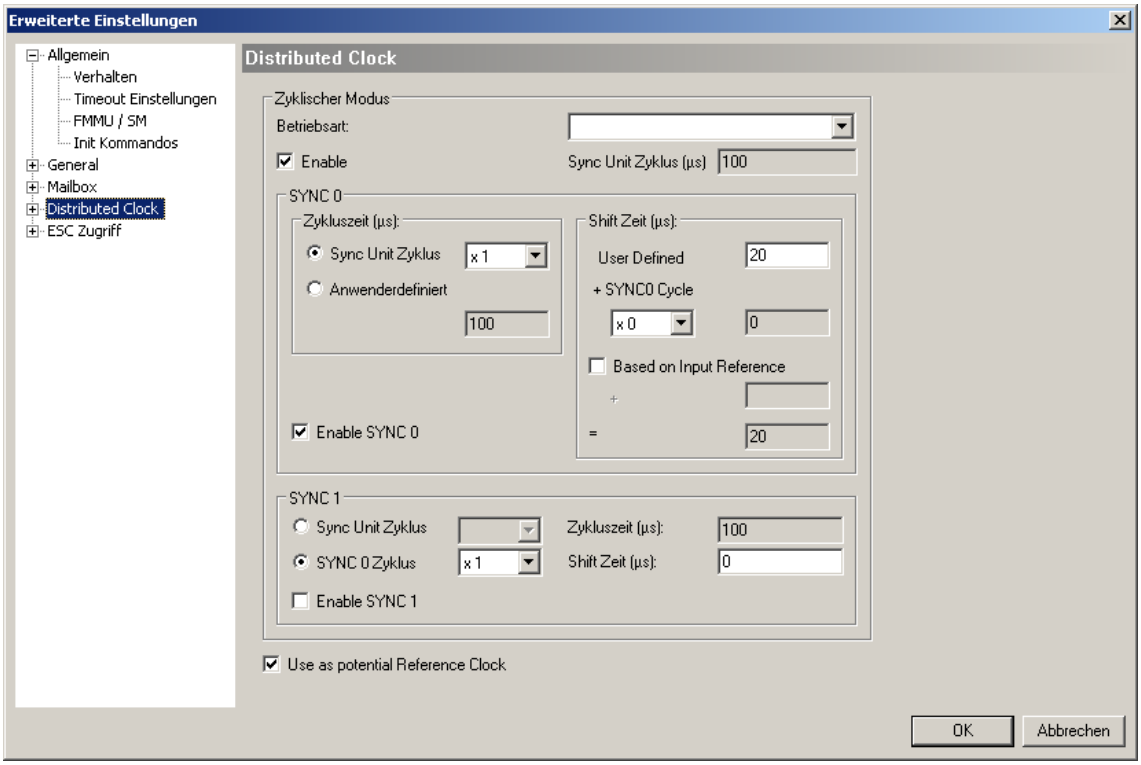
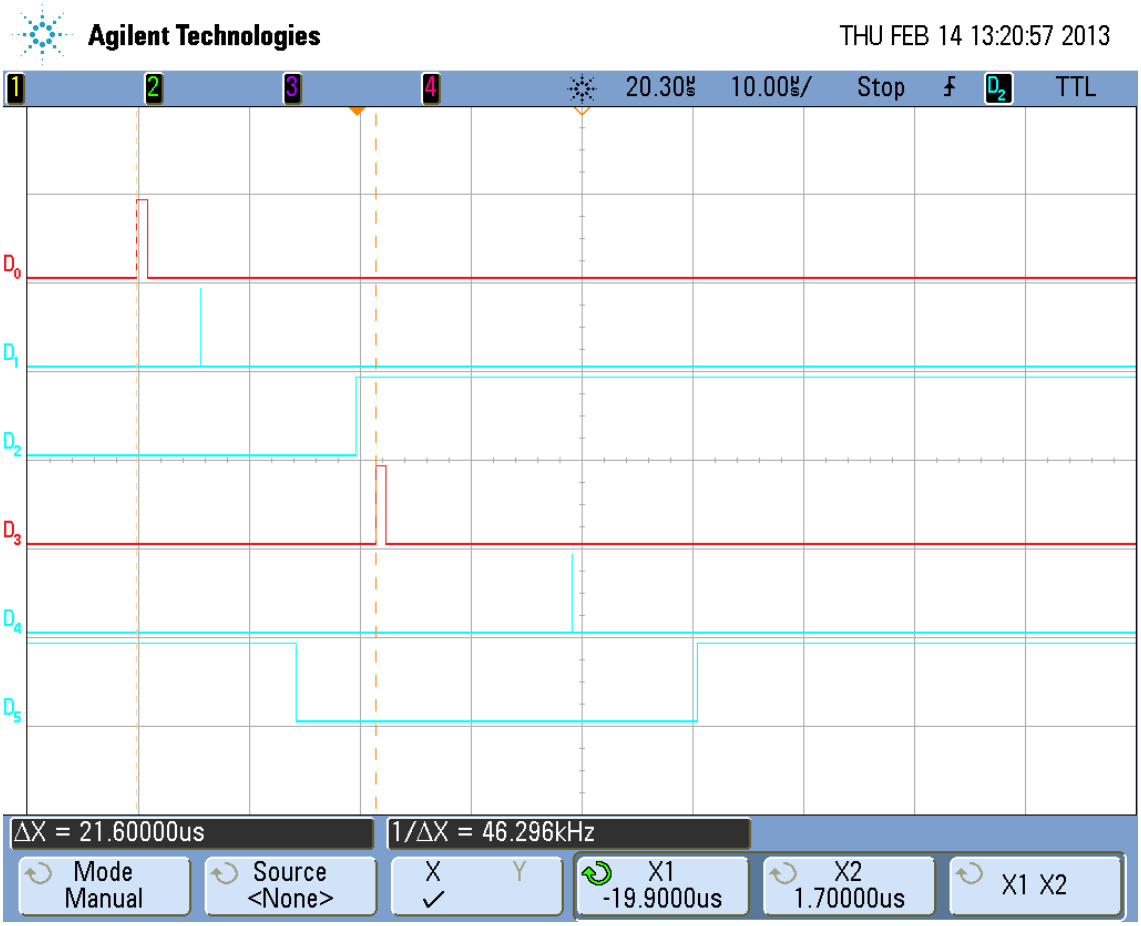


Abbildung 5.1: Screenshot TwinCAT mit SSB DC-Konfiguration

Die ausgewählte User Shift Time von 20 µs wird unter Berücksichtigung der in Kapitel 3.2.3 angestellten Überlegungen empirisch ermittelt. Dies ist nötig, da das SYNC 0 Signal keinen erkennbaren zeitlichen Bezug zum SOF aufweist. Das SYNC 0 Signal wird folglich iterativ solange vor das SOF geschoben, bis eine ausreichende Vorhaltezeit eingestellt ist. Da das SSB der erste Teilnehmer des Netzwerkes ist, wird es zudem als Referenzuhr verwendet (Use as potential Reference Clock ist aktiviert).

Durch das Auslösen der AD-Wandler mithilfe des SYNC 0 Signals (D0) ergibt sich der in der Abbildung 5.2 oszillographierte Signalverlauf.



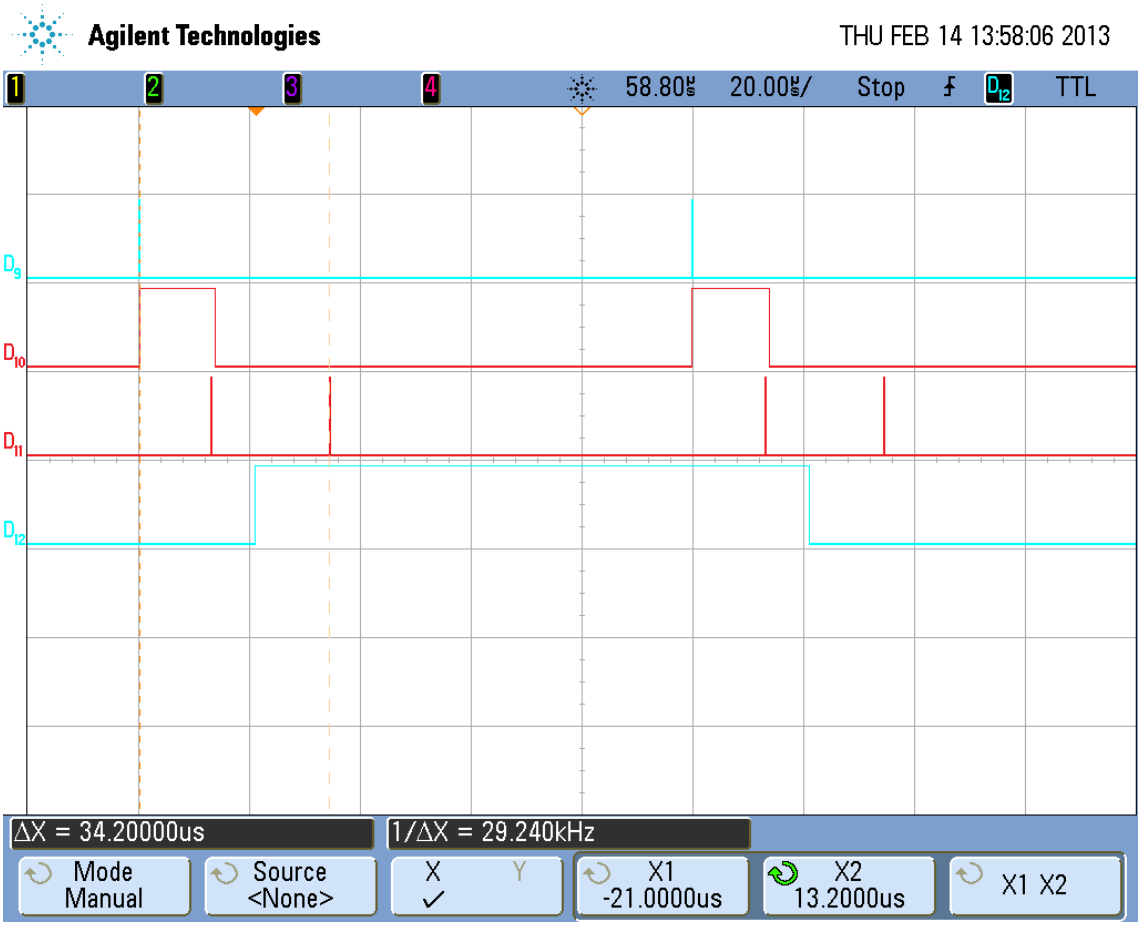
D0	SYNC 0 Signal zum Starten der AD-Wandler
D1	AD-Wandlung abgeschlossen, Interrupt für Microblaze
D2	Prüf-Bit - wechselt Status in jedem Zyklus
D3	Interrupt - Routine zum Umkopieren abgeschlossen
D4	SOF - Start of Frame
D5	Interrupt des SyncManagers - fallende Flanke: Zugriff von PDI auf SyncManager beginnt - steigende Flanke: Zugriff von ECAT beendet

Abbildung 5.2: Messung Signalverlauf SSB

Aufgrund der ausreichend dimensionierten Vorhaltezeit ist das Umkopieren der AD-gewandelten Messwerte abgeschlossen (D3), bevor ein neuer Frame (D4) das SSB erreicht. Dies ist auch der Fall bei einem Jitter des SOFs von ±10% der Zykluszeit (entspricht ±10µs).

5.2. Inbetriebnahme des SDBs

Beim SDB wurde das Starten der Interrupt-Routine mithilfe des SyncManager Interrupts initiiert. Es ergibt sich der anhand der Abbildung 5.3 ersichtliche Signalverlauf. Die Interrupt-Routine ist in diesem Fall mithilfe zweier Peaks (D11) eingerahmt.



D9	EOF – End of Frame
D10	Interrupt des SyncManagers - steigende Flanke: Zugriff von ECAT beendet - fallende Flanke: Zugriff von PDI auf SyncManager beginnt
D11	Beginn und Ende der Interrupt-Routine zum Umkopieren
D12	Empfangenes Prüf-Bit

Abbildung 5.3: Messung Signalverlauf SDB

Unmittelbar nach dem Ende des Frames (D9) sind die Messdaten auf dem SDB mittels EtherCAT eingetroffen (Steigende Flanke D10). Der hierdurch generierte Interrupt ruft die Interrupt-Routine (D11) auf, die die Messdaten in den Registerbereich des Reglers umkopiert. Das hierbei deutlich zu erkennende Delay zwischen dem Interrupt des

SyncManagers (D10) und dem Start der Interrupt-Routine (D11) ist der Reaktionszeit des MicroBlaze geschuldet. Das Beenden der Interrupt-Routine startet wiederum den Regelalgorithmus.

5.3. Slave-to-Slave Kommunikation

Um die Messdaten möglichst schnell vom SSB zum SDB zu transferieren, wurde das Zugriffsverfahren bei diesen beiden Teilnehmern von Master-Slave auf Slave-to-Slave geändert. Da hierdurch einige Sicherungsmechanismen des Protokolls aufgehoben werden, muss diese Konfiguration manuell vorgenommen werden. Hierfür wird anhand der FMMU des SDBs dessen logischer Adressbereich notiert. Die Abbildung 5.4 zeigt, im oberen Bereich, die aktuelle FMMU-Konfiguration des SDBs.

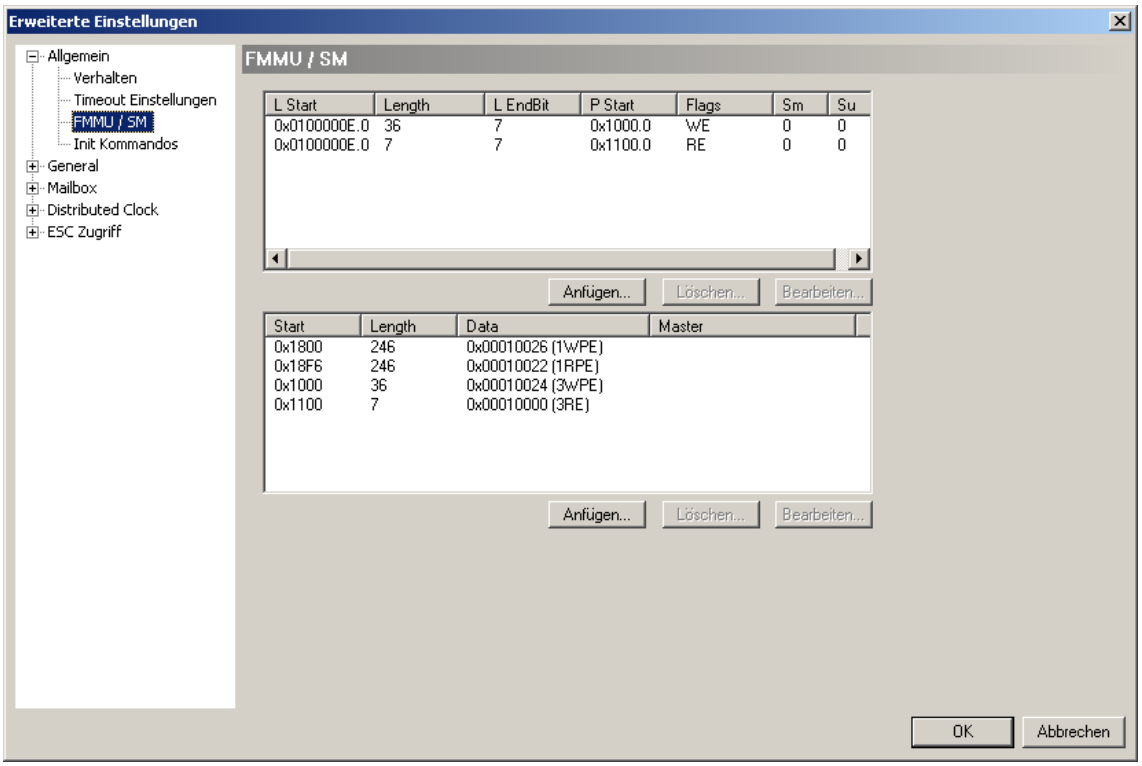


Abbildung 5.4: Screenshot TwinCAT mit FMMU-Konfiguration des SDBs

Die für das weitere Vorgehen relevanten und beschreibbaren Daten liegen im logischen Adressbereich 0x0100000E (Adressbereich zur Aufnahme der Sensordaten) des Prozessabbildes.

Um die Slave-to-Slave Kommunikation der Teilnehmer zu realisieren, wird im ESC des SSBs eine zusätzliche FMMU eingerichtet, die die Sensordaten des SSBs in den logi-

schen Adressbereich des SDBs kopiert. Die hierfür vorgenommene Konfiguration ist der Abbildung 5.5 zu entnehmen.

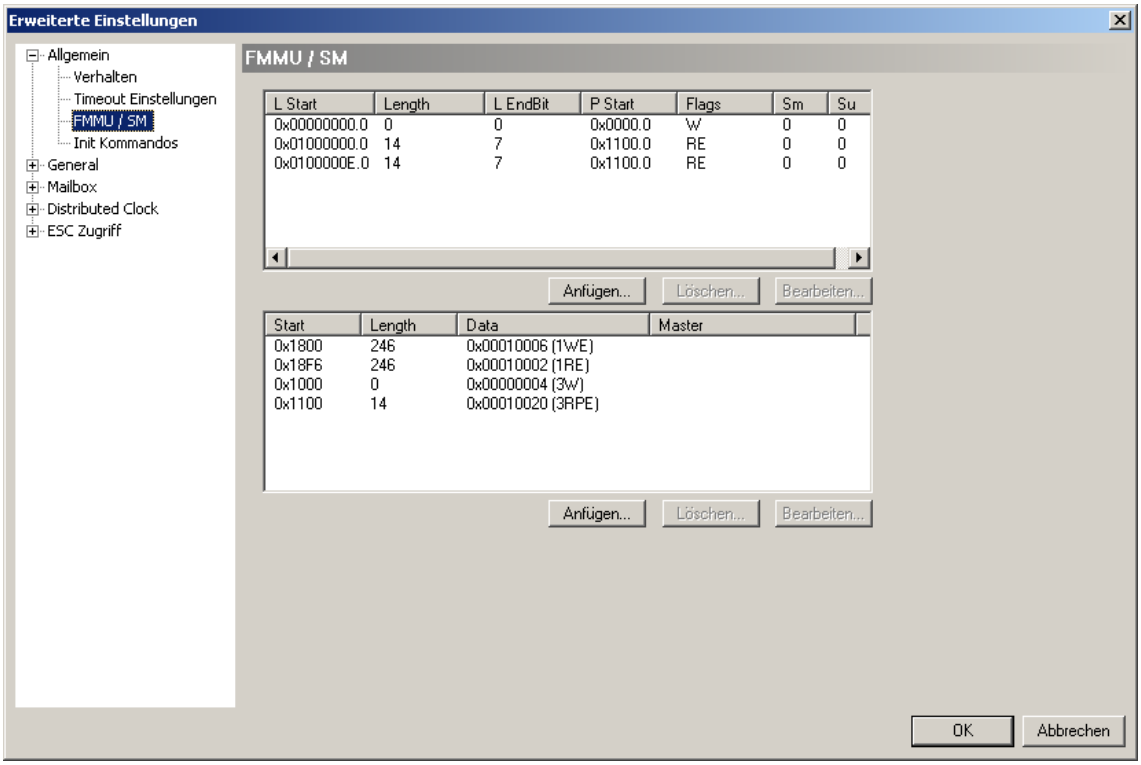
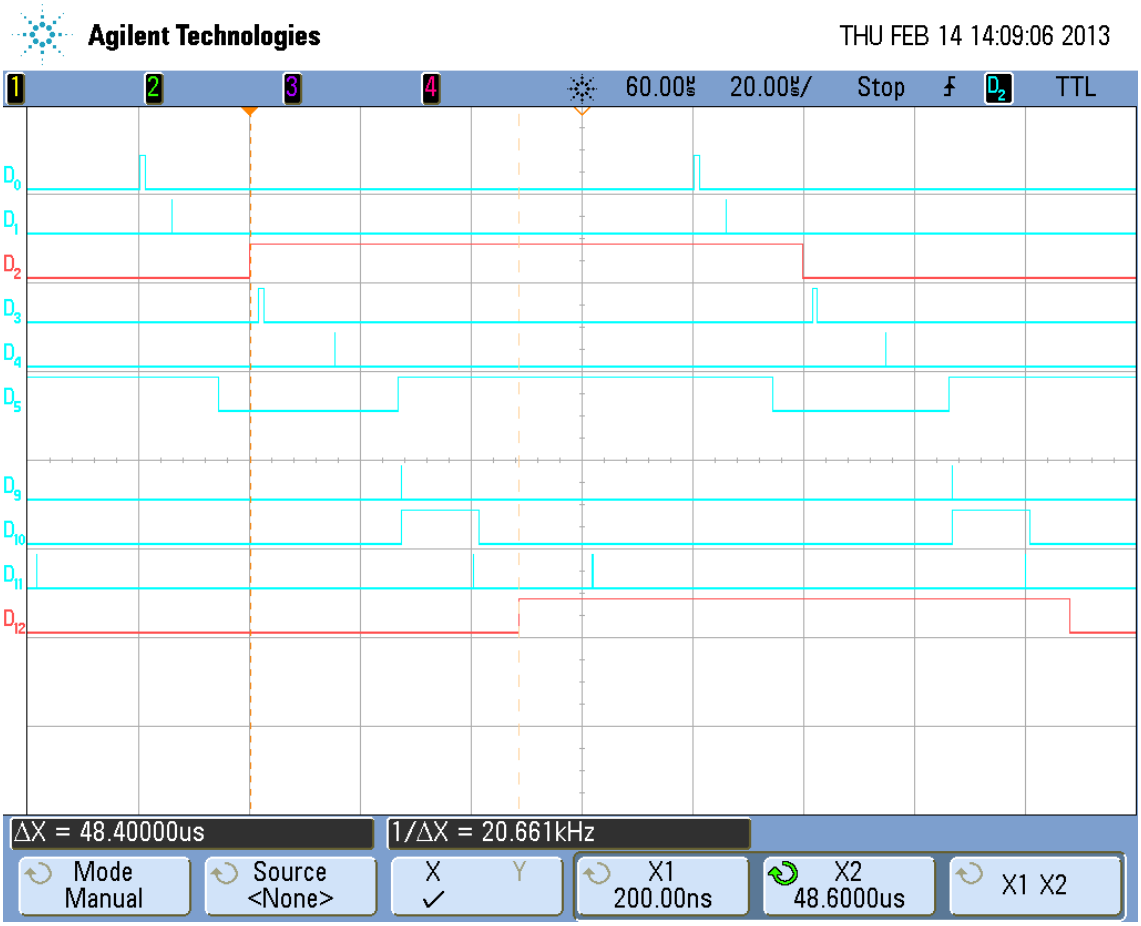


Abbildung 5.5: Screenshot TwinCAT mit zusätzlicher FMMU im SSB

Die Sensordaten des SSBs werden hierdurch sowohl in den automatisch generierten logischen Adressbereich 0x01000000 sowie in den ebenfalls vom SDB genutzten logischen Adressbereich 0x0100000E kopiert. Die Daten des SSBs stehen folglich auch dem SDB zur Verfügung, die Slave to Slave Kommunikation ist hergestellt.

