Hochschule Bochum
Bochum University
of Applied Sciences

# Bachelorarbeit

Angefertigt im Studiengang Elektrotechnik mit der Vertiefung Automatisierungstechnik an der Hochschule Bochum zur Erlangung des akademischen Grades Bachelor of Engineering

# Industrielles Testsystem für optische Velocimeter

Wintersemester 2015/2016


**Autor:** Lukas Gehrmann
lukas.gehrmann@hs-bochum.de
Matrikelnummer: 012202316

**Erstprüfer** Prof. Dr.-Ing. Arno Bergmann
**Zweitprüfer:** Dipl.-Ing. Thorsten Bartsch

**Eidesstattliche Erklärung**

# Eidesstattliche Erklärung zur Abschlussarbeit: »Industrielles Testsystem für Kamerasysteme«

„Ich versichere, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe und die Regelungen der geltenden Prüfungsordnung zu Versäumnis, Rücktritt, Täuschung und Ordnungsverstoß zur Kenntnis genommen habe."

*Unterschrift* : _____ *Ort, Datum* : _____

# Inhaltsverzeichnis

# 1 Einleitung

In der heutigen Zeit steigen zunehmend die Anforderungen an die Qualität und Quantität in der Fertigungsindustrie, aufgrund dessen werden in modernen Industrieproduktionen zahlreiche Mess- und Prüftechniken eingesetzt. Diese dienen vor allem der Überwachung und Sicherstellung der Produktqualität, aber auch zur Minimierung von Stillstandszeiten.[1]

In einer Produktionskette wird das Material oder Produkt über Förderbänder transportiert. Um sicherzustellen, dass die einzelnen Arbeitsschritte korrekt durchgeführt werden, ist es wichtig die Geschwindigkeit des Materials auf dem Transportweg in und zu den Arbeitsschritten zu überwachen. Ist diese Überwachung nicht gegeben oder fehlerhaft kann dies Minderqualität oder Ausschuss nach sich ziehen. Zusätzlich können Bauteile der Produktionsmaschinen stark verschleißen oder zerstört werden, welches eine erhöhte Stillstandszeit der Produktion zur Folge hätte.[1]

Optische Geschwindigkeitsmesser (engl. Velocimeter) bieten eine berührungslose Überwachung. Viele dieser Velocimeter erkennen, neben der genauen Ist-Geschwindigkeit, einen Stillstand oder eine Bewegungsrichtungsumkehrung. Bewegt sich das Material beispielsweise nicht gleichmäßig über das Förderband, kann durch eine korrekte Erkennung dieses Effektes im darauffolgenden Arbeitsschritt entsprechend reagiert werden.

Im Labor für elektrische Antriebe der Hochschule Bochum befindet sich ein linearer Teststand (Abbildung 1.1). Dieser dient zur Verifikation von optischen Velocimetern. Ziel ist es verschiedene dieser Sensor-Systeme an dem Teststand auf deren korrekte Funktion zu überprüfen.
Auf dem Transportschlitten können verschiedene Materialien montiert werden, je nach Einsatzgebiet des zu testenden Systems (Device under Test, DUT). Der Teststand ahmt verschiedene Bewegungen des Materials auf einem Förderband in einer Produktion nach, wie zum Beispiel Richtungswechsel mit verschiedenen Beschleunigungen und Geschwindigkeiten. Die Messdaten des DUT werden mit denen der Referenz verglichen und somit die Spezifikationen des DUT überprüft.

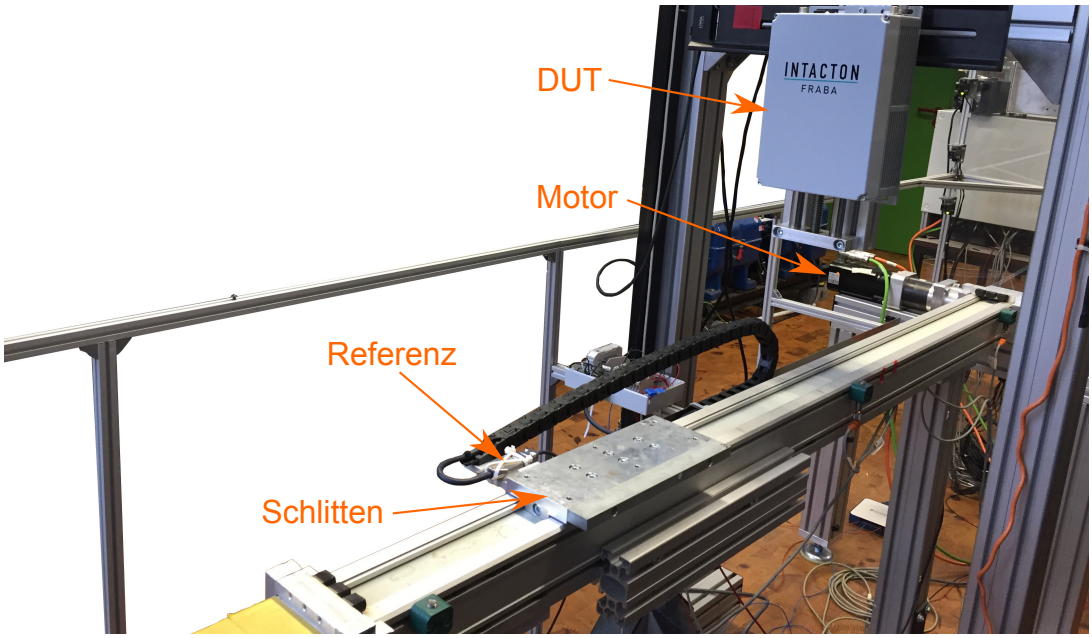



**Abbildung 1.1:** Linearer Teststand zur Verifikation von optischen Velocimetern

## 1.1 Systembeschreibung

### 1.1.1 Teststand

**Fahrprogramme:** Über eine Benutzeroberfläche (GUI) werden verschiedene Bewegungsabläufe oder auch Fahrprogramme am Computer vorgegeben. Der Servoverstärker (Ecovario 414) zum Ansteuern des Motors (80B33-0460-D08JD-AA) und der Motor selbst sind von der Firma Jenaer Antriebstechnik GmbH und setzten die vorgegebenen Fahrprogramme auf den Schlitten um (siehe Abbildung 1.1).