Zur Verifikation der vorgenommenen Konfiguration dient die Abbildung 5.6, in der die Übertragung eines Prüf-Bits messtechnisch erfasst ist. Das Prüf-Bit (D2) wird SSB-seitig bei jedem Aufruf der Interrupt-Routine getoggelt und anschließend übertragen. Zusätzlich ist das Bit an einen Test-Pin geführt. SDB-seitig wird das empfangene Prüf-Bit (D12) ebenfalls an einen Test-Pin geführt. Die Dauer der Datenübertragung von SSB zu SDB kann folglich mit ca. 50µs angegeben werden.

Um das für die Regelung zu kompensierende Messdatenalter zu bestimmen, ist der Datenübertragungsdauer noch eine Zeit von ca. 20µs aufzuaddieren (Dauer AD-Wandlung + Umkopieren der Register). Dieses Delay wird nachträglich im Regelalgorithmus implementiert und somit von der Regelung kompensiert.



D0	SYNC 0 Signal zum Starten der AD-Wandler
D1	AD-Wandlung abgeschlossen, Interrupt für Microblaze
D2	Prüf-Bit - wechselt Status in jedem Zyklus
D3	Interrupt - Routine zum Umkopieren abgeschlossen
D4	SOF - Start of Frame
D5	Interrupt des SyncManagers - fallende Flanke: Zugriff von PDI auf SyncManager beginnt - steigende Flanke: Zugriff von ECAT beendet
D9	EOF – End of Frame
D10	Interrupt des SyncManagers - steigende Flanke: Zugriff von ECAT beendet - fallende Flanke: Zugriff von PDI auf SyncManager beginnt
D11	Beginn und Ende der Interrupt-Routine zum Umkopieren
D12	Empfangenes Prüf-Bit

Abbildung 5.6: Messung der Übertragung eines Prüf-Bits

5.4. Parametrierung des Reglers

Die Inbetriebnahme des Testaufbaus wird mit der Parametrierung des Reglers abgeschlossen. Die Regelparameter wurden als über das Netzwerk beschreibbare Variablen vorgesehen, sodass eine empirische Ermittlung geeigneter Parameter mithilfe des TwinCAT System Managers möglich ist.

Die Abbildung 5.7 zeigt die fertig beschriebenen Parameter. K1 und K2 sind hierbei P- und I-Anteil des PI-Stromreglers. K1_W und K2_W beinhalten den P- und I-Anteil des PI-Drehzahlreglers. Der SOLLWERT gibt die Soll-Drehzahl vor, der Parameter RAMP_UP_DOWN beschreibt die Steilheit der Anfahr-Rampe der Soll-Drehzahl. Zusätzlich kann der maximal mögliche Strom über den Parameter MAX_CURRENT begrenzt werden. Sämtliche Parameter werden intern skaliert, sodass bspw. der Wert 512_{Dez} des Parameters MAX_CURRENT einem maximalen Strom von 1A entspricht.

Name	Online	Typ	Größe	>Adre...	Ein/Aus
IST_BLINDSTROM	0xFFFF (-1)	INT	2.0	72.0	Eingang
IST_WIRKSTROM	0xFFFC (-4)	INT	2.0	74.0	Eingang
OVERCURRENT_...	0x00 (0)	USINT	1.0	76.0	Eingang
OVERCURRENT_...	0x00 (0)	USINT	1.0	77.0	Eingang
ERROR_SIGNAL	0x00 (0)	USINT	1.0	78.0	Eingang
WcState	0	BOOL	0.1	1522.2	Eingang
InputToggle	0	BOOL	0.1	1524.2	Eingang
State	0x0008 (8)	UINT	2.0	1566.0	Eingang
AdsAddr	0A 74 03 F2 04 01 ...	AMSADDRESS	8.0	1568.0	Eingang
DcOutputShift	0x000107AC (67500)	DINT	4.0	1576.0	Eingang
DcInputShift	0x00020594 (132500)	DINT	4.0	1580.0	Eingang
IST_STROM_PHA...	0x0000 (0)	INT	2.0	72.0	Ausg...
IST_STROM_PHA...	0x0000 (0)	INT	2.0	74.0	Ausg...
IST_SPANNUNG_...	0x0000 (0)	INT	2.0	76.0	Ausg...
IST_SPANNUNG_...	0x0000 (0)	INT	2.0	78.0	Ausg...
IST_WINKEL	0x0000 (0)	UINT	2.0	80.0	Ausg...
IST_GESCHWIND...	0x0000 (0)	INT	2.0	82.0	Ausg...
IST_TRANSMIT_...	0x0000 (0)	INT	2.0	84.0	Ausg...
MOTOR_EN	0x01 (1)	USINT	1.0	86.0	Ausg...
RESET_OVERCU...	0x00 (0)	USINT	1.0	87.0	Ausg...
SOLLWERT	0x0111 (273)	UINT	2.0	88.0	Ausg...
K1	0xACC1 (44225)	UINT	2.0	90.0	Ausg...
K2	0x7E64 (32356)	UINT	2.0	92.0	Ausg...
FREQUENZ_SOLL...	0x0000 (0)	UINT	2.0	94.0	Ausg...
MODULATIONSG...	0x0000 (0)	UINT	2.0	96.0	Ausg...
RAMP_UP_DOWN	0x0050 (80)	UINT	2.0	98.0	Ausg...
MAX_CURRENT	0x0200 (512)	UINT	2.0	100.0	Ausg...
K1_W	0x0FFF (4095)	UINT	2.0	102.0	Ausg...
K2_W	0x8000 (32768)	UINT	2.0	104.0	Ausg...
UBLAZE2PERI	0x0000 (0)	UINT	2.0	106.0	Ausg...

Abbildung 5.7: Screenshot TwinCAT der E/A-Daten des SDBs incl. Regelparameter

6. Auswertung und Ausblick

Die Inbetriebnahme verdeutlicht, dass das unter Kapitel 3.3 erarbeitete, EtherCAT-basierende Lösungskonzept nahezu alle Anforderungen an ein modernes Antriebssystem erfüllt. Der Gedanke der Modularisierung sowie die allgemeingültigen Grundlagen konnten sowohl bei der Entwicklung, als auch bei der Implementierung umgesetzt werden. Die zuvor in der Baugruppe Umrichter verschachtelten Teilkomponenten der Regelung und Sensorik konnten erfolgreich zu modularen Einzelkomponenten eines flexiblen Gesamtsystems extrahiert werden. Durch die Verwendung der digitalen Ausgangsklemme wurde zudem ein Teil der Sicherheitsfunktion für den Master zugänglich. Im Fall einer genormten Schnittstellendefinition zwischen den Teilnehmern auf Software-Ebene ließe sich jedes Teilelement auch herstellerunabhängig ersetzen, eine Modularisierung bis auf Regelungs-/Steuerungsebene wäre künftig realisierbar.

Zudem konnte die hohe Performance des Bussystems EtherCAT zumindest teilweise verdeutlicht werden. Der Drehzahl-Regelkreis mit unterlagertem Strom-Regler wurde via EtherCAT geschlossen. Die Datenübertragung erfolgt hierbei zyklisch alle 100µs, wodurch ein nahezu quasi-analoges Regelungsverhalten entsteht. Die Zykluszeit wird hierbei nicht durch EtherCAT, sondern der verwendeten Hardware limitiert. Durch den verhältnismäßig niedrig taktenden MicroBlaze (75MHz) entsteht sowohl auf dem SSB, als auch auf dem SDB ein hohes zeitliches Delay (SSB ca. 16µs; SDB ca. 34µs) aufgrund des Umkopier-Vorgangs der Sende-/Empfangsdaten.

Auf Basis des Buszugriffsverfahrens von EtherCAT, musste das bestehende System um einen Master erweitert werden. Durch diesen konnten die üblicherweise feststehenden Parameter der Regler als beschreibbare Variablen ausgeführt und folglich im laufenden Betrieb manipulierbar gestaltet werden. Grundeinstellungen sind zudem mithilfe der XML-Konfigurationsdateien auch permanent auf einem Teilnehmer speicherbar.

Der konzipierte Testaufbau konnte ebenfalls den gestellten Anforderungen gerecht werden und bietet zudem noch Platz für zukünftige Erweiterungen. Durch die Verwendung von industriellen Durchgangsklemmen ist ein rascher Umbau möglich und ein hohes Maß an Flexibilität und Übersichtlichkeit gegeben. Eine Verwendung als Demonstrator ist prinzipiell möglich, jedoch bedarf es hierfür einiger abschließender kon-

struktiver Anpassungen. Eine vor Berührung schützende Ober-Konstruktion sowie die Adaption auf eine mobile Unter-Konstruktion konnten aufgrund mangelnder Zeit nicht umgesetzt werden.

Für die ESW GmbH lassen sich anhand der angestellten Überlegungen und Ausführungen zwei wesentliche Ergebnisse extrahieren. Zum einen konnte eine bislang nicht verwendete Technologie komplett erschlossen und dokumentiert werden. Anhand des Testaufbaus ist ein kompletter Entwicklungszyklus, von der Markt-Recherche über die Entwicklung, bis hin zur Implementierung und Inbetriebnahme abgearbeitet worden. Sämtliche Ergebnisse und Erkenntnisse wurden nachvollziehbar dokumentiert und in ein funktionsfähiges und auf andere Plattformen portierbares Design zusammengeführt. Die Grundlage zur Verwendung von EtherCAT als Standardschnittstelle ist somit vorhanden und eine mögliche Integration in zukünftige Projekte gegeben.

Das zweite relevante Ergebnis besteht in der praktischen Umsetzung der theoretisch erarbeiteten Grundlagen. Durch die Adaption auf einen real verfügbaren Test-Aufbau steht der ESW GmbH ein komplettes EtherCAT Netzwerk zur Verfügung. Mit Hinblick auf zukünftige Applikationen bietet sich somit die Möglichkeit, diverse Erprobungen und deren Evaluierungen am realen System durchzuführen.

Zur Weiterführung des Projektes könnte zunächst die Optimierung der entwickelten Teilkomponenten weiter voran getrieben werden. Auf Seiten des SSBs ließe sich die ermittelte Zeit von 21,6µs zwischen dem Auslösen der AD-Wandlung und dem Ende der Interrupt-Routine maßgeblich reduzieren, indem auf die Verwendung des MicroBlaze verzichtet wird. Hierfür müssen die PDI-seitig eingestellten Konfigurationen von der XML-Konfigurationsdatei vorgenommen werden. Den Vorgang des Umkopierens der Daten kann der FPGA durchführen.

Da das SDB, durch das Trennen von zeitkritischen und nicht zeitkritischen Daten, den MicroBlaze in einem größeren Umfang nutzt, könnte hier dessen Anbindung an den EtherCAT IP Core von PLB (75MHz) durch AXI (100MHz) ersetzt werden. Zudem besteht die Möglichkeit, die Daten der Interrupt-Routine durch die Verwendung eines weiteren SyncManagers zu reduzieren.

Darüber hinaus werden die Timings aller Teilnehmer im großen Maße vom Jitter des Masters beeinflusst. Der Jitter konnte durch die Verwendung einer speziellen Netz-

werkkarte zwar reduziert werden, befindet sich jedoch noch immer im Mikrosekunden-Bereich. Optimierungspotenzial besteht hier durch die Entwicklung eines eigenen Embedded Masters, der die EtherCAT-Technologie somit auch für mobile Zielapplikationen anwendbar macht. Um ein Optimum an Performance bei geringsten Materialkosten zu erhalten wäre es möglich, einen solchen Embedded Master bspw. zusammen mit der SSB-Funktionalität zu implementieren. Hierdurch können zwei Funktionen zusammengeführt werden, sodass ein vergleichbares System bei geringerem Hardware-Einsatz entsteht. Die erarbeiteten Systemgrenzen von Prozess, Sensorik und Regelung blieben hierbei weiterhin bestehen.

Für weitere Performance-Untersuchungen, insbesondere in Bezug auf die Tauglichkeit von EtherCAT zur Echtzeit-Regelung dynamischer Prozesse, kann der entwickelte Test-Aufbau unter Last, bei unterschiedlichen Betriebsbedingungen erprobt werden. Hierbei ist es möglich den Regelkreis der Last ebenfalls über dasselbe Bussystem zu schließen, wodurch die Option einer „virtuellen Kopplung“ von Last und Antrieb gegeben wäre.

Als Perspektive für die ESW GmbH bietet sich eine Vielzahl unterschiedlichster Einsatzmöglichkeiten für die neu erarbeitete Technologie. Das Prinzip eines Servoantriebes mit verteilten Komponenten lässt sich überall dort anwenden, wo lokale Probleme mit dem verfügbaren Bauraum vorhanden sind. Die Messeinrichtung und Regelung kann räumlich getrennt verbaut werden, wodurch unter Umständen auch geringere Anforderungen an EMV eingehalten werden müssen. Die Baugröße eines solchen Umrichters könnte sich hierdurch reduzieren.

Durch die Verwendung einer standardisierten Schnittstelle auf Basis von EtherCAT besteht zudem die Möglichkeit, die Kompatibilität mit weiteren Baugruppen (ESW-intern, Jenoptik-intern und Fremd-Produkten) zu erhöhen. Die übergeordnete Kommunikation mittels EtherCAT bspw. zwischen ESW-Energiesystemen und ESW-Antriebssystemen kann hierbei die Grundlage zu einem intelligenten und hochperformanten Gesamtsystem bilden. Die kostenintensive Eigenentwicklung unterschiedlicher Bauteile kann durch die Möglichkeit des Zukaufens von Standardkomponenten reduziert werden. Die von der ESW GmbH selbst vertriebenen Produkte werden hingegen durch die Anbindung mittels eines gängigen Industriestandards für den Markt noch interessanter.

Literaturverzeichnis

AVNET Electronics. (2013, 02 11). *AVNET Electronics Marketing*. Retrieved from http://www.em.avnet.com/en-us/design/drc/Pages/ISM-Networking-FMC-Module.aspx

Beckhoff Automation. (2013, 02 16). *Beckhoff Information System*. Retrieved from http://www.beckhoff.de/default.asp?ethercat/el2004.htm

Beckhoff Automation. (2013, 02 16). *Beckhoff Information System*. Retrieved from http://www.beckhoff.de/default.asp?pc_cards_switches/fc9002_fc9004.htm

Beckhoff Automation. (2013, 02 15). *Beckhoff Infosystem*. Retrieved from http://www.beckhoff.de/default.asp?twincat/default.htm

Beckhoff Automation GmbH. (2013, 02 04). *Beckhoff Information System*. Retrieved from http://tcinfosys.beckhoff.de/index.php?content=../content/1031/ethernetcabling/html/bt_ec_cabling_design.htm&id=5774

Beckhoff Automation GmbH. (2013, 02 09). *Hardware Data Sheet - EtherCAT Slave Controller IP Core for Xilinx FPGAs.* Verl. Retrieved from http://www.beckhoff.de/default.asp?download/ethercat_development_products.htm

Biere, A., Kroening, D., Weissenbacher, G., & Wintersteiger, C. (2008). *Digitaltechnik - Eine praxisnahe Einführung.* Heidelberg: Springer Verlag.

Elektro-Automatik. (n.d.). *Datenblatt: EA-PS 8000 T 320W - 1500W Labornetzgeräte.*

EPSG. (2012, 08 20). *Ethernet POWERLINK*. Retrieved from http://www.ethernet-powerlink.org/

ESW GmbH. (n.d.). *Datenblatt (intern) Synchronmotor.*

ESW GmbH. (n.d.). *Datenblatt (intern) Umrichter.*

EtherCAT Technologie Group. (2012, 08 19). *EtherCAT*. Retrieved from http://www.ethercat.org/

EtherCAT Technologie Group. (2013, 02 06). *EtherCAT.* Retrieved from http://www.ethercat.de/pdf/german/EtherCAT_Implementierung.pdf

EtherCAT Technologie Group. (2013, 02 07). *EtherCAT.* Retrieved from http://www.ethercat.de/pdf/ethercat_d.pdf

EtherCAT Technologie Group. (11. 02 2013). *EtherCAT.* Von http://www.ethercat.org/pdf/english/ETG2200_V2i0i1_SlaveImplementationGuide.pdf abgerufen

Fuest, K., & Döring, P. (2007). *Elektrische Maschinen und Antriebe* (7. ed.). Wiesbaden: Vieweg & Sohn Verlag.

Giersch, H.-U., Harthus, H., & Vogelsang, N. (2003). *Elektrische Maschinen* (5 ed.). Wiesbaden: GWV Fachverlage GmbH.

Gizopoulos, D., Paschalis, A., & Zorian, Y. (2004). *Embedded Processor-Based Self-Test.* Dordrecht, Niederlande: Kluwer Academic Publishers.

Griesenbruch, M. (2013, 01 09). *Marktstudien der Automatisierungstechnik.* Retrieved from http://www.marktstudien.org/marktstudien/ergebnisauszug-servoantriebe.pdf

Group EtherCAT Technologie. (2013, 02 06). *EtherCAT*. Retrieved from http://www.ethercat.org/

HMS Industrial Network. (2013, 02 03). *Feldbusse*. Retrieved from http://www.feldbusse.de/Vergleich/vergleich_ethernet.shtml

Kraftfahrt-Bundesamt. (2013, 01 09). *Internetpräsenz des Kraftfahrt-Bundesamt Deutschlands*. Retrieved from http://www.kba.de/nn_125398/DE/Statistik/Fahrzeuge/Bestand/2012__b__jahresbilanz.html

Leonhard, W. (2000). *Regelung elektrischer Antriebe.* Heidelberg: Springer-Verlag.

Modbus Organization, Inc. (2012, 08 20). *Modbus*. Retrieved from http://www.modbus.org/

ODVA. (2012, 08 20). *Ethernet IP*. Retrieved from http://www.ethernetip.de/

PI. (2012, 08 20). *PROFIBUS | PROFINET*. Retrieved from http://www.profibus.com/

Quest Trend Magazine. (2012, 08 21). *Quest Trend Magazine Online*. Retrieved from http://www.quest-trendmagazin.de/The-market-shares-of-Ethernet.143.0.html?&L=1

Roddeck, W. (2012). *Einführung in die Mechatronik* (4. ed.). Heidelberg: Vieweg + Teubner Verlag.

Schröder, D. (2009). *Elektrische Antriebe - Regelung von Antriebssystemen* (3. ed.). Berlin: Springer-Verlag.

Schulze, M. (2008). *Elektrische Servoantriebe.* München: Fachbuchverlag Leipzig.

Sercos International. (2012, 08 20). *SERCOS*. Retrieved from http://www.sercos.de/

Valvana, J. W. (2011). *Embedded Microcomputer Systems - Real Time Interfacing.* Stamford, USA: Cengage Learning.