**Referenz:** Als Referenz dient ein inkrementeller Magnetsensor der Firma Siko (MSK 1000). In Abbildung 1.1 ist zu erkennen, dass dieser direkt am Schlitten montiert ist. Verfährt der Schlitten, bewegt sich der Magnetsensor über ein Magnetband, wodurch dieser inkrementelle Impulse erfasst und somit die Position des Schlittens bestimmt. Diese Positionsangaben in Abhängigkeit der Zeit können in Geschwindigkeitswerte umgerechnet und mit denen des DUT verglichen werden.

**DUT:** Das zu überprüfende System wird auf dem Teststand montiert. Zur Anbindung stehen verschiedene Kommunikationssysteme zur Verfügung, wie CAN, Profibus, Profinet und Ethernet. Auch ein direktes Einlesen von inkrementellen Signalen über digitale Eingänge ist möglich.

### 1.1.2 National Instruments System

Die bestehende Datenerfassung und Ansteuerung des Lineareteststands wird durch ein National Instruments (NI) System ersetzt. Das Chassis (NI PXIe-1078) ist mit vier Karten bestückt.

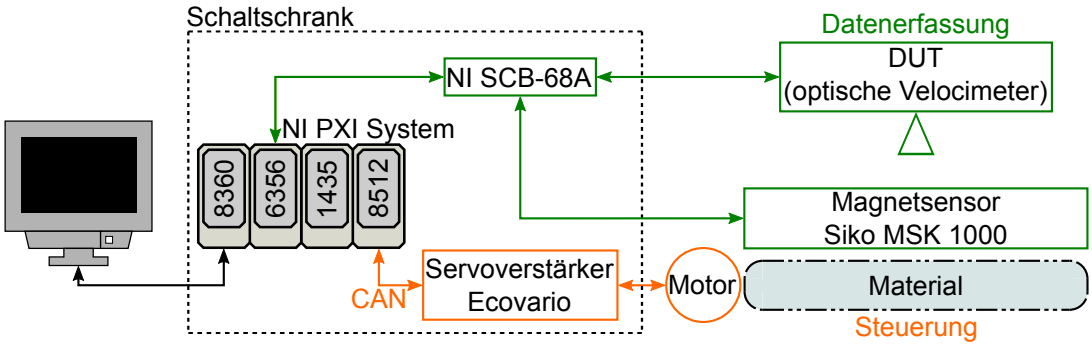



**Abbildung 1.2:** Blockdiagramm Teststand

**NI PXIe-8360:** Diese Karte dient der Kommunikation mit dem angeschlossenen Computer.

**NI PXIe-6356:** Das Zusammenspiel dieser Karte mit der NI SCB-68A Connector Box bietet zahlreiche Anschlussmöglichkeiten für Analoge und Digitale Ein- und Ausgänge [2]. Zusätzlich bietet diese vier Zähleranschlüsse, worüber die Daten der Referenz (Magnetsensor) und des zurzeit eingebaute DUT (Intacton GmbH - CVD-A1-300-RS-S4-M00) eingelesen werden.

**NI PXIe-1435:** Das System bietet mit Hilfe dieser Karte eine Möglichkeit Bildmaterial von Kamera unterstützten Testsystemen einzulesen.

**NI PXI-8512:** Die NI PXI-8512 ist die Schnittstelle zum CAN-Bus des Teststands. Über diesen wird die Ansteuerung des Servoverstärkers realisiert.

## 1.2 Ziel

Ziel dieser Arbeit ist es eine Software in LabVIEW zu implementieren. Diese umfasst das Steuern des Teststands und die Messdatenerfassung. Über die implementierte GUI werden verschiedene Fahrprogramme eingegeben und über den CAN-Bus an den Servoverstärker übertragen. Der Motor setzt die Bewegung auf den Schlitten um. Die Messdaten der Referenz und des DUT werden in LabVIEW eingelesen und zur weiteren Verarbeitung in eine gemeinsame Datei abgespeichert.

## 1.3 Motivation

Der Teststand bietet Entwicklern von optischen Velocimetern die Möglichkeit das Produkt gezielt an bestimmten Materialien und für unterschiedliche Bewegungsprofile zu testen. Optische Velocimeter werden zur Geschwindigkeits- und Längenmessung an festen Oberflächen eingesetzt. Einsatzgebiet für solche Systeme bietet vor allem die Papier, Folien, Textilien und Metallblech Produktion oder Beförderung [3].
Zusätzlich können Produktionsfirmen Sensor-Systeme, vor der Integration in neue oder modifizierte Prozessabschnitte, an diesem Teststand auf die korrekte Funktion mit dem zu bearbeitenden Materialien und der vorkommenden Bewegungsprofile prüfen. Wird ein System mit diesem Prüfstand auf eine zuverlässige Funktion im Zusammenspiel mit der Oberfläche des Materials getestet, können Ausfälle und Fehler in der Produktionskette während der Produktion vermieden werden.

Da die bestehende Datenerfassung und Ansteuerung des Teststands nicht korrekt arbeitete, sollte diese durch einen Industriestandart ersetzt werden. „LabVIEW von National Instruments gilt als die führende Programmierumgebung für Mess- und Prüfmaschinen“ [Onl1]. Durch das NI System, wird die Datenerfassung und die Ansteuerung über die Software LabVIEW realisiert.

# 2 Projektphasen

Die Umsetzung des im Abschnitt 1.2 beschriebenen einmaligen und zeitlich begrenzten Vorhabens ist die Schaffung eines neuartigen Produktes. Die Spezifikationen weisen daraufhin, dass es sich in dieser Arbeit um ein Projekt handelt [4].
Die durch die DIN festgelegten Projektmerkmale „begrenzt“, „einmalig“, „eindeutiges Ziel“ und „hohe Komplexität“ sind eindeutig zu erkennen [DIN 69901].
Bei dieser Bachelorarbeit handelt es sich um ein zeitlich „begrenztes“ und „einmaliges“ Vorhaben. Das neuartige Produkt ist die implementierte Software in LabVIEW und ein „eindeutiges Ziel“. Die „hohe Komplexität“ ist durch die verbundenen Risiken und das notwendige Wissen gegeben.
Im Folgenden werden die einzelnen Projektphasen, die den Aufbau dieser Arbeit widerspiegeln und dem Ablauf des Projektes entsprechen, beschrieben. Die Vorgehensweise ist in Abbildung 2.1 dargestellt, diese ist eine Anpassung des Standard „V-Modells“ [5].

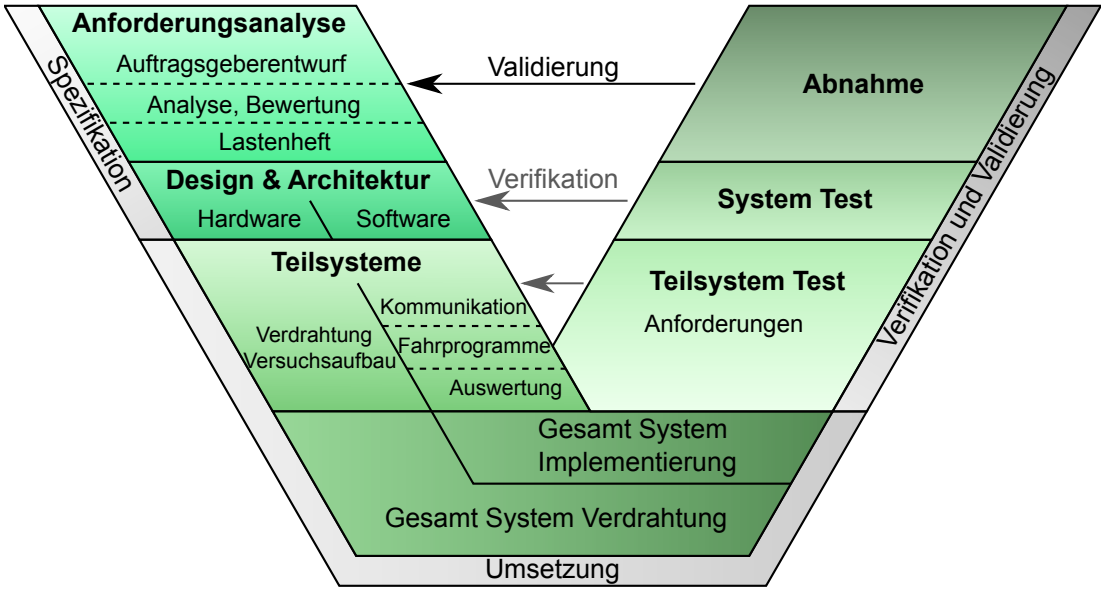



**Abbildung 2.1:** Angepasstes V-Modell

## 2.1 Spezifikation

In der Phase „Spezifikation“ sind die Anforderungen an das Projekt erfasst und dokumentiert worden. „Anforderungen beschreiben Eigenschaften, Funktionalitäten und Qualitäten an ein Projekt“ [6, S. 5]. Die genaue Erfassung der Anforderungen sorgt dafür, dass das spätere Produkt den Wünschen des Auftraggebers entspricht.
Ein erster Entwurf der Anforderungen wurde vom Auftraggeber verfasst. In enger Zusammenarbeit, in Form von Projektbesprechungen, wurden alle Anforderungen von beiden Parteien analysiert, Risiken besprochen und ein Lastenheft angefertigt. Das freigegebene Lastenheft beinhaltet alle Anforderungen an dieses Projekt und befindet sich im Anhang dieser Bachelorarbeit [7].

Jede Anforderung (ANF) hat eine eindeutige Nummer und einen Titel. Zusätzlich wurde jeder Anforderung vom Auftraggeber eine Priorität zugewiesen. Diese ist ein Richtwert dafür wie bedeutsam die Anforderung für die spätere Abnahme ist, ob sie für die spätere Gesamtfunktion unerlässlich oder nur wünschenswert ist.
Die Beschreibung einer Anforderung wurde lösungsneutral gehalten, um die spätere Implementierung nicht einzuschränken. Da die Funktionen einer Anforderung eine andere beeinträchtigen oder in Konflikt mit dieser stehen könnten, wurde dies im Lastenheft unter „Wechselwirkungen“ dokumentiert.
Wie bereits erwähnt, wurden Risiken für jede Anforderung besprochen und im Lastenheft vermerkt. Die Punkte „Testhinweis“ und „Grobschätzung des Aufwands“ wurden vom Projektleiter, als Hilfestellung für den Verifikationsplan und den Meilensteinplan zu jeder Anforderung ergänzt.

Im Bereich „Design & Architektur“ wurde die Realisierbarkeit aller Anforderungen vom Projektleiter geprüft und das Lastenheft freigegeben. Ein weiterer wichtiger Bestandteil des Lastenheftes ist der Verifikationsplan. Dieser stellt sicher, dass jede Anforderung verifizierbar ist und wird am Ende eines Projektes abgearbeitet. In Kapitel 6 wird dieses Dokument beschrieben.