Xilinx. (2013, 02 10). *Xilinx All Programmable*. Retrieved from http://www.xilinx.com/tools/microblaze.htm

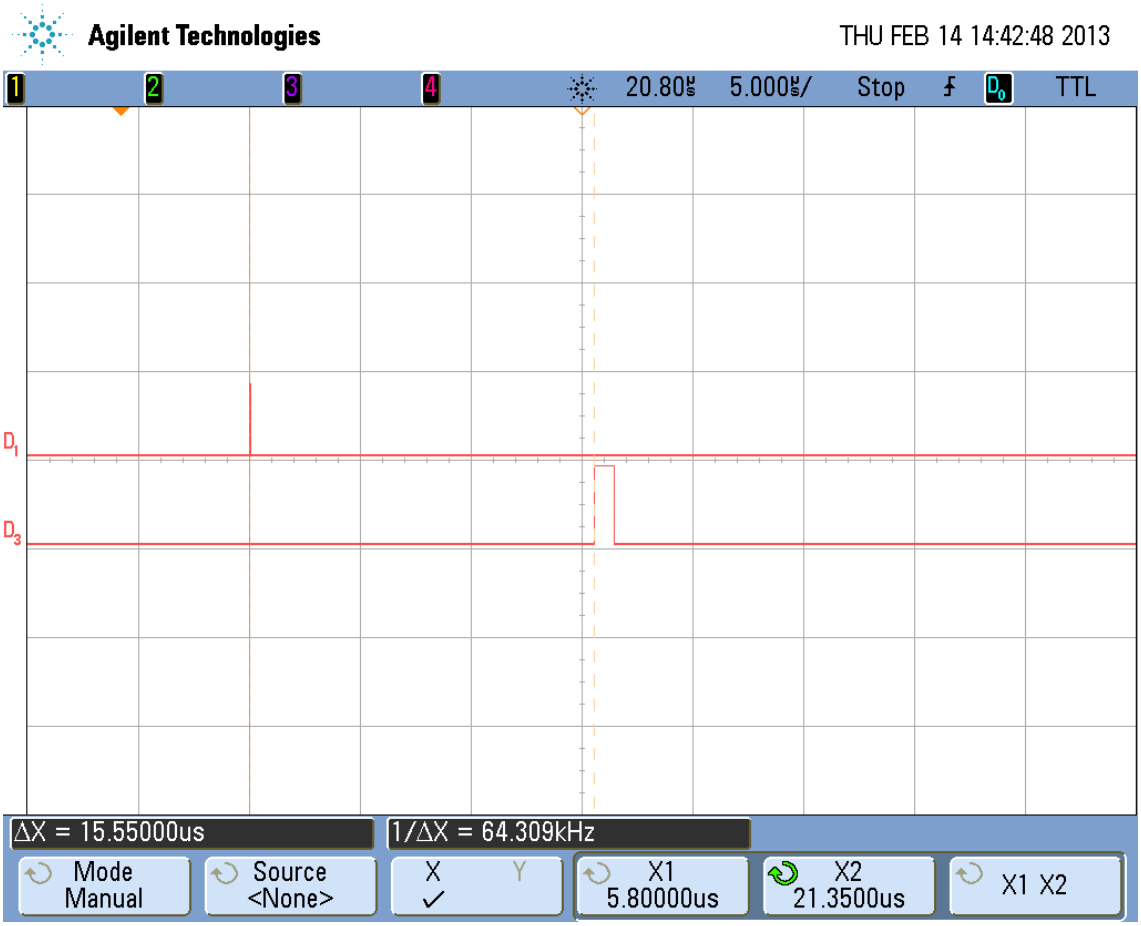
Xilinx. (2013, 02 11). *Xilinx All Programmable*. Retrieved from http://www.xilinx.com/products/boards-and-kits/HW-FMC-XM105-G.htm

Xilinx. (2013, 02 11). *Xilinx All Programmable*. Retrieved from http://www.xilinx.com/products/boards-and-kits/AES-S6DEV-LX150T-G.htm

Anhangverzeichnis

Anhang 1: Ermittlung der Zeiten t_{ADC} und t_{copy} auf dem SSB

Die Zeite t_{copy} konnte mithilfe eines Oszilloskops messtechnisch erfasst werden. Hierfür wurde sowohl das Ready-Signal der AD-Wandlung als auch ein Prüf-Bit, welches am Ende der Umkopier-Routine für einen Takt den Zustand 1 (3,3 Volt) annimmt, an sog. Test-Pins nach außen geführt. Die Messung stellt sich wie folgt dar.



Zu dem messtechnisch erfassten Δt von 15,5µs addiert sich die eigentliche Zeit der AD-Wandlung. Diese lässt sich wie folgt berechnen:

Frequenz der AD-Wandler = 50MHz = 50 1/s

Anzahl der Takte von ADC-go bis ADC-ready = 287

Anzahl der Takte für Offsetabgleich = 3

$$(287 + 3) * \frac{1}{50\frac{1}{s}} = 5{,}8\mu s$$

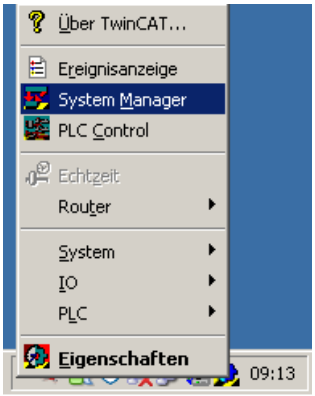
Anhang 2: Pin-Belegung Durchgangsklemmen

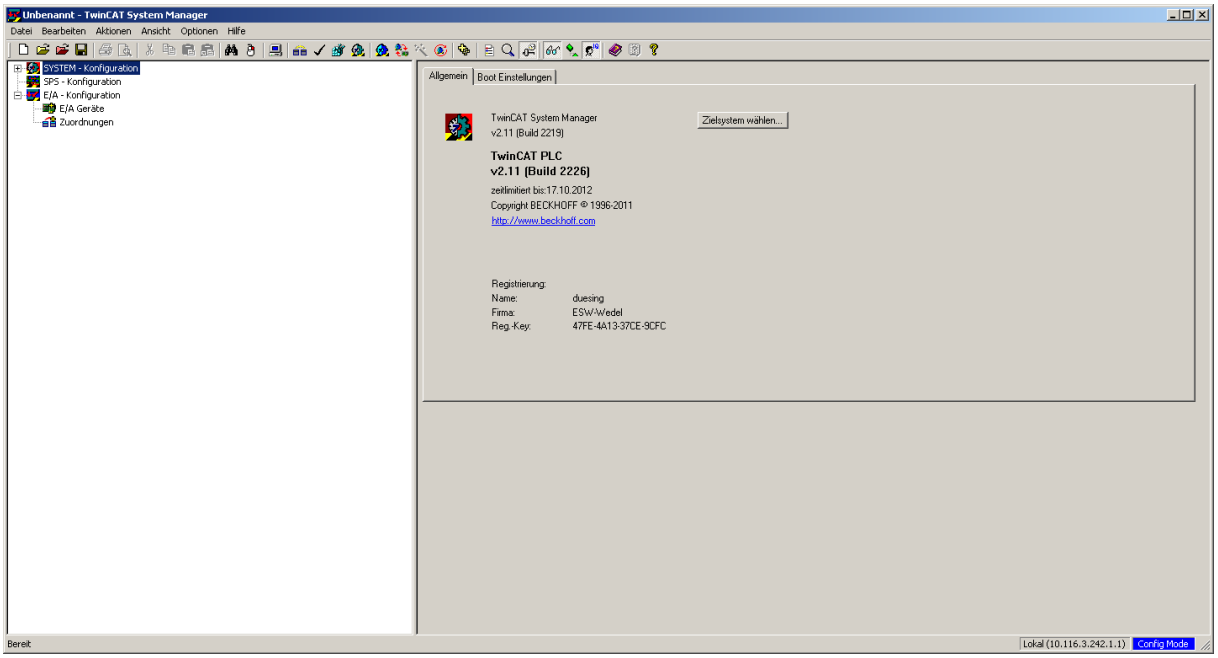
Nummer der Klemme	Signalbezeichnung
1	Sensorsignal Strom Phase A
2	-15 Volt
3	Gnd
4	+15 Volt
5	Sensorsignal Strom Phase B
6	-15 Volt
7	Gnd
8	+15 Volt
9	Sensorsignal $U_{ZW}/2$
10	-15 Volt
11	not in use
12	+15 Volt
13	Sensorsignal U_{ZW}
14	-15 Volt
15	not in use
16	+15 Volt
17	Gnd
18	Schaltsignal Schütz 1
19	Schaltsignal Schütz 2
20	Resolver Cosinus S3
21	Resolver Cosinus S1
22	Resolver Sinus S1
23	Resolver Sinus S3
24	Resolver Referenz R1
25	Resolver Referenz R2
26	Temperatur 1 Motor low
27	Temperatur 1 Motor high
28	Gnd
29	+24 Volt
30	Gnd
31	+24 Volt
32-50	for future use

Anhang 3: Beschreibung einer TwinCAT geführte Netzwerkkonfiguration

1. Inbetriebnahme im Konfigurations Modus

Nach dem Öffnen des System Managers mithilfe des TwinCAT Symbols  in der Taskleiste, startet das Programm mit folgenden Haupfenster:

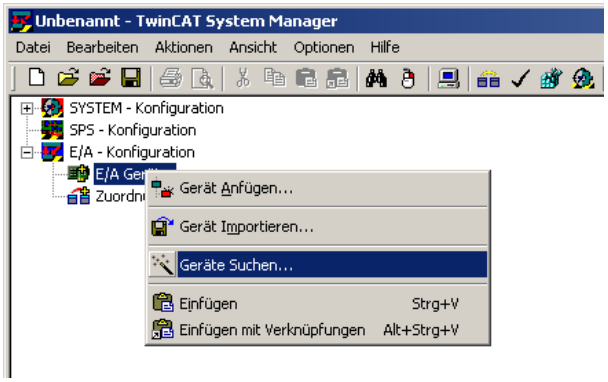




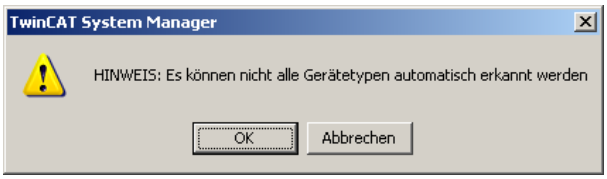
Sollte nach dem Starten des System Managers bereits eine Konfiguration geladen sein, ist durch die Menüfolge ***Datei -> Neu*** oder die Tastenkombination ***Strg + N*** eine leere Konfiguration zu öffnen.

Anschließend werden die als Master verwendbaren Netzwerkkarten oder Netzwerkadapter automatisch eingelesen, indem der Eintrag ***E/A Geräte*** mit der rechten Maus-

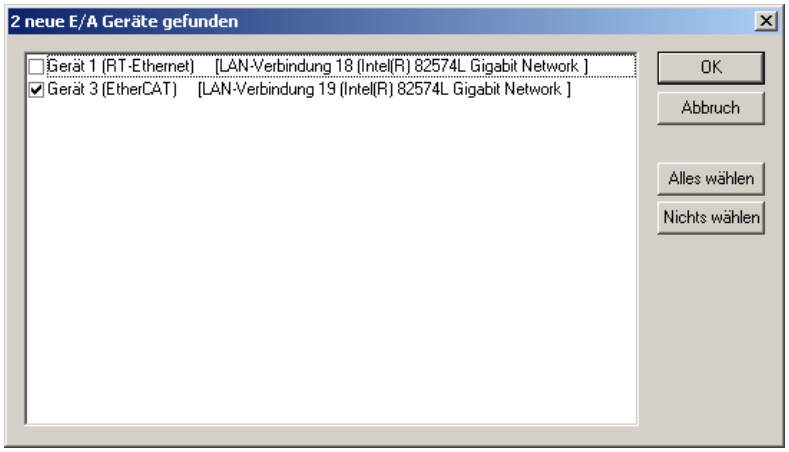
taste angeklickt und der Befehl ***Geräte Suchen*** ausgeführt wird. Alternativ kann der Eintrag E/A Geräte auch lediglich markiert und die Taste ***F5*** gedrückt werde.



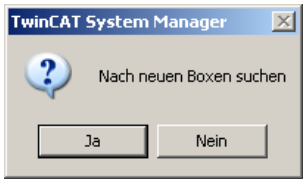
Es erscheint das folgende Hinweisfenster, welches durch den ***OK*** Button bestätigt wird.



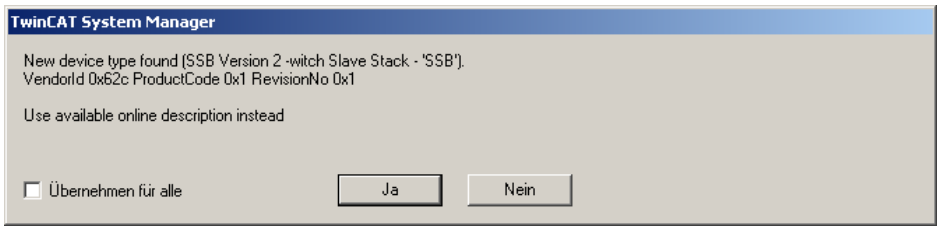
Die von der Software gefundenen E/A Geräte werden anschließend in einem Popup Fenster aufgelistet. In der Regel wird das Gerät, an welches die EtherCAT Teilnehmer angeschlossen sind, automatisch mit dem Auswahlhaken vormarkiert. Die gegebenenfalls korrigierte Auswahl ist mit ***OK*** zu bestätigen.



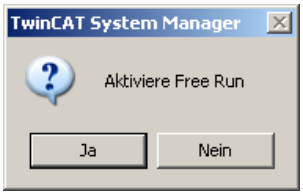
Das hieraufhin aufpoppende Fenster ist mit ***Ja*** zu bestätigen, um nach den an der Netzwerkkarte angeschlossenen Teilnehmern (Boxen) zu suchen.



TwinCAT kann neue Hardware erkennen, indem entweder deren XML-Beschreibungsdatei in den TwinCAT-Ordner ***C:\TwinCAT\Io\EtherCAT*** kopiert wird, oder indem TwinCAT diese aus den Teilnehmern ausließt. Soll TwinCAT die XML-Beschreibungsdatein selbstständig auslesen, so ist folgede Info-Box mit ***Ja*** zu quittieren:

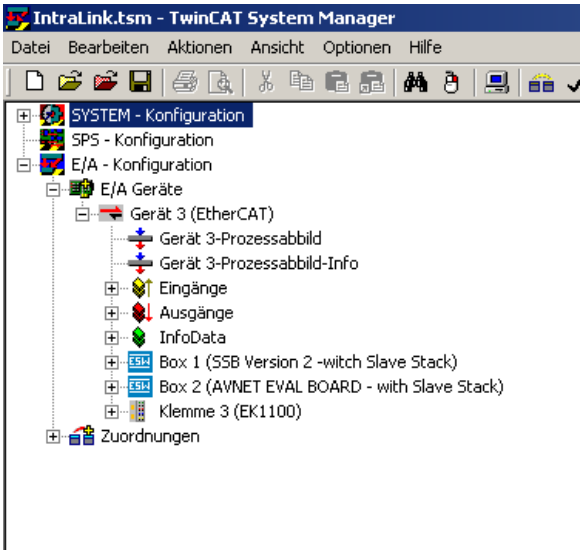


Abschließend ist durch ein weiteres Popup Fenster zu entscheiden, ob der ***Free Run*** Modus aktiviert werden soll.



Der ***Free Run*** Modus ist ein Test Modus in dem beispielsweise eine neue Konfiguration erstellt, eine bestehende getestet oder lediglich einzelne Bauteile untersucht werden können. Der ***Free Run*** Modus ist zu starten falls Ein- und Ausgänge der Teilnehmer geschrieben oder gelesen werden sollen. Darüber hinaus lassen sich die Geräte, die über Ein- und Ausgänge verfügen auch nur im ***Free Run*** Modus (oder alternativ im ***Echtzeit Modus***) in den ***OP*** (Operational) Status versetzen.

Nach der Aktivierung des ***Free Run*** Modus ist das Netzwerk zur Konfiguration bereit. Die angeschlossenen Teilnehmer werden im linken Bereich des Hauptfensters angezeigt.



Unter dem Punkt ***E/A Geräte*** wurde in diesem Fall das ***Gerät 3 (EtherCAT)***, welches den EtherCAT-Master wiederspiegelt, angefügt. Hierarchisch eine Ebene tiefer wurden die Slaves ***Box 1 (SSB Version 2 – with Slave Stack), Box 2 (AVNET EVAL BOARD – with Slave Stack)*** und ***Klemme 3 (EK1100)*** angefügt. Durch das Aufblättern der Slave-Einträge werden deren ***In-*** und ***Output*** Variablen angezeigt. Diese können ausgewählt und anschließend beobachtet oder manipuliert werden.

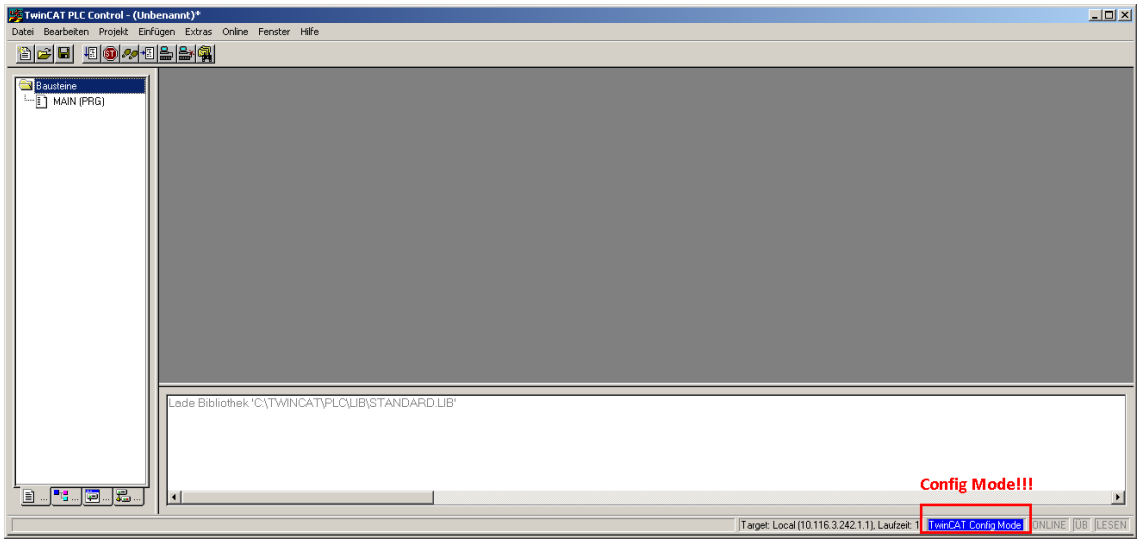
2. SPS-Programmierung mit TwinCAT PLC

Im ***Free Run*** Modus wird vom Master lediglich ein zyklisches Aktualisieren des Prozessabbildes angestoßen, sodass mit jedem Zyklus der Status der Eingänge auf die Eingangsvariablen und die ggf. manipulierten Ausgangsvariablen zu den Slaves übertragen werden.

Um in den Master darüber hinaus weitere Funktionalität zu implementieren, kann mit Hilfe des PLC Control von TwinCAT ein SPS-Programm geschrieben und über den System Manager mit dem Netzwerk verknüpft werden.

Das PLC Control wird analog zum System Manager über das TwinCAT Symbol  in der Taskleiste gestartet.

Es öffnet sich das PLC Hauptfenster, in dem ein neues SPS-Programm sowohl geladen, als auch in den Sprachen AWL, AS, FUP, ST, KOP und CFC geschrieben werden kann.



Das PLC Control nimmt parallel zum System Manager denselben Modus (***Config Mode*** oder ***Echtzeit Mode***) an.

Hinweis: Um geschrieben Programme zu testen, muss mit Hilfe des System Managers in den ***Echtzeit Modus*** gewechselt werden!