## 2.2 Umsetzung

Um den Arbeitsaufwand abzuschätzen, wurde ein Meilensteinplan als Gantt-Chart aufgetragen. Dieser dient als Hilfestellung, um nicht zu viel Zeit durch Optimierung in einzelne Teilaufgaben zu investieren. Eine Anforderung kann laut Lastenheft verifizierbar sein, allerdings beliebig weit optimiert werden. Die einzelnen Meilensteintermine verhindern, das Sprengen des Projektaufwandes und das nicht Erreichen des Endtermins [4].
In dieser Phase des Projektes sind alle Anforderungen gelöst und als Gesamtsystem implementiert worden. Alle Lösungen, der im Lastenheft beschriebenen Anforderungen sind im Kapitel 5 beschrieben.
Durch die Aufteilung des Gesamtsystems in Teilsysteme, ist es schon in dieser Phase möglich einzelne Teilstrukturen zu Testen. Somit können Fehler in einer frühen Projektphase erkannt und behoben werden.

## 2.3 Verifikation und Validierung

In dieser Projektphase wurden die einzelnen Teilsysteme und Anforderungen mit Hilfe des Verifikationsplans getestet, daraufhin erfolgte ein Gesamtsystemtest. In diesem wurde vor allem das Zusammenspiel aller Teilfunktionen vor Ort geprüft.

Das Projektende bildete die Endabnahme durch den Auftraggeber. Die Validierung entscheidet über die Qualität des Projektes, im schlimmsten Fall wird in diesem Schritt auch ein Scheitern des Projektes festgelegt [4].

# 3 Technische Grundlagen

In diesem Kapitel sind die für diese Arbeit relevanten technischen Grundlagen aufgeführt. Abschnitt 3.1 beschreibt den verwendeten CAN-Bus zur Kommunikation zwischen NI System und dem Servoverstärker. Diese Grundlagen werden für das Verständnis in Bezug auf die Hierarchie der Knoten und der Priorisierung von Nachrichten benötigt.
Zusätzlich wird die Erweiterung des vom Servoverstärker verwendeten CanOpen Protokolls beschrieben, insbesondere die verschiedene Kommunikationsmechanismen, die in der Implementierung verwendet wurden.
In Abschnitt 3.2 werden die unterschiedlichen Signale zur Datenerfassung beschrieben.

## 3.1 Controller-Area-Network

Der CAN-Bus (**C**ontroller-**A**rea-**N**etwork) ist ein Feldbussystem und dient zur Kommunikation zwischen einzelnen Komponenten in einem technischen System. Ein technisches System ist die Zusammenfassung verschiedener Komponenten in einer Maschine oder Anlage [8]. Dabei besitzt jede Komponente eine Schnittstelle um Informationen zu senden und zu empfangen.
Die in dieser Arbeit verwendete lineare Busstruktur ist eine bevorzugte Topologie bei Feldbussen, wobei die einzelnen Komponenten auch als Knoten bezeichnet werden [9]. In Abbildung 3.1 ist diese Struktur beispielhaft mit drei Komponenten abgebildet. Die einzelnen Busteilnehmer werden über kurze Stichleitungen an die Bus-Datenleitung angeschlossen [Onl2]. Dies ermöglicht eine einfache Erweiterung des Systems mit zusätzlichen Komponenten.

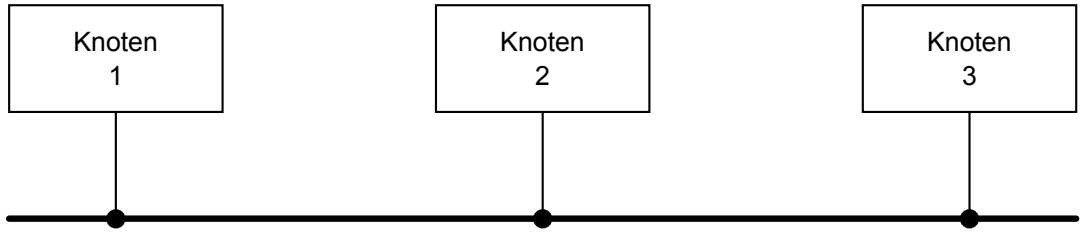



**Abbildung 3.1:** Lineare Bus-Struktur

### 3.1.1 ISO/OSI-Schichtenmodell

„Basis für die Beschreibung von Kommunikationssystemen ist heute allgemein das von der ISO (International Standardization Organization) entwickelte Modell der Datenkommunikation (OSI(Open-Systems-Interconnection)-Referenzmodell).“ [10, S. 3] Dieses Modell besitzt sieben Schichten (Layer) mit ihren eigenen international anerkannten Funktionen [ISO 7498-1]. Im Feldbusbereich werden häufig nur Schicht 1, 2 und 7 verwendet [10].

Kurzbeschreibung der Aufgaben der Schichten [9][ISO 7498-1]:

1: Bitübertragungsschicht (Physical Layer)
Die Bitübertragungsschicht spezifiziert die elektrischen und mechanischen Eigenschaften der Schnittstelle zum Übertragungsmedium. Elektrische Eigenschaften sind zum Beispiel Spannungspegel und Signalform, mechanische Eigenschaften sind Kabelart und Pinbelegung.

2: Sicherungsschicht (Data Link Layer)
Die Sicherungsschicht ist für die zuverlässige Verbindung zwischen Übertragungsmedium (Bus-Datenleitung) und der einzelnen Busteilnehmer zuständig. Verschiedene Zugriffsverfahren werden eingesetzt um Übertragungsfehler und Datenverluste zu erkennen und zu beheben.

7: Anwendungsschicht (Application Layer)
Diese Schicht stellt die Verbindung zwischen Anwendungsprogramm und Kommunikationsdienst dar. Der Anwender bekommt lediglich Statusinformationen der Kommunikationsdienste. Diese Instanz dient zur Daten Ein- und Ausgabe.

### 3.1.2 CAN

Das CAN Protokoll wurde Anfang der 80er Jahre von der Firma BOSCH entwickelt und 1993 als Standard „*Road vehicles - Controller area network (CAN)* [IS0 11898]“ genormt. Ursprünglich war das CAN Protokoll für den Einsatz in Kraftfahrzeugen angedacht um die Komplexität des Kabelbaumes in Fahrzeugen zu reduzieren [Onl3]. Der Einsatzbereich dieses Protokolls ist heutzutage vielseitiger. Zum Beispiel ist es in der Industrie im Bereich der Automatisierungstechnik wie z.B. in intelligente Motorsteuerungen stark verbreitet [9].

Im Bezug auf Abschnitt 3.1.1 implementiert CAN die unteren beiden Schichten (Physical Layer & Data Link Layer) des ISO/OSI Schichtenmodels.
In Part 1 des Standards [IS0 11898] („*Data link layer and physical signalling*“) werden die Schichten beschrieben. Der CAN-Bus wird durch eine verdrillte zwei Draht Leitung (Twisted-Pair-Leitung) mit Abschlusswiderständen realisiert (Abbildung 3.2). Die 120 Ohm Abschlusswiderstände werden zur Vermeidung von Reflexionen an beiden Enden des Busses eingesetzt [10].

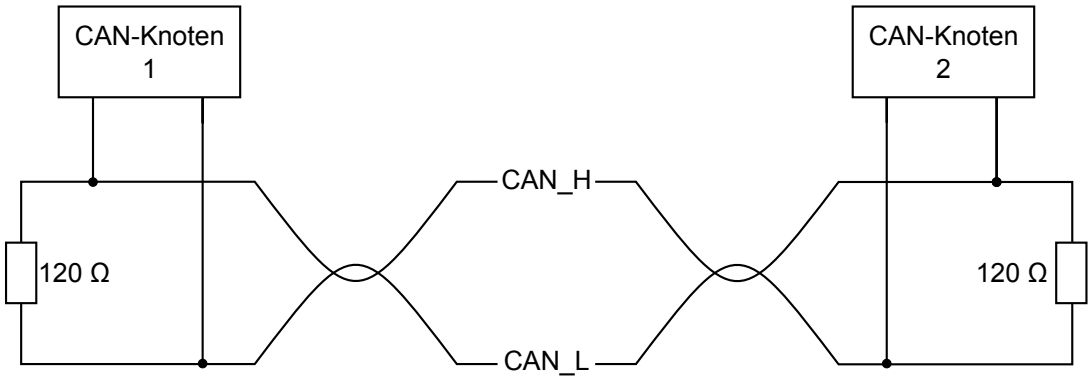



**Abbildung 3.2:** CAN-Bus mit Twisted-Pair-Leitung und Abschlusswiderständen

In Part 2 des Standards [IS0 11898] (*High-speed medium acces unit*) und Part 3 (*Low-speed, fault-tolerant, medium-dependent interface*) werden die beiden Varianten High- und Low-Speed CAN beschrieben. Die in dieser Arbeit verwendeten Hardwarekomponenten unterstützen den schnelleren High-Speed CAN, dieser unterstützt eine Datenrate von bis zu 1 MBit/s.

#### 3.1.2.1 Frame-Aufbau

Die zweite Schicht beinhaltet den Frame-Aufbau, den Netzzugriff sowie die Fehlererkennung und -behandlung. Jeder Frame hat eine festgelegte Anzahl von Bits. Diese Bits sind auf bestimmte Felder aufgeteilt [10].

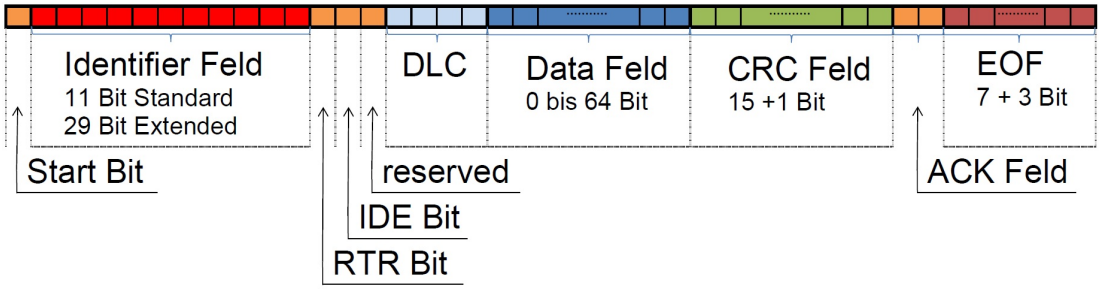



**Abbildung 3.3:** Aufbau des CAN-Dataframe [11]

Abbildung 3.3 zeigt beispielhaft ein Dataframe. Nach dem Start Bit SOF (**S**tart **o**f **F**rame) folgt das Identifier Feld.
In einem CAN-Bus ließt jeder Knoten jede Nachricht die auf dem Bus anliegt. Mit Hilfe des Identifier entscheidet jeder Knoten für sich, ob die Nachricht für ihn relevant ist. Es gibt zwei Identifier Varianten, der Standard Identifier ist 11 Bit lang und kann somit 2048 verschiedene Adressen abbilden. Sollte dies nicht ausreichen verwendet man den Extended Identifier. Dieser wird aus 29 Bit gebildet, damit sind $2^{29}$ verschiedene Telegramme möglich.
Ob der Frame selbst Daten enthält (RTR = low) oder die aktuellen Daten anfordert beschreibt das **R**emote-**T**ransmission-**R**equest-Bit (RTR Bit).
Das anschließende Control Feld besteht aus dem IDE-Bit (**Id**entifier **E**xtension), ein Reservebit und der DLC (**D**ata **L**enght **C**ode). Das IDE-Bit ist im Standard Frame low und gibt damit an, dass nachfolgend kein Extended Identifier folgt. Im DLC wird die Länge des Datenfeldes angegeben. Die Daten können eine Länge von Null bis Acht Byte haben (Data Feld) [9].
Zur Fehlererkennung wird die CRC-Prüfsumme verwendet (15 Bit). Diese endet mit dem CRC-Delimiter Bit (high).