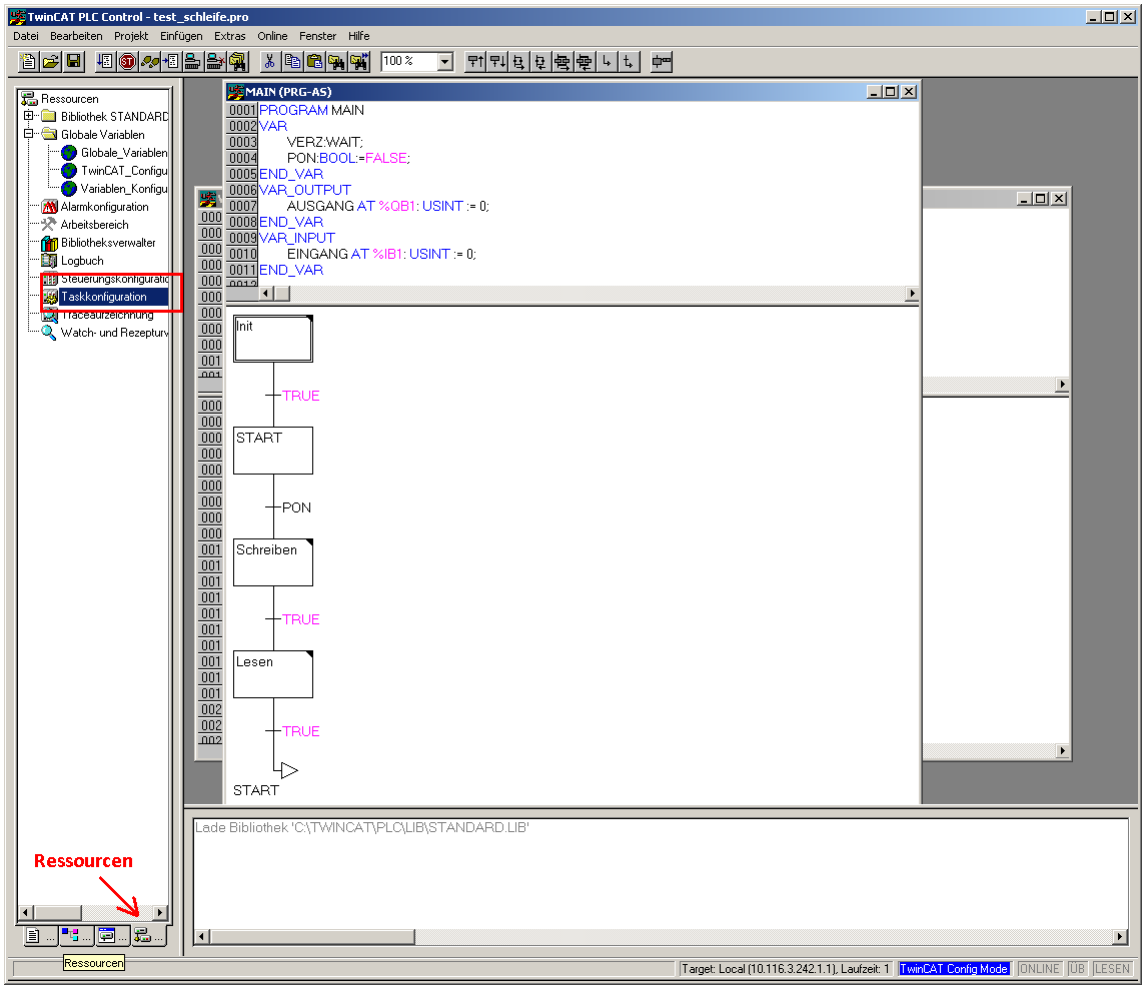
Für die SPS-gesteuerte EtherCAT-Kommunikation stellt die Firma Beckhoff zudem spezielle EtherCAT Funktionsblöcke zur Verfügung. Eine Übersicht findet sich unter:

http://infosys.beckhoff.com/index.php?content=../content/1031/tcplclibethercat/html/tcplclibtcethercat_fb_ecphysicalreadcmd.htm&id=13683

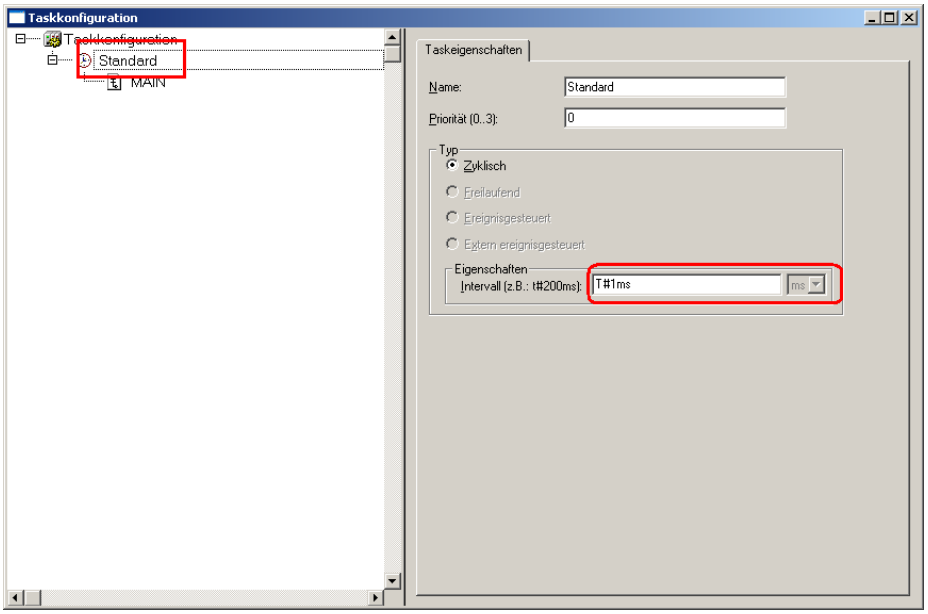
Eine beispielhafte SPS-Programmierung im PLC Control befindet sich unter:

http://infosys.beckhoff.com/index.php?content=../content/1031/tcplclibethercat/html/tcplclibtcethercat_fb_ecphysicalreadcmd.htm&id=13683

Nach dem Erstellen eines SPS Programms ist die Zykluszeit der Task anzupassen. Hierbei gilt es zu vermeiden, dass eine Task schneller aufgerufen, als abgearbeitet wird. Das Menü zum konfigurieren der Task-Zykluszeit lässt sich unter dem Reiter ***Ressourcen*** und dem Objektbaumpunkt ***Taskkonfiguration*** durch einen Doppelklick öffnen.



Unter dem Menüpunkt ***Standard***, des aufpoppenden Taskkonfiguration-Fensters kann die gewünschte Zykluszeit eingetragen werden. Der kleinste Wert ist ***T#1ms*** und entspricht 1ms.



Hinweis: Es ist möglich eine geringere Zykluszeit als 1ms zu konfigurieren. Hierfür muss die Einstellung ***Zykluszeit als Ticks interpretieren*** im TwinCAT System Manager aktiviert werden! Nähere Informationen unter dem Anhang 3 - 4.3.2. Zykluszeit SPS Task.

Langsame SPS-Aktionen können in eine weitere Task ausgelagert werden. Es werden bis zu vier verschiedene Task mit unterschiedlichen Zykluszeiten unterstützt.

Hinweis: Im Fall der Verwendung von mehrerer Tasks, muss die Task mit der geringsten Zykluszeit, die höchste Priorität zugewiesen werden!

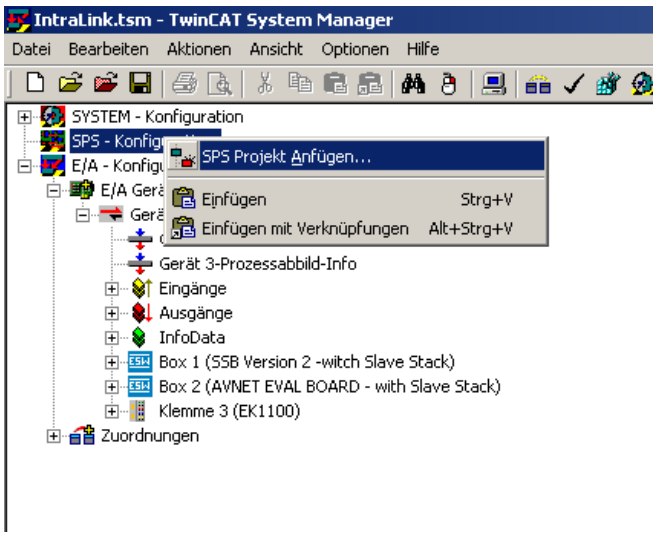
Hinweis: Im PLC muss nach der erfolgreichen Programmierung und Erprobung eines Programms über die Befehlskette ***Online -> Erzeugen eines Bootprojektes*** ein für den System Manager bootfähiges Projekt erzeugt werden!

3. Inbetriebnahme im Echtzeit Modus

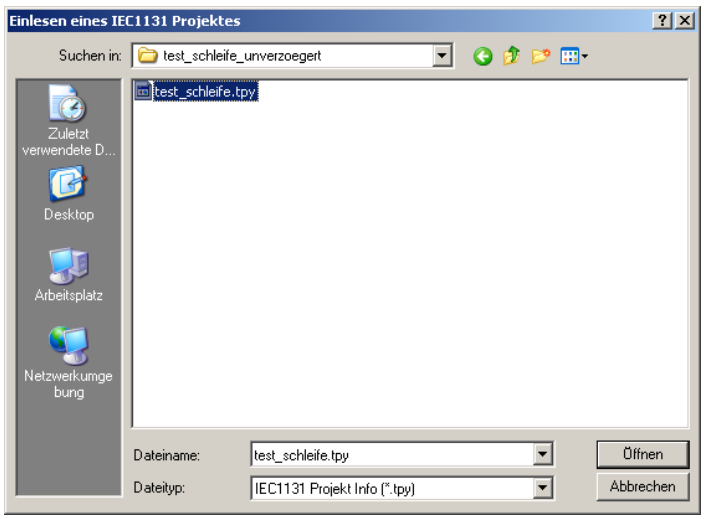
Um mit dem System Manager in den ***Echtzeit Modus*** zu wechseln, muss zuvor eine Netzwerkkonfiguration anhand des Anhangs 3 Kapitel 1 in Betrieb genommen worden sein.
Anschließend muss mindestens eine Task mit einem Teilnehmer des Netzwerks verbunden werden. Hierfür kann entweder eine ***SPS-Task*** oder eine sog. ***Zusatz-Task*** verwendet werden.

3.1. Echtzeitzeit-Modus mit SPS Task

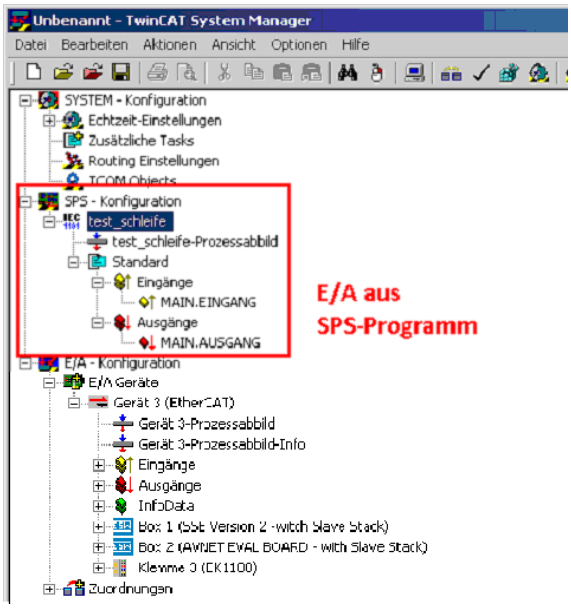
Nach der Inbetriebnahme des Netzwerkes im ***Config Mode*** wird das zu verwendende SPS-Programm mittels Rechtsklick auf den Objektbaumeintrag ***SPS-Konfiguration*** und dem Befehl ***SPS Projekt Anfügen*** zunächst ausgewählt.



Im daraufhin erscheinenden Auswahlfenster ist das zu verwendende Projekt zu suchen und auszuwählen. Der Dialog ist mit dem Button ***Öffnen*** zu quittieren.

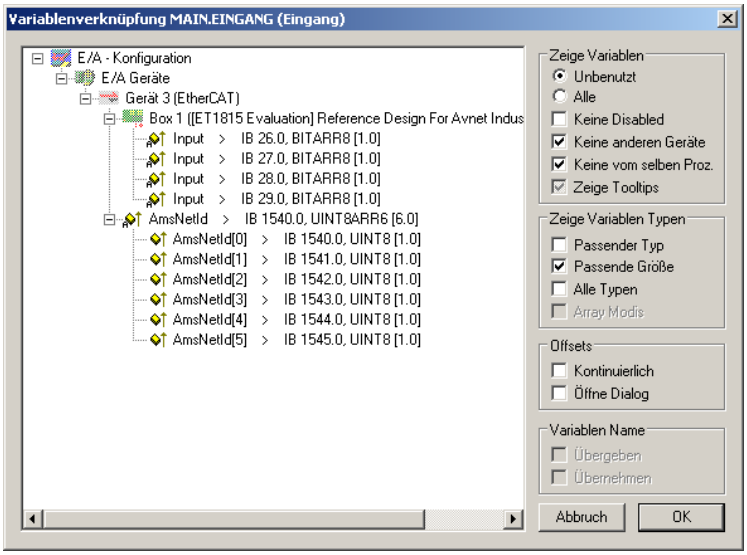


Das SPS-Programm wird in den System Manager eingebunden, die in dem Programm deklarierten Ein- und Ausgangsvariablen stehen anschließend im System Manager zur Verfügung.

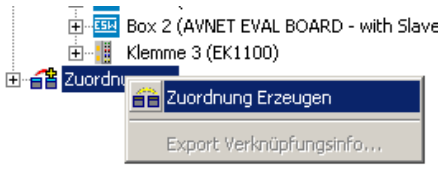


Um abschließend eine Verknüpfung zwischen der Task und dem Prozess herzustellen, ist mindestens eine Variable der SPS mit der rechten Maustaste anzuklicken und der Befehl ***Verknüpfung ändern*** auszuwählen.

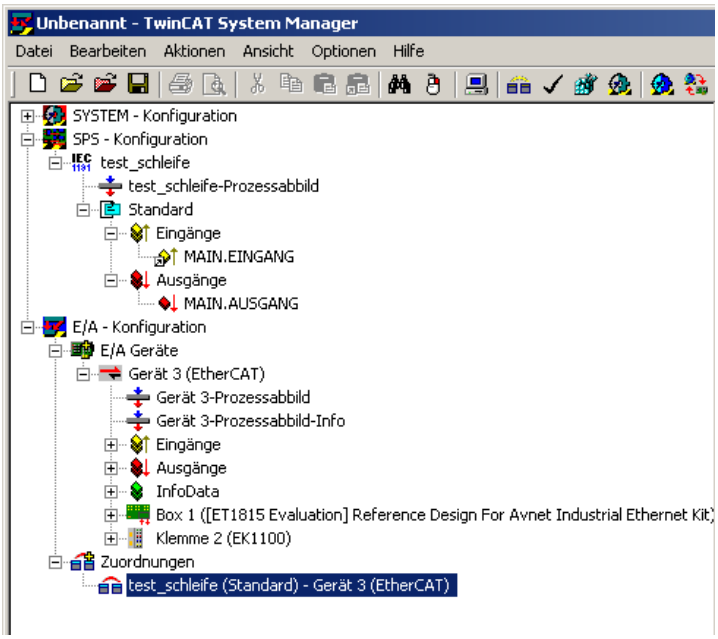
Im hieraufhin erscheinenden Popup-Fenster ist eine Prozess-Variable zu wählen, mit der die SPS-Variable verknüpft werden soll. Das Fenster ist nach der Auswahl mit OK zu schließen.



Um die Verknüpfung abzuschließen muss mit einem Rechtsklick auf den Objektbaumeintrag ***Zuordnungen*** geklickt und der Befehl ***Zuordnungen erzeugen*** ausgeführt werden.



Es stellt sich folgende Konfiguration ein:

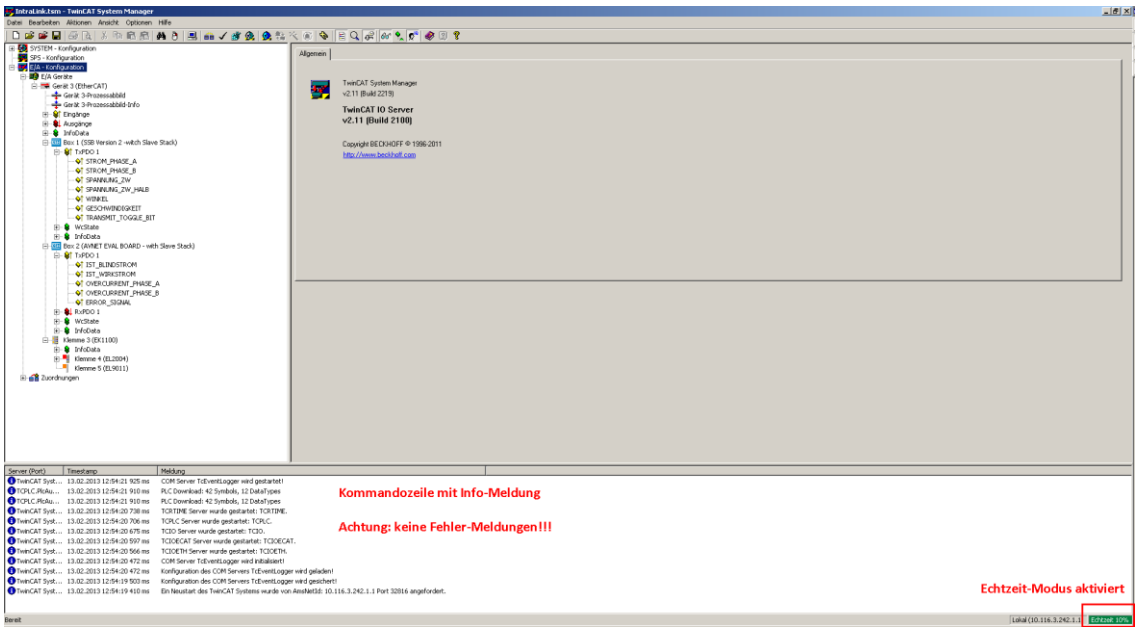


Die erstellte Konfiguration ist abschließend durch Klicken des Symbols  oder die Menüfolge ***Aktion -> Überprüfen der Konfiguration*** zu überprüfen. Sofern keine Fehler in der Kommandozeile erscheinen, kann die Konfiguration durch drücken des Symbols  oder durch Auswahl der Menüfolge ***Aktion -> Aktiviert Konfiguration*** gestartet werden.

Es erscheint das Hinweisfenster, dass die alte Konfiguration überschrieben wird, welches mit ***OK*** zu bestätigen ist.

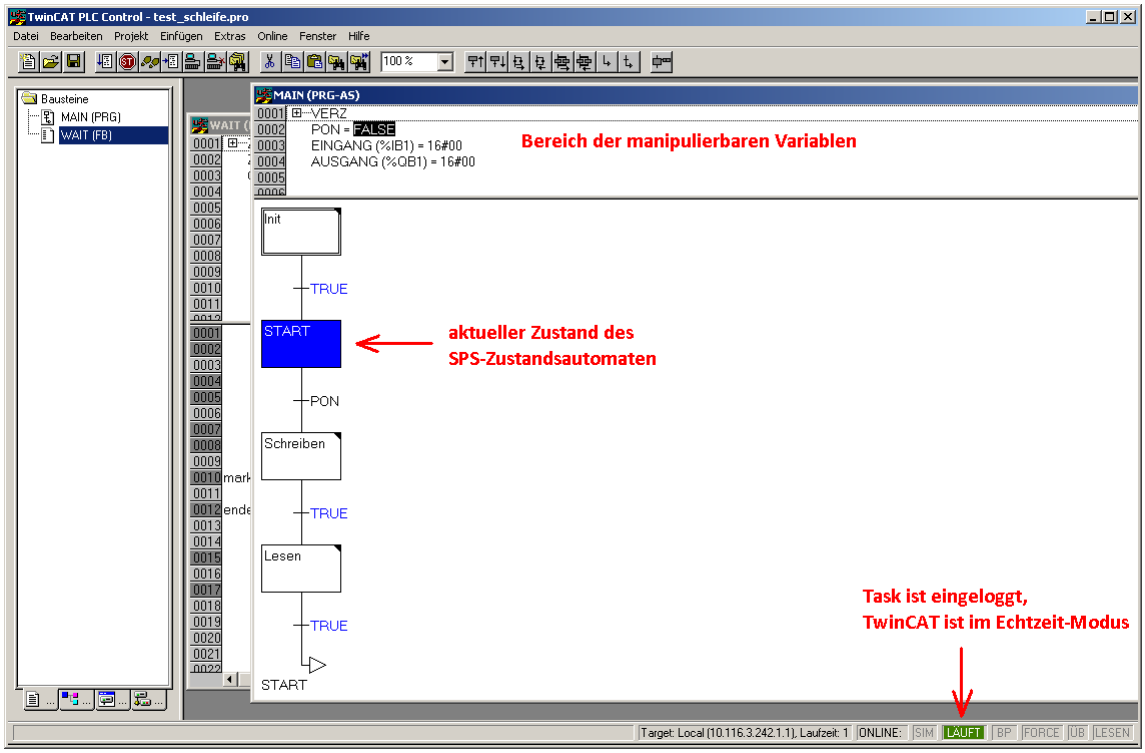
Durch das Quittieren des nächsten Popup-Fensters mit ***OK*** wechselt TwinCAT in den ***Run Modus*** (***Echtzeit-Modus***).

Das EtherCAT-Netzwerk befindet sich nun im Echtzeit-Betrieb. Das Hauptfenster von TwinCAT stellt sich wie folgt dar:



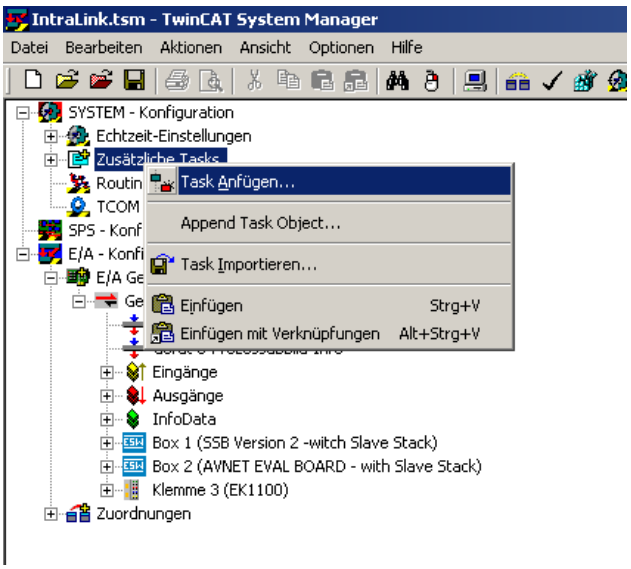
Parallel zum TwinCAT System Manager kann bzw. muss das PLC Control gestartet werden um die Variablen der SPS, die keine E/A-Variablen sind, zu beobachten oder zu manipulieren (schreiben oder forcen).