Das Bestätigungsfeld oder auch **Ack**nowledge Field (ACK) besteht aus zwei Bits, mit dem hinteren Delimiter-Bit (high). Das vordere ACK-Slot-Bit wird zur Bestätigung der Nachricht verwendet. Vom Sender wird hier ein rezessives high Bit gesendet. Im CAN ist die logische 1 rezessiv und die logische 0 dominant. Alle Empfänger, die die Nachricht fehlerfrei erkannt haben, legen hier einen dominanten „low“ Pegel auf den Bus, der den vom Sender gesendeten „high“ Pegel überschreibt. Somit erkennt der Sender, dass mindestens ein Teilnehmer die Nachricht fehlerfrei empfangen hat [10].
Das Ende des Telegrammes signalisiert der **E**nd **o**f **F**rame (EOF).

### 3.1.2.2 Arbitrierung

Um Kollisionen auf dem Bus zu vermeiden verwendet CAN das Zugriffsverfahren CSMA/CD + AMP (**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection and **A**rbitration on **M**essage **P**riority). Das Arbitrations Feld besteht aus dem Identifier und dem RTR Bit. Da der logische 0 Pegel im CAN dominant ist, setzt sich der Identifier mit der niedrigsten Wertigkeit durch. In Abbildung 3.4 senden Station A, B und C gleichzeitig.

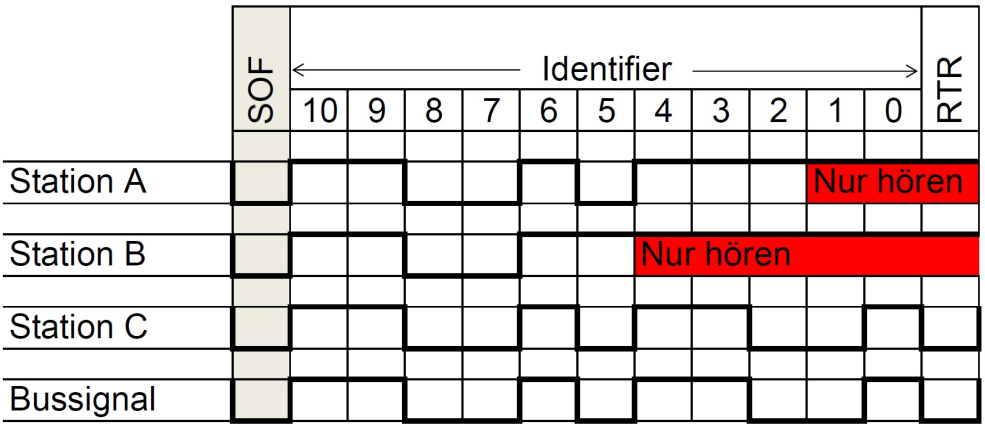



**Abbildung 3.4:** Beispiel der Arbitrierung [11]

Es werden Bit für Bit die Pegel der einzelnen Stationen verglichen. An der fünften Stelle des Identifiers sendet Station B eine logische 1, A und C dominieren durch eine logische 0. Ab diesem Zeitpunkt sendet Station B nicht weiter.
An der zweiten Stelle des Identifiers setzt sich Station C auch gegen Station A durch. Das Bussignal sendet die Daten von Station C. Sobald der Bus in Ruhezustand versetzt wird beginnen A und B erneut den Frame zu senden [12].

### 3.1.3 CANopen

Die CANopen-Spezifikationen sind durch den Verein ***C**AN **i**n **A**utomation* (CiA) verfasst worden. In diesen Spezifikationen sind alle Funktionen von CANopen eindeutig beschrieben. Das CANopen Protokoll definiert in erster Hinsicht die Anwendungsschicht (Schicht 7) des ISO/OSI-Schichtenmodells (vgl. Abschnitt 3.1.1).

Das Kommunikationsprofil eines CANopen-Gerätes ist im *„CiA Darft Standard 301“* [CiA 301] beschrieben. In diesem Profil sind verschiedene Mechanismen zum Austausch von Daten vorgesehen (vgl. Abschnitt 3.1.3.1).
Der Kern eines CANopen-Gerätes ist das Objektverzeichnis, welches das bereits realisierte Objektmodell in der zweiten Schicht des ISO/OSI-Schichtenmodells ausnutzt [11]. Dieses in Tabellenform organisierte Dokument beschreibt den kompletten Funktionsumfang des Gerätes [13]. Jedes Objekt setzt sich aus einem Index und einem Sub-Index zusammen und ist somit klar definiert.
Der im Zuge dieser Arbeit verwendete Servoverstärker (Ecovario 414) verwendet die von der CiA vorgegeben Identifier um die Kommunikationsart einer Nachricht eindeutig zu definieren (Abschnitt 3.1.3.1). Die acht Datenbytes des CAN-Frames (vgl. Abschnitt 3.1.2.1) werden für die Übertragung des Objektes, der Zugriffsart als auch für die eigentlichen Daten verwendet.
In der Spezifikation *„CiA Darft Recommendation DR-303-1“* [CiA 303] wird zusätzlich die erste Schicht des ISO/OSI-Schichtenmodells (vgl. 3.1.1 ISO/OSI-Schichtenmodell) beschrieben. Abbildung 3.5 und Tabelle 3.1 zeigen Beispielhaft die Pinbelegung eines neun Poligen Steckers die in der Spezifikation [CiA 303] definiert ist.

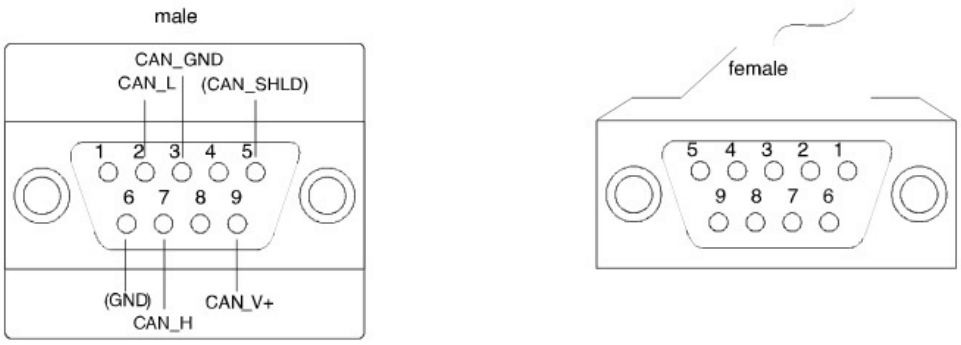



**Abbildung 3.5:** 9-pin D-Sub connector [CiA 303, S. 9]

| Pin | Signal | Description |
|---|---|---|
| 1 | - | Reserved |
| 2 | CAN_L | CAN_L bus line |
| 3 | CAN_GND | CAN Ground |
| 4 | - | Reserved |
| 5 | (CAN_SHILD) | Optional CAN Shield |
| 6 | (GND) | Optional Ground |
| 7 | CAN_H | CAN_H bus line |
| 8 | - | Reserved |
| 9 | (CAN_V+) | Optional CAN external positive supply |

**Tabelle 3.1:** 9-pin D-Sub connector [CiA 303, S. 9]

Die Datenleitung liegt an Pin 2 und 7. Zwischen diesen muss der 120 Ohm Abschlusswiederstand eingelötet werden, falls es im Gerät selbst keine Terminierung gibt. CANopen sieht optional eine Spannungsversorgung an Pin 9 und 3 vor, falls diese nicht von einem Busteilnehmer zur Verfügung gestellt wird. Mit denen in dieser Arbeit verwendeten Komponenten muss solch eine externe Versorgung realisiert werden. An Pin 5 kann optional die Schirmung der Leitung angeschlossen werden.

### 3.1.3.1 Identifier

Im Folgenden werden vier wichtige Kommunikationsmechanismen, die in der Spezifikation [CiA 301] definiert sind, vorgestellt. CANopen bietet noch weitere, allerdings werden nur diese vier in dieser Arbeit verwendet.
Das SDO (**S**ervice **D**ata **O**bject) ist eine quittierte Form der Datenübermittlung. Sowohl Schreibzugriffe als auch Lesezugriffe werden von der übergeordneten Steuerung getätigt. Die Steuerung erhält vom jeweiligen Teilnehmer eine Bestätigung. Das PDO (**P**rocess **D**ata **O**bject) ist eine nicht quittierte Datenübermittlung. Diese beiden Formen werden in Abschnitt 3.1.3.2 SDO und 3.1.3.3 PDO ausführlich beschrieben. Fehlermeldungen werden in so genannten **Em**ergen**cy** Messages (EMCY) übermittelt. Das **N**etwork **M**anagemen**t** (NMT) ermöglicht ein gleichzeitiges Ansteuern aller CANopen-Geräte und gibt den Kommunikationszustand vor.
Diese Kommunikationsobjekte (engl. **C**ommunication **Ob**ject, COB) sind Teil des Identifier (COB-ID) und definieren somit jede Nachricht mit einer Übertragungsform. Die COB-ID setzt sich aus der Summe des Kommunikationsobjektes (COB) und der Knotenadresse (ID) des Teilnehmers zusammen [14].

| **Objekt** | **Resultierende COB-IDs** |
|---|---|
| NMT | 0x00 |
| Emergency | 0x80 + Knotenadresse |
| Tx-PDO1 | 0x180 + Knotenadresse |
| Rx-PDO1 | 0x200 + Knotenadresse |
| Tx-PDO2 | 0x280 + Knotenadresse |
| Rx-PDO2 | 0x300 + Knotenadresse |
| Tx-PDO3 | 0x380 + Knotenadresse |
| Rx-PDO3 | 0x400 + Knotenadresse |
| Tx-PDO4 | 0x480 + Knotenadresse |
| Rx-PDO4 | 0x500 + Knotenadresse |
| Tx-SDO | 0x580 + Knotenadresse |
| Rx-SDO | 0x600 + Knotenadresse |

**Tabelle 3.2:** COB-ID aus Sicht des Servoverstärkers [14]

Wie bereits in Abschnitt 3.1.2.2 beschrieben wird die Nachricht mit dem niederwertigsten Identifier bei Konflikt zuerst gesendet. So ergibt sich eine Priorisierung nach Tabelle 3.2 von oben nach unten.

### 3.1.3.2 SDO

Wie bereits erwähnt sind Service Data Objects eine quittierte Form der Datenübermittlung und müssen mit dem für diese Übertragungsart vorgesehenen Identifier versehen werden. Die acht Daten Bytes des CAN-Frames (vgl. 3.1.2.1 Frame-Aufbau) teilen sich in vier Bereiche auf.
In Byte 0 der gesendeten Nachricht (CMD, **C**o**m**man**d**s) wird die Zugriffsart festgelegt. Es werden zwei Zugriffsarten unterschieden. Die erste ist der Schreibzugriff, in dem Daten in ein Objekt im Zielgerät des Busses geschrieben werden. Der Lesezugriff ist die zweite Zugriffsart und fordert die Daten eines bestimmten Objekts an. Das Byte 0 der Antwort wird RES (**Res**ponse) genannt und gibt an ob die Daten erfolgreich übermittelt wurden oder ob ein Fehler aufgetreten ist. In Tabelle 3.3 werden die verschiedenen Wertigkeiten des CMD-Byte und RES-Byte erläutert.

| **CMD** | **Bedeutung** |
|---|---|
| 0x23 | Senden von 4 Byte Daten |
| 0x2B | Senden von 2 Byte Daten |
| 0x2F | Senden von 1 Byte Daten |
| 0x40 | Lesezugriff |
| **RES** | **Bedeutung** |
| 0x60 | Daten erfolgreich gesendet |
| 0x80 | Fehlermeldung |
| 0x43 | Byte 4 -7 enthalten Daten |
| 0x4B | Byte 4 und 5 enthalten Daten |
| 0x4F | Byte 4 enthält Daten |