Das PLC Control ist hierfür zu Starten und die im System Manager verwendete Task zu laden. Anschließend wird die Task mit der Befehlsfolge ***Online -> Einloggen*** aktiviert.

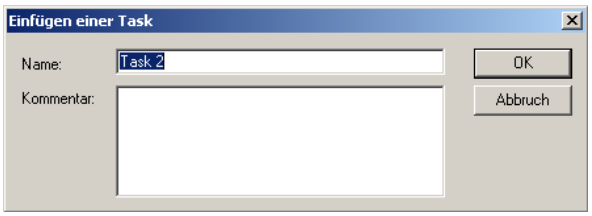


3.2. Echtzeit Modus mit Zusatz-Task

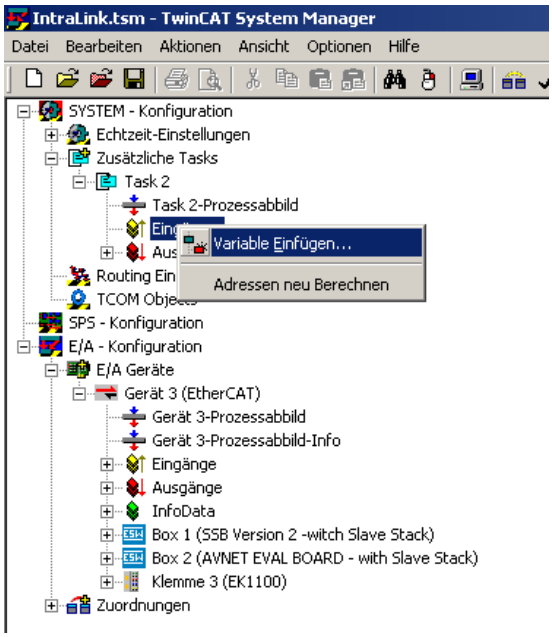
Die Inbetriebnahme des Netzwerkes mit Hilfe einer Zusatz-Task bedarf ebenfalls eines zuvor konfiguriertem Netzwerks im ***Config Mode***. Anschließend wird die sog. Zusatz-Task durch einen Rechtsklick auf den Objektbaumeintrag ***Zusätzliche Tasks*** und dem Befehl ***Task Anfügen*** hinzugefügt.



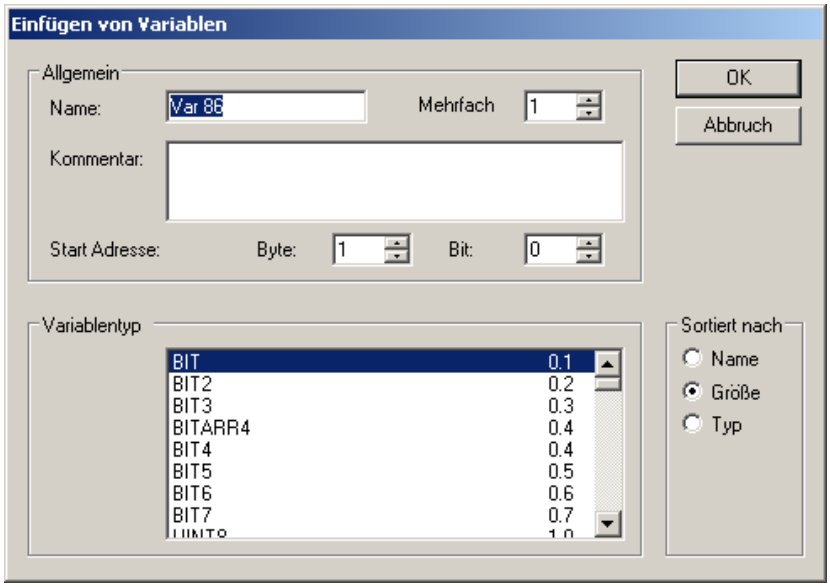
Die neue Task kann im daraufhin aufpoppenden Fenster umbenannt werden. Die Umbenennung ist mit dem ***OK*** Button zu bestätigen.



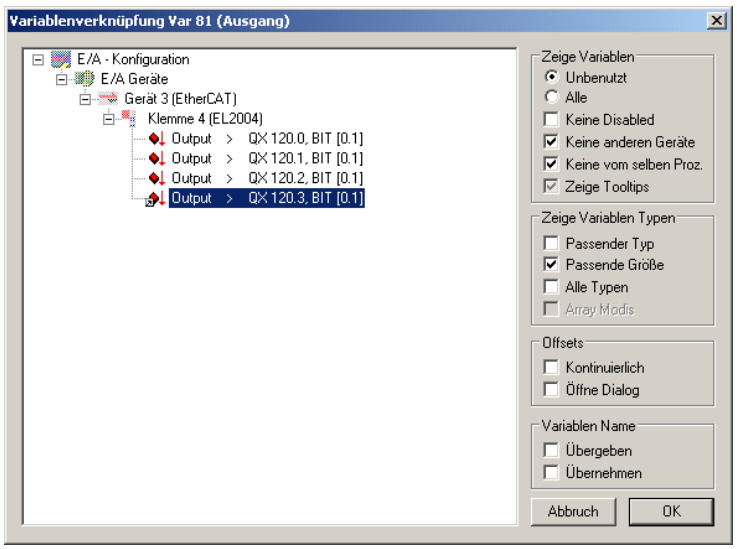
Da die neue Task, genau wie im „***Echtzeit-Modus*** mit SPS-Task“, mindestens eine Variablenverknüpfung zum Prozess braucht, muss zunächst eine beliebige Task-Variable erzeugt werden. Hierfür ist auf einen der Unterpunkte ***Eingänge*** oder ***Ausgänge*** der neuen Task im Objektbaum mit der rechten Maustaste zu klicken. Es ist der Befehl ***Variable einfügen*** auszuführen.



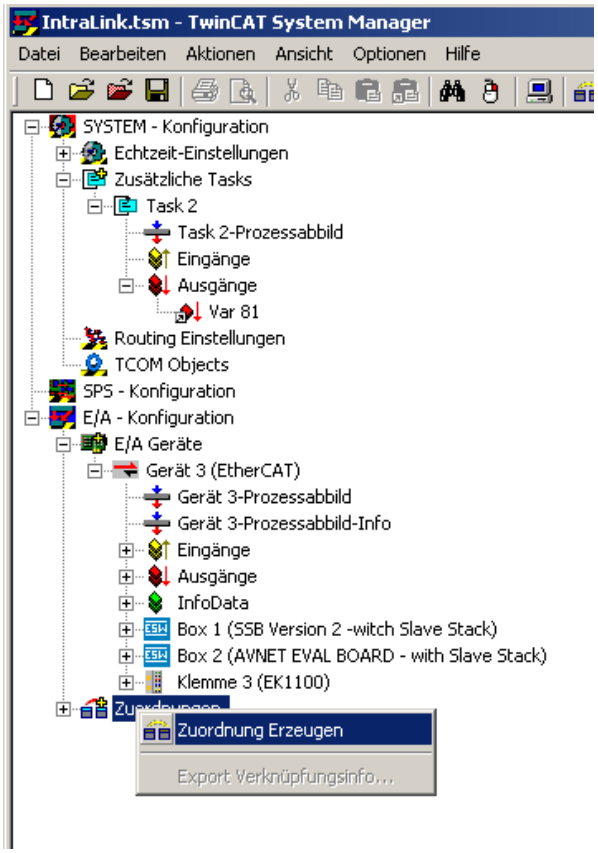
Im daraufhin erscheinenden Popup-Fenster kann der Variablen-Typ, -Name, usw. ausgewählt werden. Das Fenster ist mit ***OK*** zu schließen. Die neue Variable ist erstellt und kann verknüpft werden.



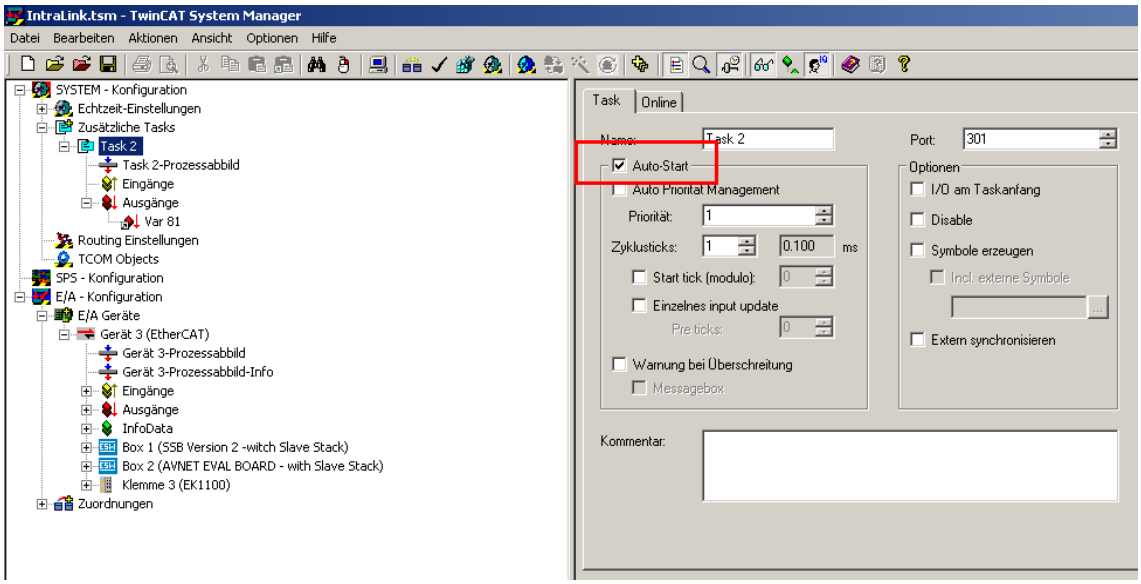
Hierfür muss die Variable mit der rechten Maustaste angeklickt und der Befehl ***Verknüpfung ändern*** ausgeführt werden. Im aufpoppenden Fenster ist eine der Prozess-Variablen zu wählen, mit der die SPS-Variable verbunden werden soll. Das Fenster ist mit ***OK*** zu schließen, die Variablen-Verknüpfung ist hergestellt.



Im Anschluss muss die neue Variablen-Verknüpfung durch das Erzeugen einer Zuordnung abgeschlossen werden. Hierfür ist im Objektbaum der Punkt ***Zuordnungen*** zu wählen und mittels Rechtsklick der Befehl ***Zuordnung erzeugen*** auszuführen.

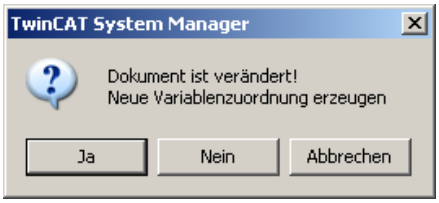


Da die erstellte Zusatz-Task nicht über das PLC Control gesteuert werden kann, muss an dieser Stelle die Auto-Start Funktion der Task aktiviert werden. Hierfür ist die Task zu markieren. Im rechts erscheinenden Menü des System Managers ist der Reiter ***Task*** zu wählen und der Punkt ***Auto-Start*** durch setzten des Hakens zu aktivieren.



Abschließend ist die Konfiguration zu überprüfen, indem das Symbol ✓ aus der Menüleiste gedrückt oder die Befehlsfolge ***Aktionen -> Überprüfen der Konfiguration***

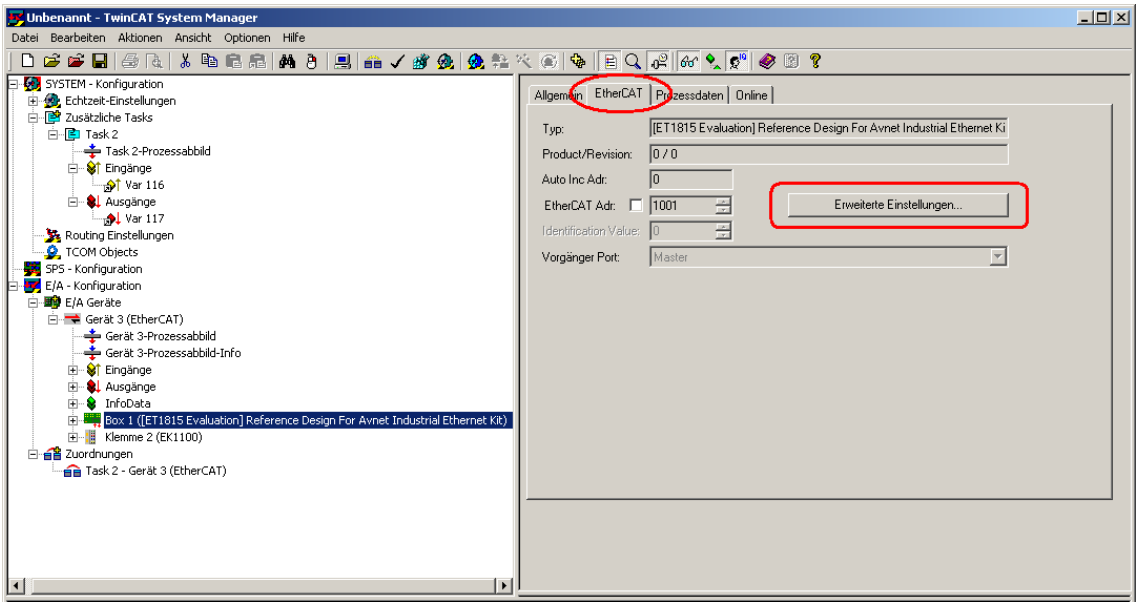
ausgeführt wird. Falls keine Fehler-Meldungen im Kommandofenster erscheinen, kann TwinCAT in den ***Echtzeit-Modus*** versetzt werden. Hierfür ist das Symbol  zu drücken oder die Befehlsfolge ***Aktionen -> Aktiviert Konfiguration*** auszuführen.
Die folgenden drei Popup-Fenster sind mit ***OK*** zu bestätigen.



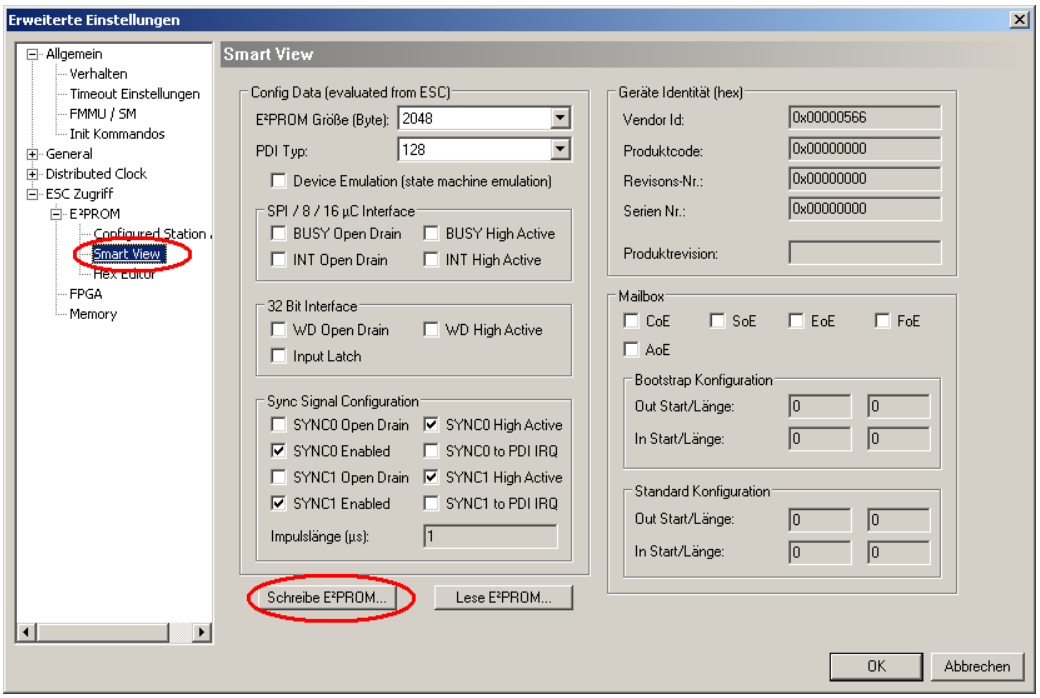
4. Allgemeine Hinweise im Umgang mit TwinCAT

4.1. XML-Konfigurationsdatei in EEPROM laden

Um einen Slave via TwinCAT neu zu konfigurieren ist zunächst ein Netzwerk im ***Config-Mode*** (vgl. Anhang 3 Kapitel 1 Inbetriebnahme im Konfigurations Modus) zu erstellen. Der Teilnehmer ist anschließend zu markieren, wobei im rechten Bereich des System Managers das Register ***EtherCAT*** ausgewählt wird. Es ist der Button ***Erweiterte Einstellungen*** zu drücken.



Im aufpoppenden Fenster ist im links befindlichen Objektbaum ***ESC Zugriff*** -> ***E²PROM*** -> ***Smart View*** aufzublättern. Im dazugehörigen Menü rechts, ist der Button ***Schreibe E²PROM*** zu drücken.

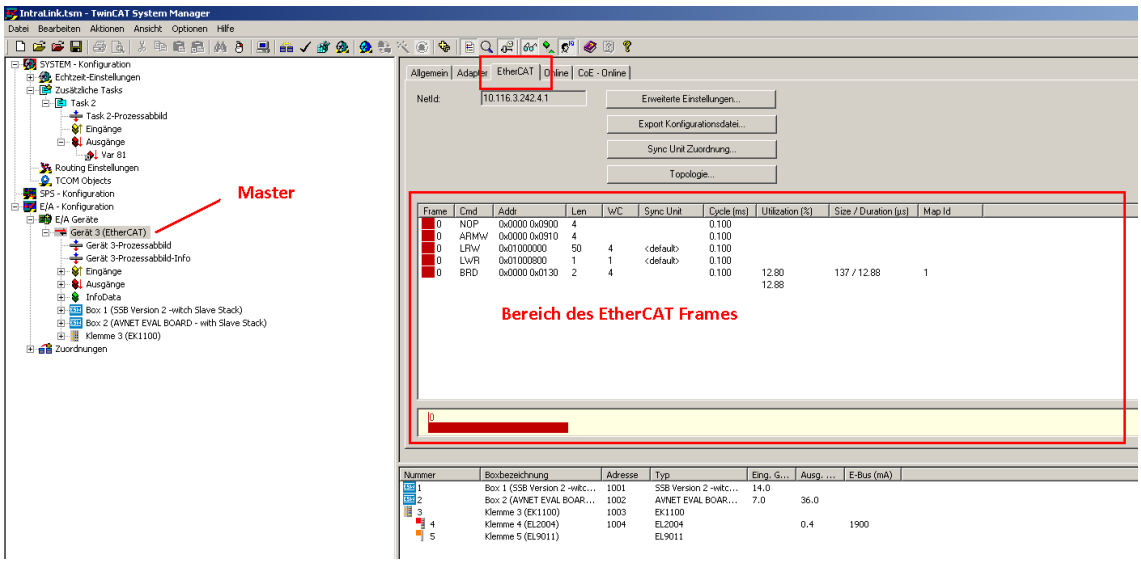


Es öffnet sich das Dialog-Fenster zur Suche der XML-Datei. Nach der korrekten Datei-Auswahl kann der Dialog mit ***OK*** geschlossen werden. Das EEPROM wird anschließend neu beschrieben, der Teilnehmer ist somit neu konfiguriert.

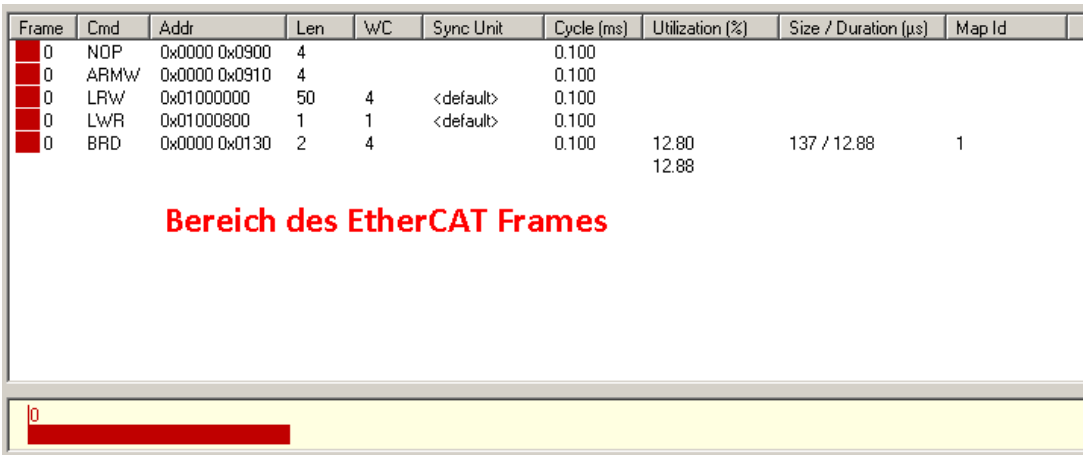
4.2. EtherCAT Frame in TwinCAT

Sowohl im ***Config-Mode*** als auch im ***Echtzeit-Mode***, lassen sich die vom Master erstellten Frames anzeigen. Dies ist bspw. dann wichtig, wenn eine Slave-to-Slave Kommunikation hergestellt werden soll (vgl. Anhang 3 Kapitel 5.1. Slave-to-Slave Kommunikation).

Angezeigt wird der Frame indem der EtherCAT-Master (in der Abbildung ***Gerät 3***) markiert wird. Im rechten Bereich des System Managers ist der Reiter ***EtherCAT*** auszuwählen.



Im Bereich der aufbereiteten EtherCAT-Frames werden unter anderem die im Frame enthaltenen Kommandos (***Cmd***) aufgeführt. Hierbei handelt es sich bspw. um ***LRW*** (logical read write), ***LWR*** (logical write), ***BRD*** (broadcast read).

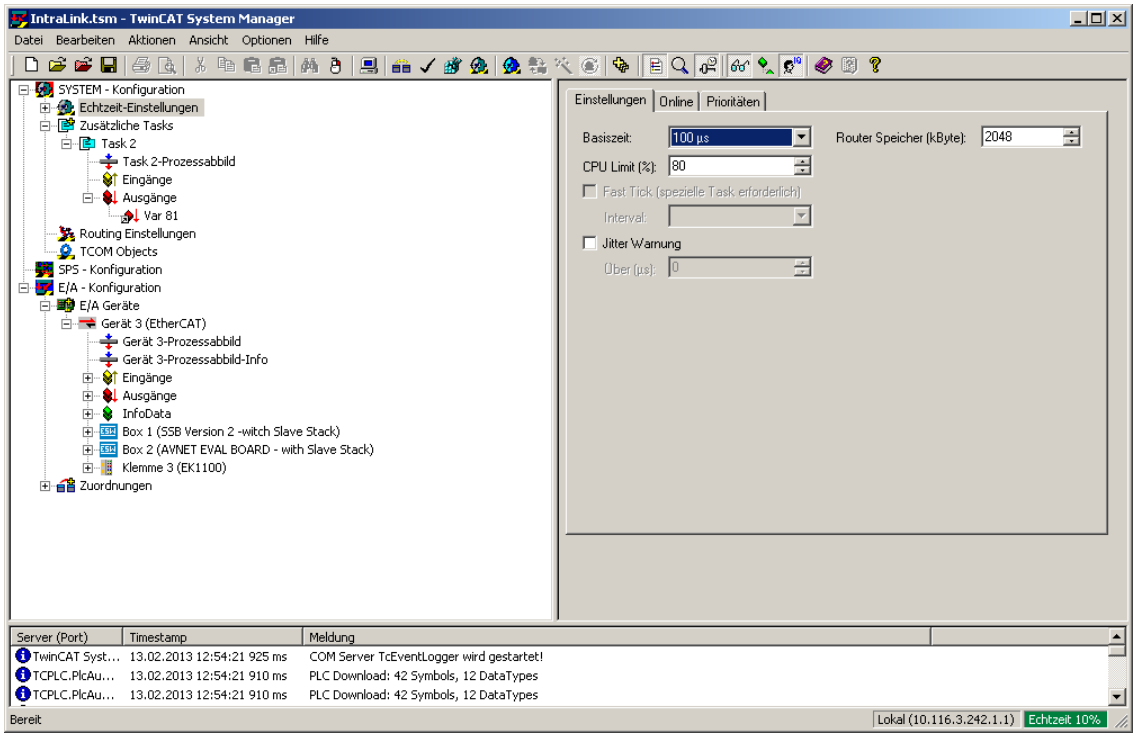


Im Anschluss an das Kommando folgt die ***logische Adresse*** sowie die ***Datenlänge***. Bei der logischen Adressen handelt es sich um einen festen Bereich im EtherCAT Prozess-

abbild, der in Abhängigkeit vom Kommando durch Teilnehmer bspw. nur beschrieben, gelesen oder geschrieben und gelesen werden kann. Dies ist bei der direkten Slave-to-Slave Kommunikation von Bedeutung, da in diesem Fall von beiden Teilnehmern derselbe Adressbereich der logischen Adresse bearbeitet werden muss.

4.3. Änderung der Zykluszeiten

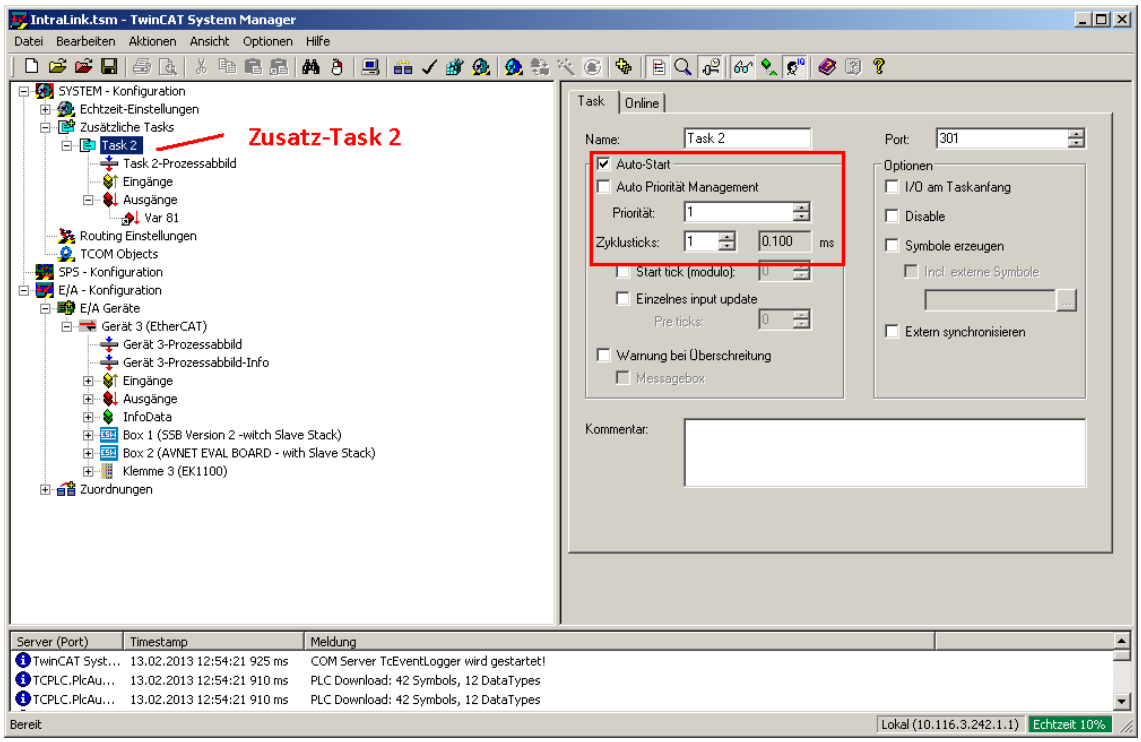
Um die Zykluszeiten zu ändern sollte TwinCAT zunächst mit einer laufenden Konfiguration im ***Echtzeit-Modus*** verwendet werden. Anschließend ist im Objektbaum der Eintrag ***Echtzeit-Einstellungen*** zu markieren und im rechten Bereich des Hauptfensters der Reiter ***Einstellungen*** zu wählen.



Anschließend kann in der Auswahlbox ***Basiszeit*** eine Art Basis-Clock aller konfigurierten Tasks eingestellt werden. Die eigentlichen Zykluszeiten der Tasks sind dann vielfache der Basiszeit und werden separat konfiguriert! Hierbei werden wiederum die zwei Fälle ***Zusatz-Task*** und ***SPS-Task*** unterschieden.

4.3.1. Zykluszeit der Zusatz-Task

Nachdem die ***Basiszeit*** eingestellt wurde (z.B. 100µs) muss im Objektbaum der Eintrag der ***Zusatz-Task*** (z.B. ***Task 2***) markiert werden. Im rechts erscheinenden Menü ist der Reiter ***Task*** auszuwählen.



Im kenntlich gemachten Bereich ist zu kontrollieren, ob die Checkbox ***Auto-Start*** aktiviert ist. Diese muss aktiviert sein um Einstellungen an der Zykluszeit vorzunehmen. In der Auswahlbox ***Zyklusticks*** kann anschließend der Multiplikator zur Basiszeit eingestellt werden. Die Zykluszeit errechnet sich folglich aus:

Anzahl Zyklusticks x Basiszeit = Zykluszeit

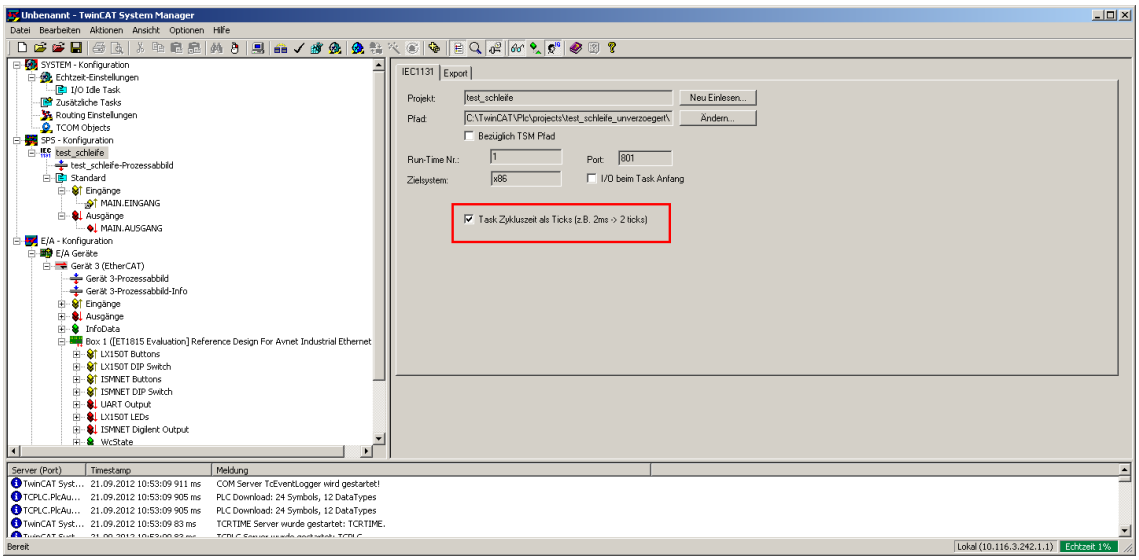
Und wird im grau hinterlegten Feld, hinter den ***Zyklusticks*** angezeigt. Die Zykluszeit wird für jede Zusatz-Task separat konfiguriert!

Um die bearbeitete Konfiguration in Betrieb zu nehmen, ist diese durch das Drücken des  ***Aktieviert Konfiguration*** -Buttons in der Menüleiste zu starten.

4.3.2. Zykluszeit der SPS-Task

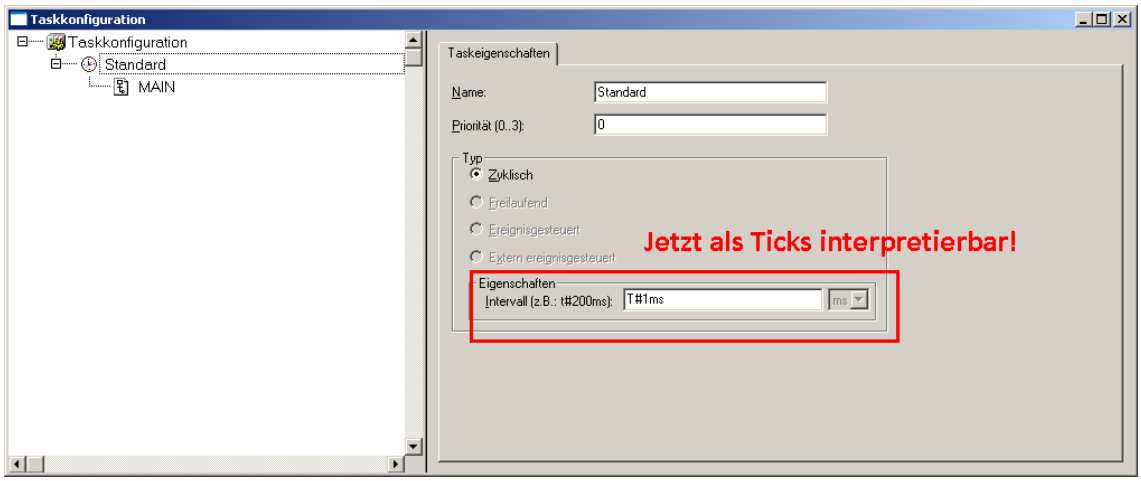
Es ist mit Hilfe des System Manager eine laufende Echtzeit-Konfiguration gemäß Anhang 3 Kapitel 3.1. Echtzeit-Mode mit SPS-Task zu starten. Die ***Basiszeit*** ist anschließend je nach Bedarf (z.B. 100µs) einzustellen. Sofern die Basiszeit kleiner 1ms ist kann

im Objektbaum unter dem Eintrag der SPS-Task (z.B. ***test_schleife***) im rechten Bereich der Reiter ***IEC1131*** ausgewählt werden.



Die Checkbox ***Task Zykluszeit als Ticks*** ist jetzt nicht mehr grau hinterlegt und muss aktiviert werden.

Um die Zykluszeit der SPS-Task zu verändern muss anschließend das PLC Control gestartet werden. Die Zykluszeit wird gemäß Anhang 3 Kapitel 2. SPS-Programmierung mit TwinCAT PLC eingestellt, jedoch in diesem Fall anders interpretiert!



Die eingetragenen Werte (z.B. ***T#1ms***) werden nach der oben beschriebenen Änderung im System Manager nicht mehr als Millisekunden sondern als Ticks der ***Basiszeit*** interpretiert. Eine Einstellung von T#5ms bei einer Basiszeit von 100µs entspricht folglich einer SPS-Zykluszeit von 500µs! Die Zykluszeit berechnet sich in diesem Fall aus:

Zykluszeit = Basiszeit x Anzahl Ticks der SPS

Wobei eine Millisekunde im PLC Control einem Tick der Basiszeit entspricht.

Die SPS-Task muss ggf. mit Hilfe der Befehlskette ***Projekt -> Alles Übersetzen*** neu geladen werden. In diesem Fall sollte auch ein neues Bootprojekt durch den Befehl ***Online -> Erzeugen eines Bootprojektes*** erzeugen erzeugt werden.

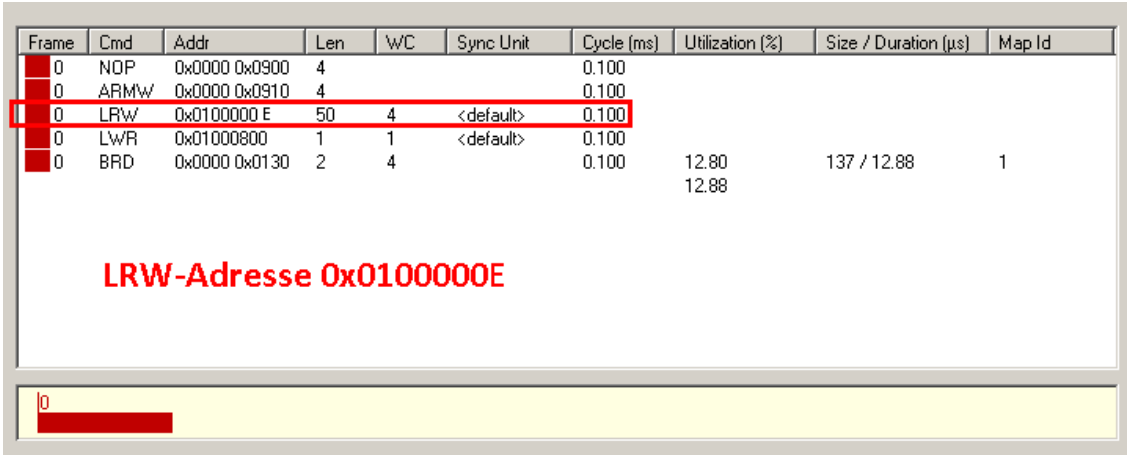
5. Besonderheiten und Extras

5.1. Slave-to-Slave Kommunikation

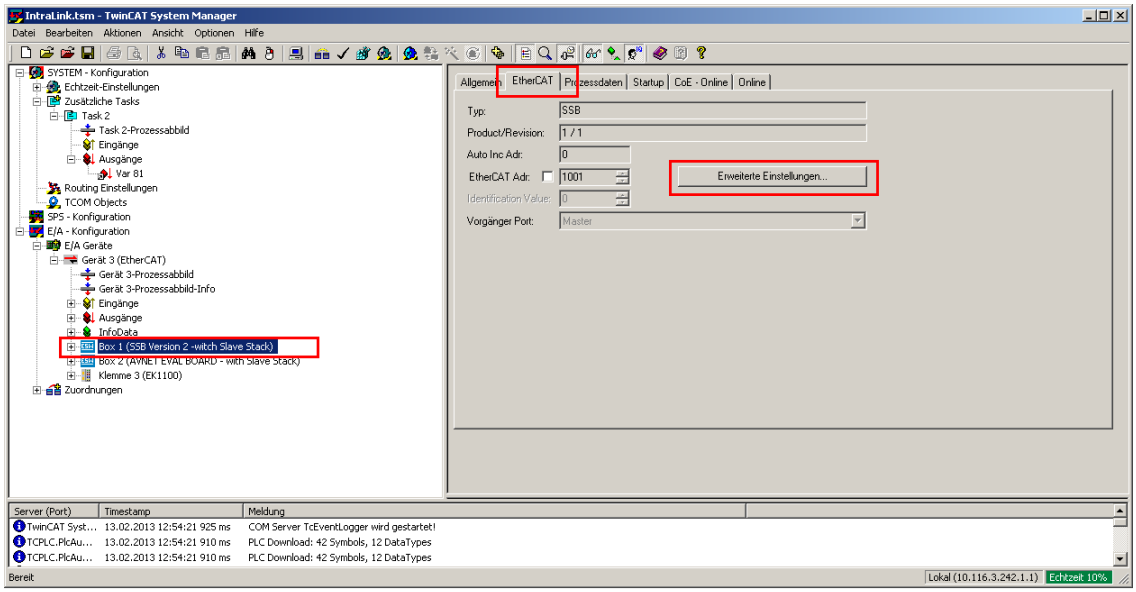
Ein Slave-to-Slave Kommunikation sollte immer über den Master mit Hilfe geeigneter Variablenverknüpfungen und einer SPS-Task erfolgen. Eine direkte Querkommunikation ist zwar möglich, verhindert aber die Working Counter Kontrolle des Masters. Bei der Working Counter Kontrolle überprüft der Master, ob die erwarteten Frame-Zugriffe der Slaves durchgeführt wurden. Hierfür wird, der dem Protokoll angefügte Working Counter eines jeden Datagramms, der in Abhängigkeit der Frame-Bearbeitung um 1 – Lesen, 2 – Schreiben oder 3 – Lesen und Schreiben erhöht wird, nach jedem Zyklus ausgewertet. Eine Manipulation von zwei oder mehr Teilnehmern eines Datagramms, welche bei der direkten Querkommunikation notwendig ist, überschreibt folglich den Working Counter Wert des Vorgängers. Es wird lediglich die letzte Frame-Bearbeitung kontrolliert. Es ist somit nicht gewährleistet, dass der Vorgänger in einer Querkommunikation die korrekte Frame-Bearbeitung durchgeführt hat, da der Nachfolger-Slave die Auswertung des Working Counters nicht selbst durchführt.

Beispiel: Slave-to-Slave Kommunikation zwischen Smart Sensor Board und Smart Driver Board (AVNET Eval Board)

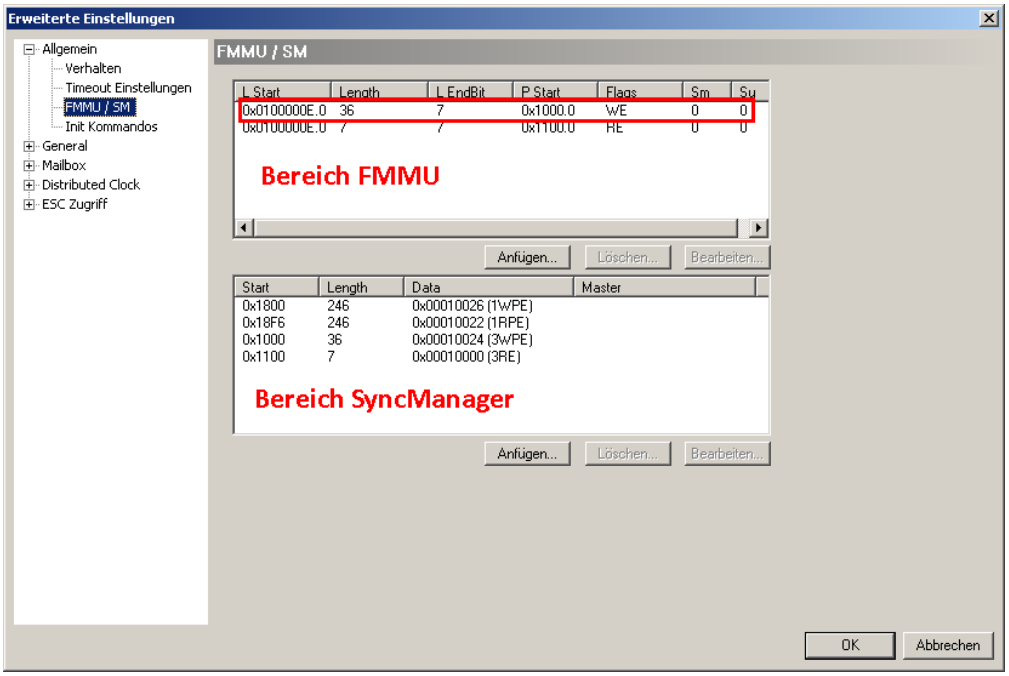
Für die direkte Querkommunikation muss die verwendete logische Adresse des LRW-Kommandos des Empfängers (Smart Driver Board) notiert werden (vgl. Anhang 3 Kapitel 4.2. EtherCAT-Frame in TwinCAT), da die Daten der Querkommunikation von den beteiligten Teilnehmern sowohl gelesen als auch geschrieben werden sollen.



Die FMMUs sind bei jedem Teilnehmer dafür verantwortlich, dass die logische Adresse (Prozessabbild bzw. Frame) mit der physikalischen Adresse (bspw. Register im Slave) verknüpft wird. Bei der Querkommunikation müssen alle beteiligten Teilnehmer folglich auf dieselbe logische Adresse, in diesem Fall ***0x0100000E***, zugreifen. Die FMMUs lassen sich durch das Markieren der Teilnehmer (z.B. ***Box 1 SSB*** und ***Box 2 AVNET Eval***) und das Öffnen der ***Erweiterten Einstellungen*** konfigurieren.



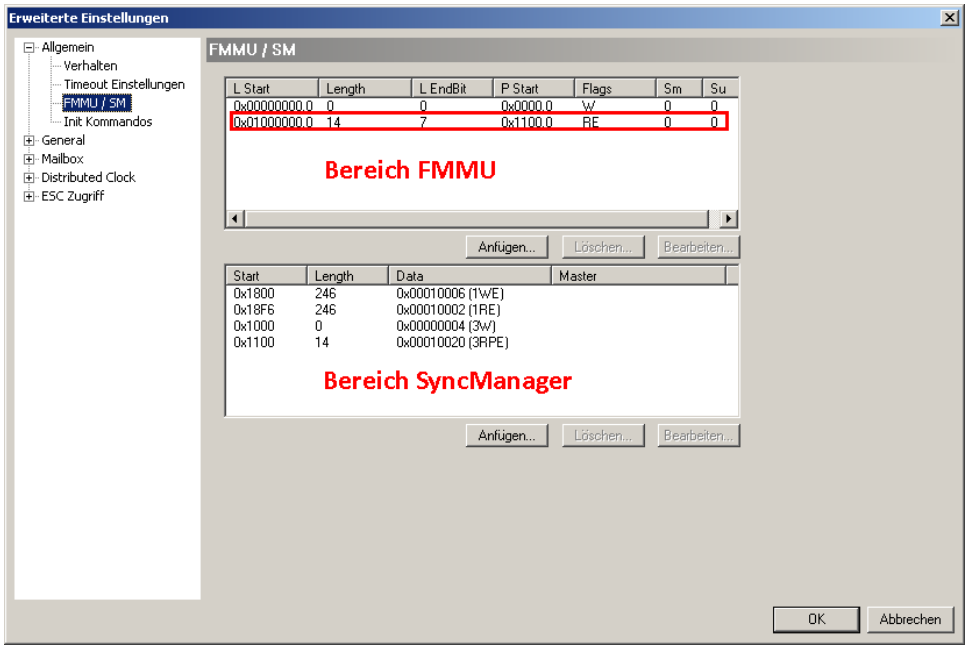
Im aufpoppenden Fenster wird unter dem Eintrag ***Allgemein*** der Unterpunkt ***FMMU/SM*** geöffnet.



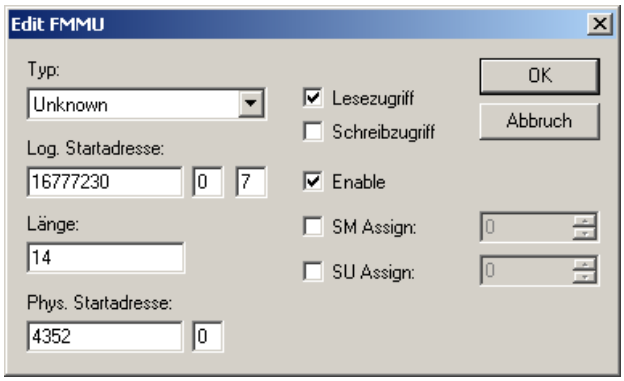
Es werden zwei FMMUs angezeigt, da das AVNET Eval Board sowohl Eingangs- als auch Ausgangsvariablen hat. Die korrekte FMMU lässt sich anhand der ***P Start Adresse*** (physikalische Startadresse) erkennen. Diese ist identisch mit der Startadresse des Sync Managers, der für die Ausgangsvariablen (u.a. Motor_Enable) genutzt wird. Die Startadresse des Output Sync Managers lässt sich zudem anhand des Slave XML-Files bestimmen:

```
</Profile>
<Fmmu>Outputs</Fmmu>
<Fmmu>Inputs</Fmmu>
<Sm DefaultSize="246" StartAddress="#x1800" ControlByte="#x26" Enable="1">MBoxOut</Sm>
<Sm DefaultSize="246" StartAddress="#x18f6" ControlByte="#x22" Enable="1">MBoxIn</Sm>
<Sm DefaultSize="36" StartAddress="#x1000" ControlByte="#x34" Enable="1">Outputs</Sm>
<Sm DefaultSize="7" StartAddress="#x1100" ControlByte="#x00" Enable="1">Inputs</Sm>
- <RxPdo Fixed="1" Sm="2">
  <Index>#x1600</Index>
  <Name>RxPDO 1</Name>
```

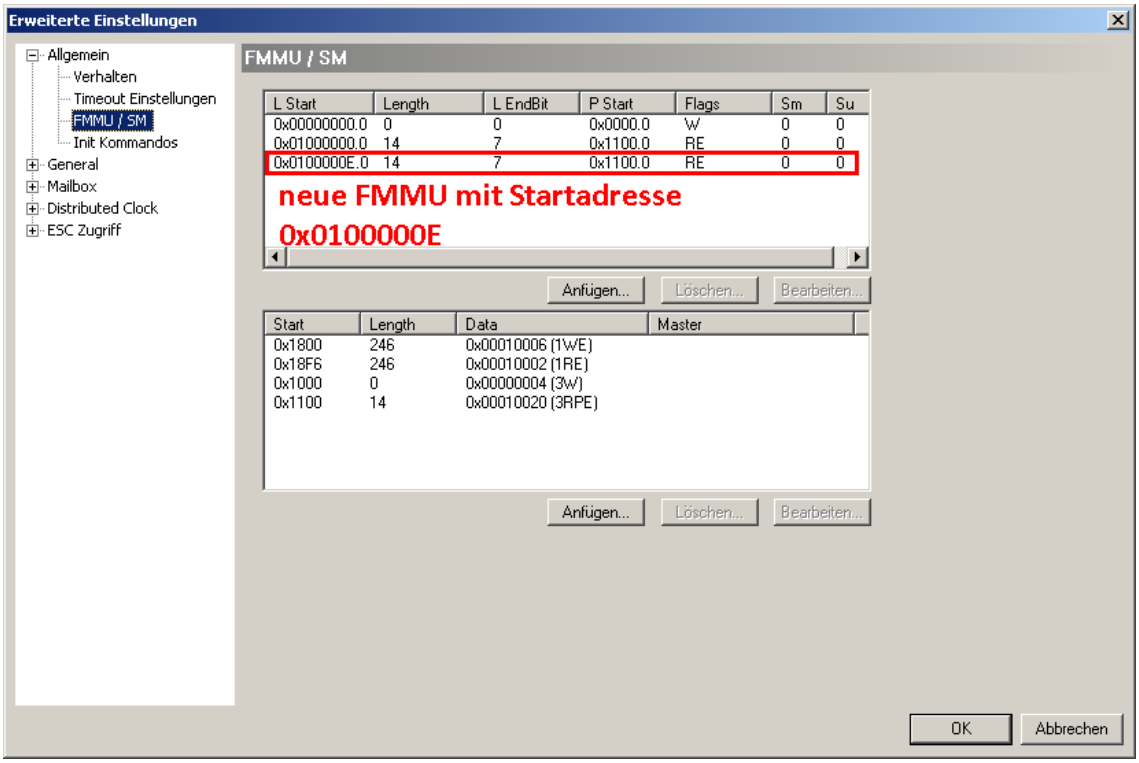
Anschließend muss auf dem Smart Sensor Board eine FMMU hinzugefügt werden, die im zuvor notierten logischen Adressbereich arbeitet.



Durch den Button Anfügen im oberen Bereich der FMMUs wird eine neue FMMU hinzugefügt. In das erste Feld der ***logischen Startadresse*** ist der Dezimalwert 16777230 (entspricht Hex-Wert von 0x0100000E) einzutragen.



Der FMMU-Typ ist zudem auf Unknown zu ändern, die Flags „Lesezugriff“ und „Enable“ werden gesetzt. Die Konfiguration wird mit OK bestätigt, die Liste der eingetragenen FMMUs hat sich entsprechend aktualisiert.

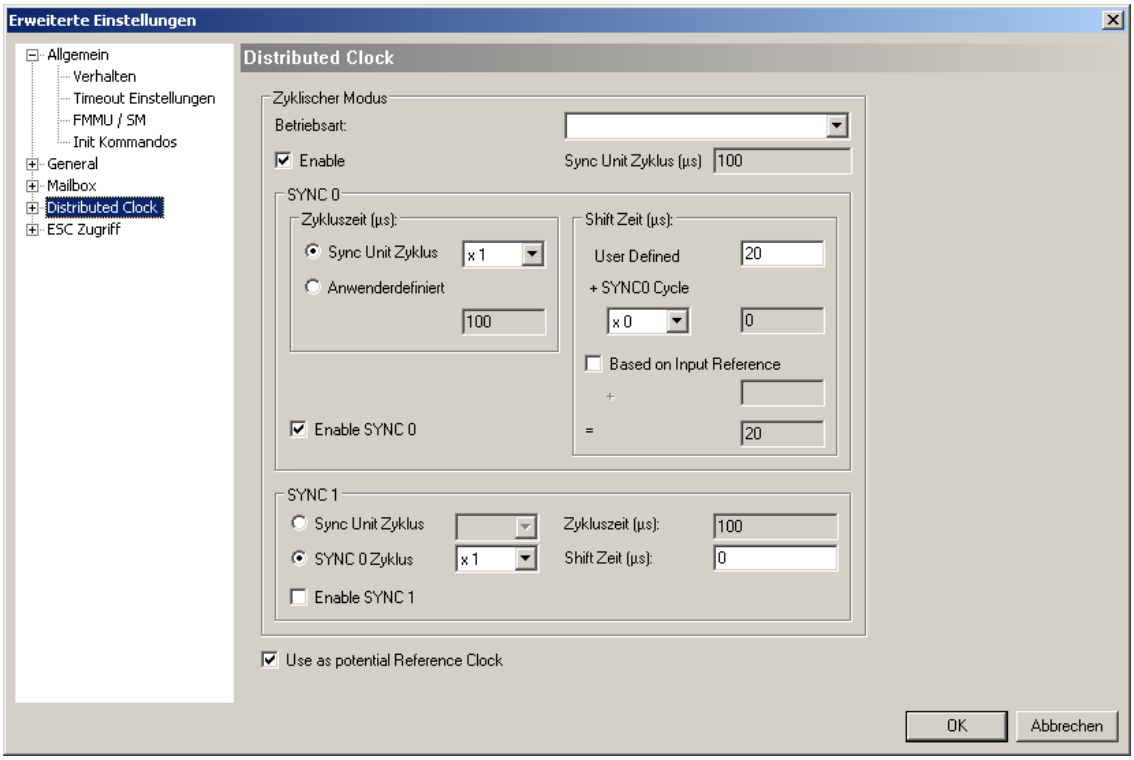


Die erweiterten Einstellungen sind ebenfalls mit ***OK*** zu bestätigen. Die neue Konfiguration muss abschließend durch den Button  ***Aktiviert Konfiguration*** neu gestartet werden. Die Konfiguration zur direkten Querkommunikation ist abgeschlossen.

Das SSB kopiert seine Messdaten nun nicht länger nur in den logischen Adressbereich 0x01000000, sondern auch in den, auch vom SDB (AVNET Eval Board) genutzten, logischen Adressbereich 0x0100000E.

5.2. Distributed Clocks – Verteilte Uhren zur Synchronisation

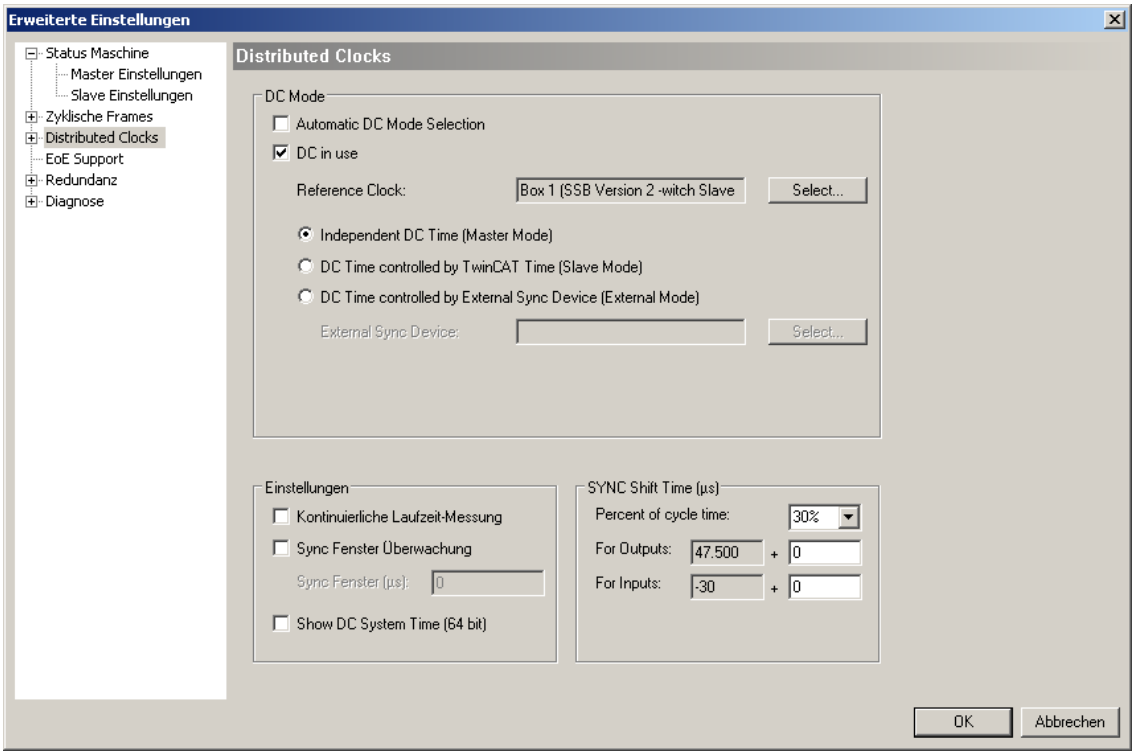
Um die Distributed Clocks (DC) zu aktivieren, muss das Netzwerk zunächst in den ***Echtzeit-Mode*** versetzt werden. Anschließend muss die Referenzuhr des Netzwerkes eingeschaltet werden. Hierfür ist der physikalisch erste Teilnehmer des Netzwerkes, der die DC-Funktion nutzt, auszuwählen. Über den Reiter ***EtherCAT*** ist das Menü der ***Erweiterten Einstellungen*** aufzurufen. Unter dem Menüpunkt ***Distributed Clock*** sind, für den Fall des SSB, das ***Enable Zyklischer Modus***, ***Enable SYNC 0*** und ***Use as potential Reference Clock*** zu aktivieren.



Das oberste ***Enable*** aktiviert die DC-Funktion. Durch das ***Enable SYNC 0*** wird die Erzeugung eines SYNC Signals freigeschaltet, welches mit Hilfe der ***Shift Zeit*** vor oder hinter das Start of Frame geschoben werden kann. Das Aktivieren des ***Use as potential Reference Clock*** legt die Referenz-Uhr des Netzwerkes fest.

Bei allen weiteren Teilnehmern, die die DC-Funktion nutzen sollen, ist identisch vorzugehen. Lediglich das ***Use as potential Reference Clock*** wird ausschließlich beim physikalisch ersten Teilnehmer des Netzwerkes aktiviert.

Anschließend wird die Konfiguration über den  -Button geladen. Abschließend ist im Master die Referenz-Uhr auszuwählen. Hierfür ist der Master (hier ***Gerät 3***) anzuwählen und über den Reiter ***EtherCAT*** das ***Erweiterte Einstellungen*** -Menü zu öffnen. Es ist wiederum der Menüpunkt ***Distributed Clocks*** anzuwählen. Die Einstellung ***Automatic DC Mode Selection*** ist zu deaktivieren. Die Einstellung ***DC in use*** ist zu aktivieren, wobei als Reference Clock der korrekte Teilnehmer (hier ***SSB***) auszuwählen ist. Als Modus wird der ***Independent DC Time Master Mode*** verwendet. Die Einstellung ist mit ***OK*** zu bestätigen und abschließend über den  -Button neu zu laden.

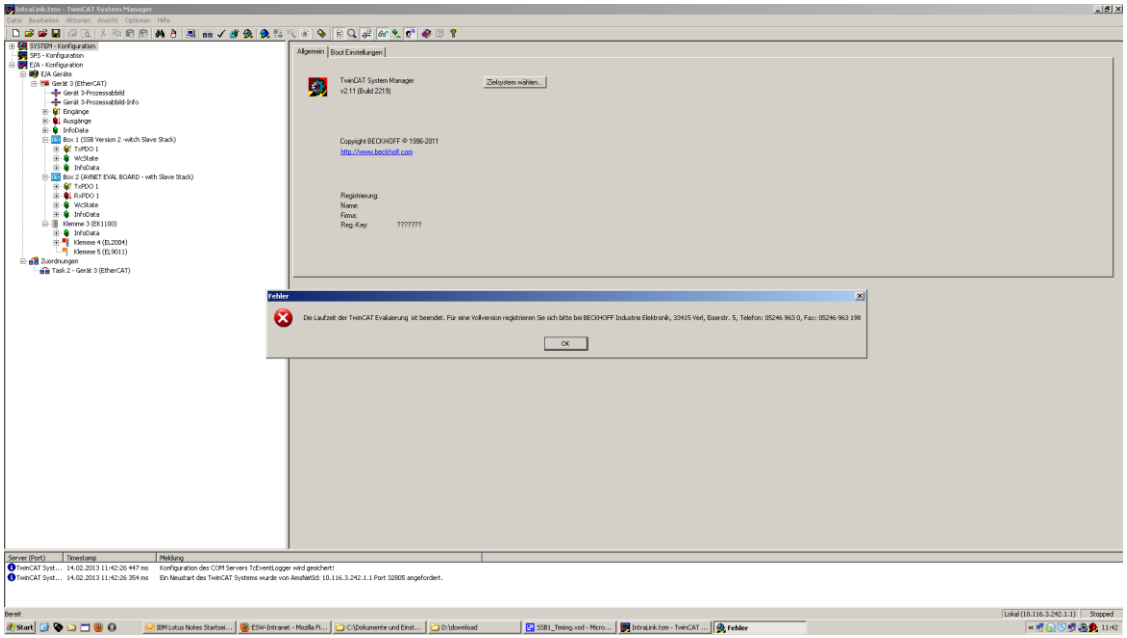


Weiterführende Informationen finden sich im Informationssystem von Beckhoff unter der Seite: http://infosys.beckhoff.com/

6. Häufig auftretende Fehler

6.1. Laufzeit der TwinCAT Evaluierung ist beendet

Der Fehler „***Laufzeit der TwinCAT Evaluierung ist beendet***“ stellt sich nach Ablauf der 30-tägigen Testversion von TwinCAT ein. TwinCAT lässt sich in diesem Fall normal starten und auch bedienen, beim Wechsel in den ***Echtzeit-Modus*** erscheint jedoch die folgende Fehlermeldung:

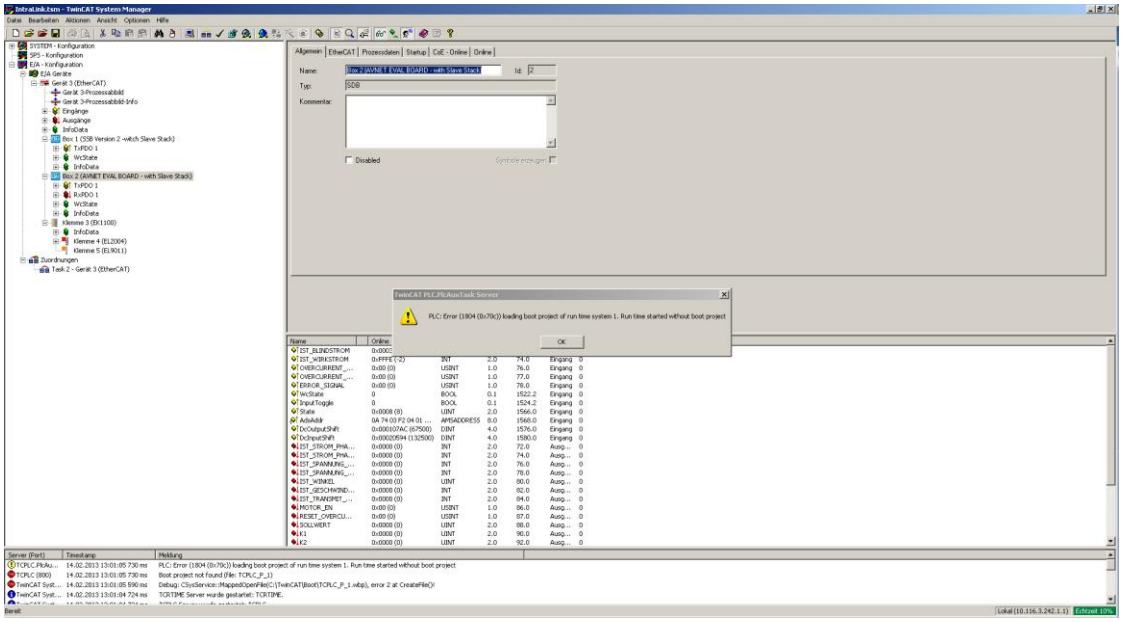


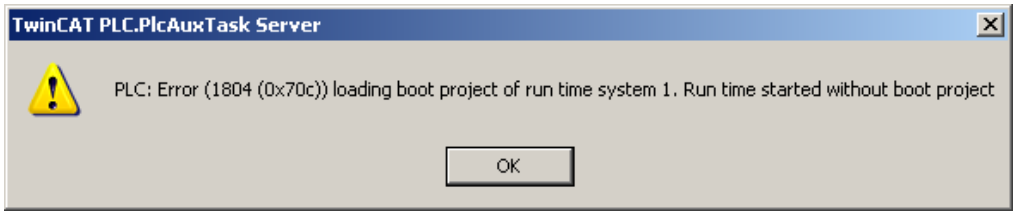
Der auftretende Fehler lässt sich lösen in dem man TwinCAT auf eine Vollversion registriert (unter Beckhoff Industrie Elektronik, 33415 Verl, Eiserstr. 5, Telefon: 05246 963 0, Fax: 05246 963 198).

Möchte man TwinCAT weiterhin als Testversion nutzen, so ist es möglich die 30-tägige Testversion erneut zu installieren. Sämtliche Projekte und Dateien bleiben hierbei erhalten.

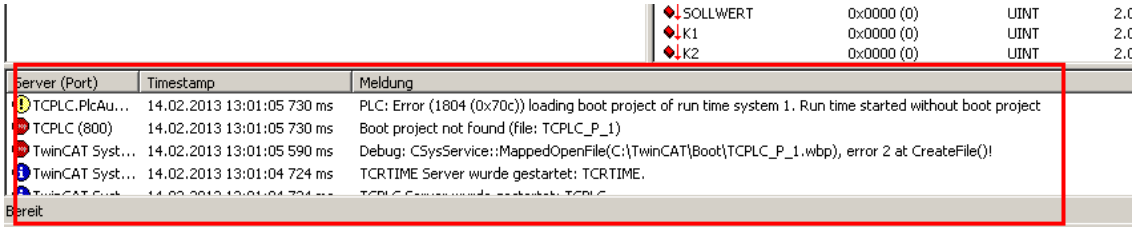
6.2. Fehler: PLC: Error (1804(0x70c)) loading boot project of run time system

Der Fehler „***PLC: Error (1804(0x70c)) loading boot project of run time system***" tritt meist nach der Neu-Installation von TwinCAT auf. Der Fehler erscheint zudem, sofern das Erstellen eines Boot-Projektes, vor dem erstmaligen Wechsel in den ***Echtzeit-Modus***, vergessen wurde. Ist kein Boot-Projekt vorhanden erscheint folgende Fehlermeldung beim Wechsel in den ***Echtzeit-Modus***:

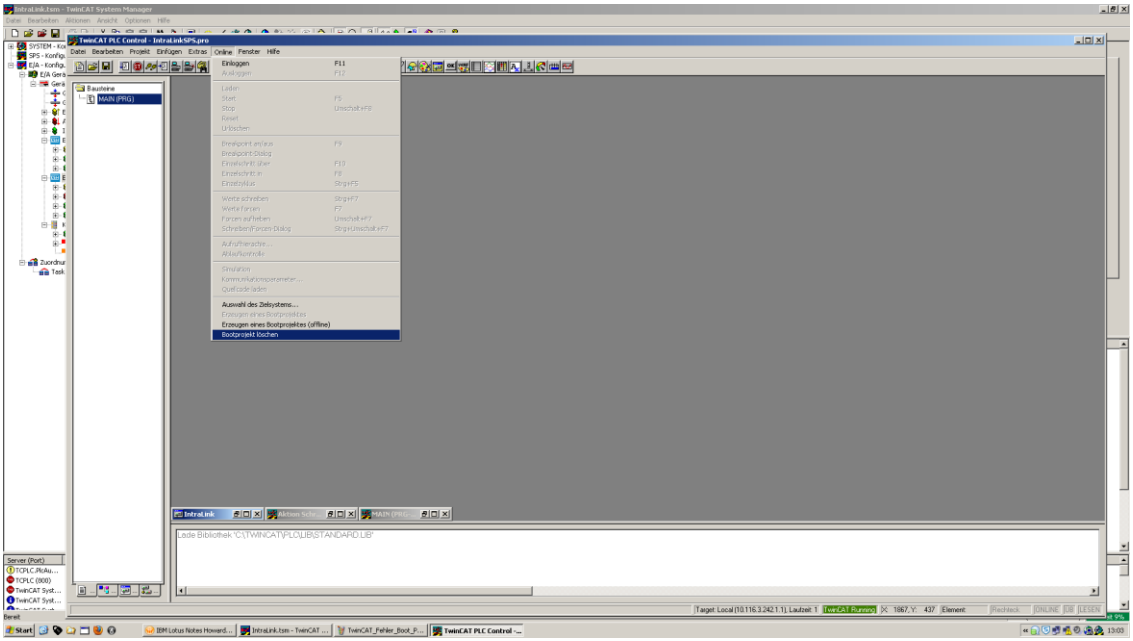




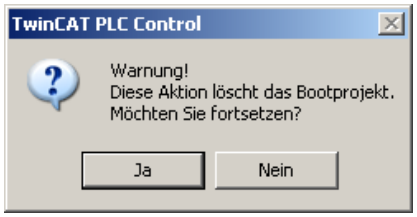
Quittiert man den Fehler mit OK so erscheinen folgende Fehler in der Kommando-Zeile:



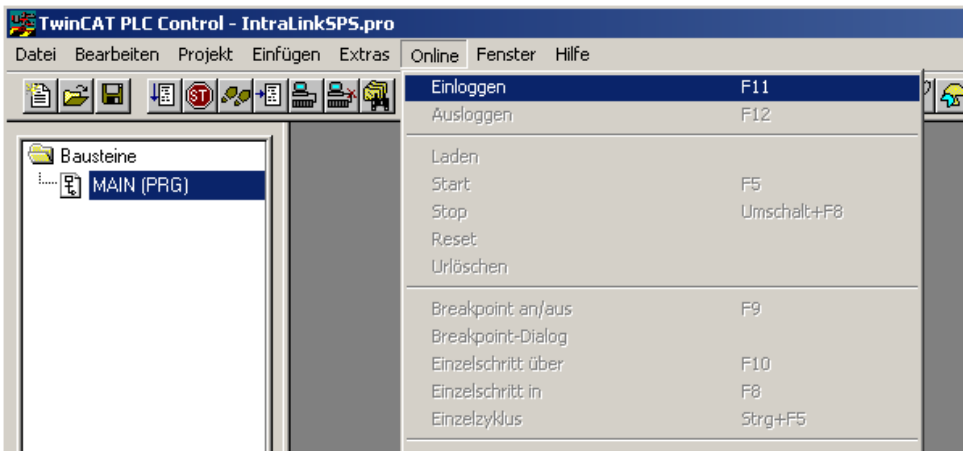
Zur Lösung des Problems ist folgender Maßen vorzugehen. Starten Sie parallel zum ***System Manager*** das ***TwinCAT PLC***. Löschen Sie das potentiell vorhandene Boot-Projekt über den Reiter ***Online*** -> ***Bootprojekt löschen***.



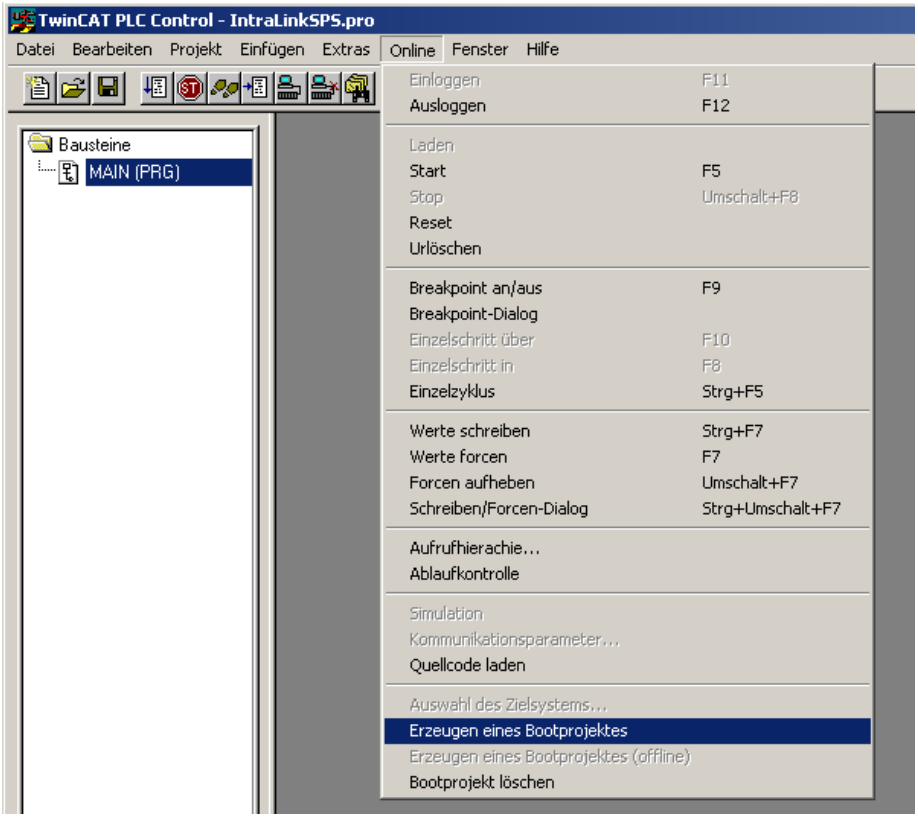
Anschließend muss das Warn-Meldung mit Ja bestätigt werden.



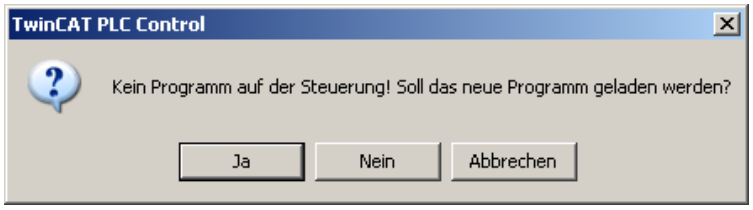
Um im Anschluss ein neues Bootprojekt zu erzeugen muss das PLC zunächst, über den Befehl ***Online*** -> ***Einloggen***, online eingeloggt werden:



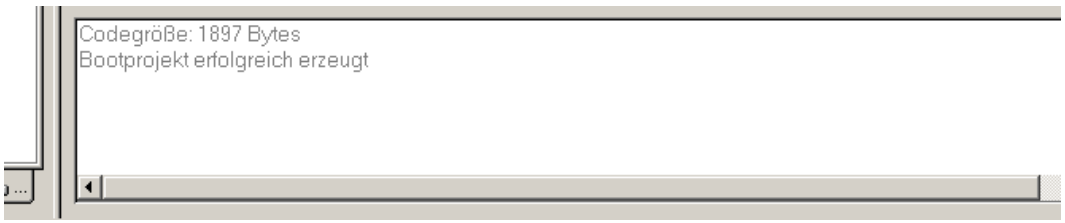
Anschließend kann, durch die Befehlskette ***Online*** -> ***Erzeugen eines Bootprojektes***, ein Bootprojekt erzeugt werden.



Folgendes Popup ist mit ***Ja*** zu bestätigen:



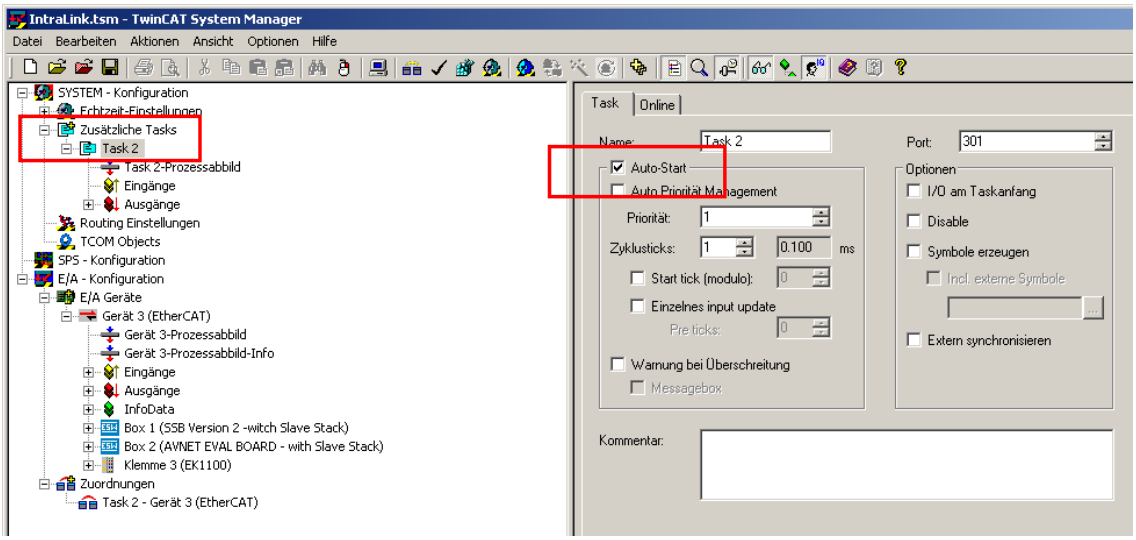
Der Vorgang ist erfolgreich abgeschlossen, sofern im Info-Bereich des ***PLCs*** folge Information eingeblendet wird:



6.3. Sämtliche Teilnehmer verharren in State PREOP

Wenn sämtliche Teilnehmer im Status Preoperational verharren und bei dem Versuch des manuellen Statuswechsels folgender Fehler auftritt:

So liegt dies meistens an der fehlenden ***Auto-Start*** Aktivierung der ***Zusatz-Task***. Zur Lösung des Problems ist die ***Auto-Start***-Funktion der ***Zusatz-Task*** zu aktivieren.

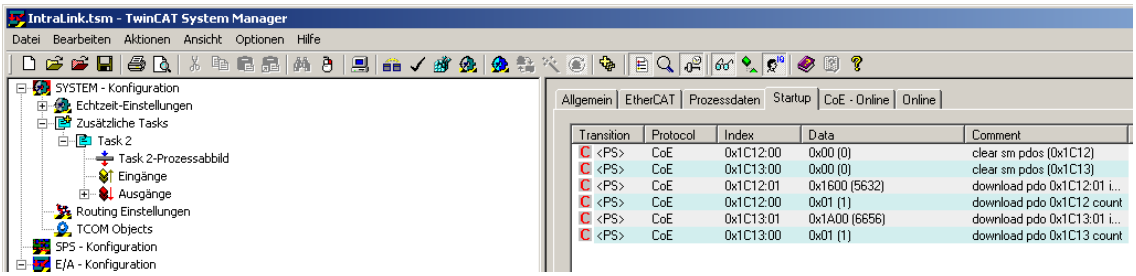


6.4. Fehlercode: 65535 Timeout: clear sm pdos (0x1C12) und state change aborted (requested 'SAFEOP', back to 'PREOP')

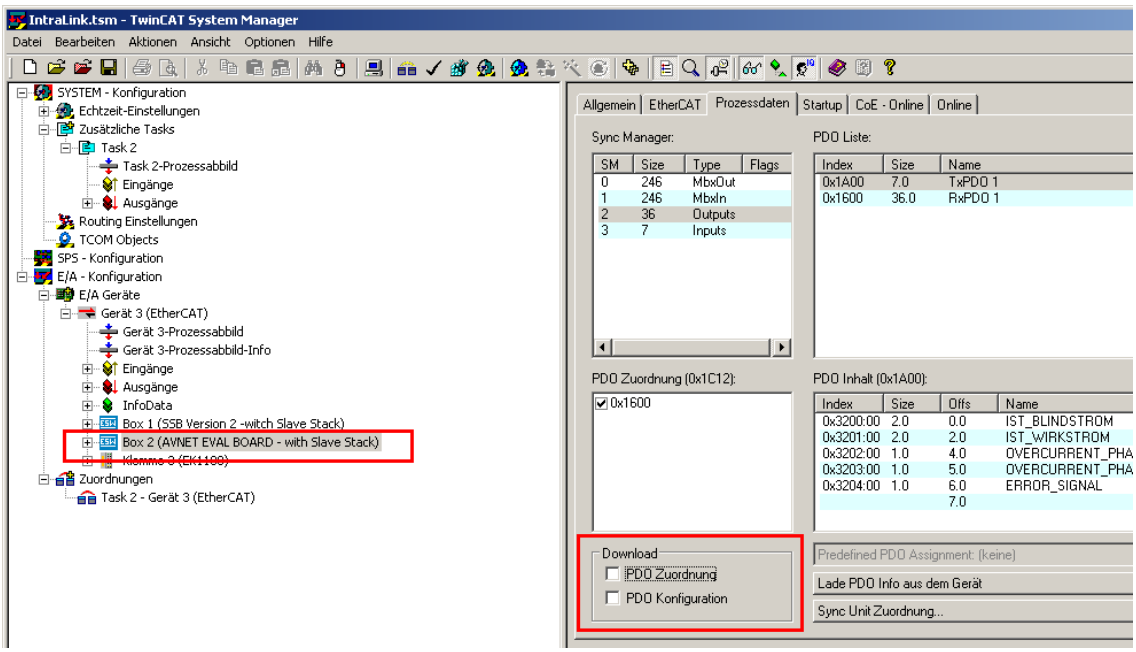
Der oben beschriebene Fehler entsteht, wenn der Master versucht Startup-Befehle an Teilnehmer zu schicken.

Server (Port)	Timestamp	Meldung
TwinCAT Syst...	18.02.2013 10:22:04 259 ms	COM Server TcEventLogger wird gestartet!
TCPLC.PlcAu...	18.02.2013 10:22:04 251 ms	PLC Download: 42 Symbols, 12 DataTypes
TCPLC.PlcAu...	18.02.2013 10:22:04 251 ms	PLC Download: 42 Symbols, 12 DataTypes
(65535)	18.02.2013 10:22:04 115 ms	'Box 2 (AVNET EVAL BOARD - ' (1002): state change aborted (requested 'SAFEOP', back to 'PREOP').
(65535)	18.02.2013 10:22:04 115 ms	'Box 2 (AVNET EVAL BOARD - ' (1002): Timeout: 'clear sm pdos (0x1C12)'.
TwinCAT Syst...	18.02.2013 10:21:58 181 ms	TCRTIME Server wurde gestartet: TCRTIME.

Ob dies der Fall ist lässt sich überprüfen, indem beim Fehler verursachenden Teilnehmer der Reiter ***Startup*** geöffnet wird.



Ist der Bereich der Startup-Befehle nicht leer (wie in Abbildung oben), so muss folgende Änderung am Teilnehmer vorgenommen werden:



Der Teilnehmer ist zu markieren, und der Reiter ***Prozessdaten*** zu öffnen. Anschließend sind sämtliche Aktivierungen im Bereich ***Download*** zu löschen. Es darf kein Haken mehr gesetzt sein. Anschließend kann die Konfiguration neu geladen werden (), der Fehler ist behoben.

Anhang 4: Parameter des Reglers

Empirisch ermittelte Parameter der Regelung; vom Anwender bei Systemstart einzutragen oder permanent in XML-File zu hinterlegen:

Parameter	$Wert_{Dez}$	$Wert_{Hex}$	Kommentar
SOLLWERT	300 - 2800	0x12C - 0xAF0	Drehzahl-Sollwert Sollwert*0,183 = Drehzahl in 1/min
K1	44225	0xACC1	P-Anteil Stromregler
K2	32356	0x7E64	I-Anteil Stromregler 1 = kein I-Anteil
RAMP_UP_DOWN	80	0x50	Zeit bis Soll-Drehzahl erreicht
MAX_CURRENT	512	0x200	Strombegrenzung 512 = 1A
K1_W	4095	0xFFF	P-Anteil Drehzahlregler
K2_W	32768	0x8000	I-Anteil Drehzahlregler 1 = kein I-Anteil

Erklärung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, den 28.02.2013 Howard Düsing