**Tabelle 3.3:** Wertigkeit des CMD- und RES-Byte [14]

In Byte 1 bis 3 wird das Objekt mit dem Index (Byte 1 + 2) und dem Sub-Index (Byte 3) übergeben. Die letzten vier Bytes werden für die Übermittlung von Daten verwendet, im Lesezugriff sind diese leer. Abbildung 3.6 zeigt die vier Bereiche in den Daten des CAN-Frames [14]

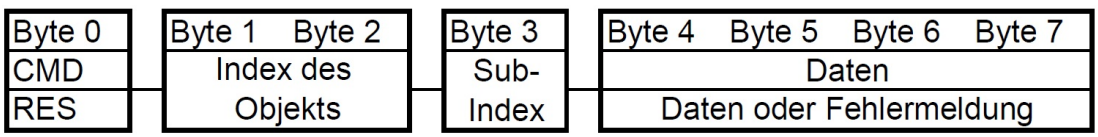



**Abbildung 3.6:** Datenbereichsaufteilung des SDO Zugriffs

### 3.1.3.3 PDO

Eine andere Form der Datenübertragung bietet das Process Data Object. Da es sich hierbei um eine unbestätigte Übertragung handelt, wird die Busauslastung gegenüber dem SDO stark reduziert. Zusätzlich können alle acht Daten-Bytes genutzt werden. Die Protokollinformationen Zugriffsart, Objekt Index und Sub-Index entfallen [14]. Je nach Größe der Objektdaten ermöglicht dies eine Übertragung von bis zu acht Objekten in einer Nachricht.
Die Übertragung eines PDO wird entweder synchron oder durch ein Ereignis durchgeführt [9, S. 206]. Die synchrone Übertragung wird in dieser Arbeit nicht behandelt. Ein Ereignis kann zum Beispiel ein Zustandswechsel eines digitalen Eingangs, ein intern ablaufender Timer oder eine Veränderung von Objektdaten sein [13]. Damit nicht bei jedem Zustandswechsel eine Übertragung erfolgt, gibt es die Möglichkeit einen Versendezeitraum einzustellen (Inhibit Time).
Wird zum Beispiel bei Veränderung der Ist-Position eines Schlittens ein PDO gesendet, würde dies beim Verfahren des Schlittens zu einer Unmenge von Daten führen. Über den Versendezeitraum ist es möglich eine Zeit vorzugeben, in welchem Abstand die PDOs gesendet werden.
Um ein PDO eindeutig zu parametrieren muss der Identifier des zu verwendenden PDO, der PDO Typ und der Versendezeitraum angegeben werden [14].

#### PDO-Mapping

Da keine Protokollinformationen übertragen werden, muss vorher festgelegt werden welche Objektdaten in einem PDO vorhanden sind. Dies nennt man PDO-Mapping und wird im für das jeweilige PDO vorgesehene Mappingobjekt durchgeführt.

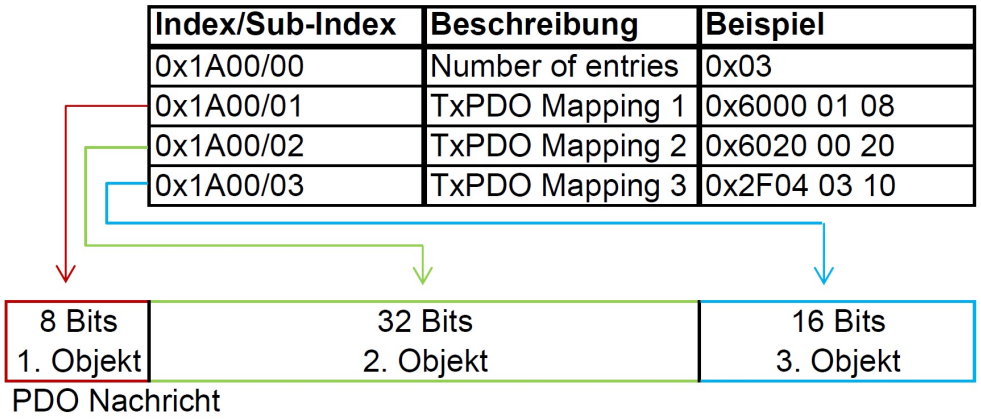



**Abbildung 3.7:** Beispiel für das PDO-Mapping

Abbildung 3.7 zeigt wie Objekte in ein PDO eingefügt werden. In diesem Beispiel wird in das Mappingobjekt mit dem Index „0x1A00“ des Tx-PDO1 drei Objekte gemappt.

Dazu wird zum Index des Objektes noch der Sub-Index und die Größe der Objektdaten angegeben. Während durch den Sub-Index die Reihenfolge der Daten in dem PDO angegeben wird, wird die Größenangabe an das Objekt angehängt (0x08 = 8 Bit, 0x10 = 16 Bit, 0x20 = 32 Bit Daten).
Um die Übertragungsart des PDO zu nutzen, muss über das Netzwerkmanagement (NMT) der Kommunikationszustand in „operational“ gebracht werden [14].

## 3.2 Inkremental-Encoder

Inkremental-Encoder werden zum Beispiel zur Positionserfassung von rotativ oder linear beweglichen Teilen verwendet. Im Gegensatz zu Absolut-Encodern liefern die Inkrementalen keinen Positionswert, sondern Impulse [15].
Ein Inkremental-Encoder wird an eine übergeordnete Steuerung angeschlossen, diese zählt die vom Encoder übertragenen Impulse. Über die Auflösung kann die Position ermittelt werden.
Die Auflösung beschreibt, wie viel Inkremente oder Impulse pro Längeneinheit oder Winkel vom Encoder ausgegeben werden. Besitz ein linearer Inkremental-Encoder beispielsweise eine Auflösung von 1 µm, gibt er 1000 Inkremente pro Millimeter aus (1000 inc/mm). Die Steuerung zählt ab einer vorgegebenen Anfangsposition die Impulse und kann somit die Ist-Position bestimmen.
Es gibt verschiedene Ausgangssignaltypen der Impulse, im Folgenden werden zwei dieser Typen beschrieben.

**Differenzielles Signal:**

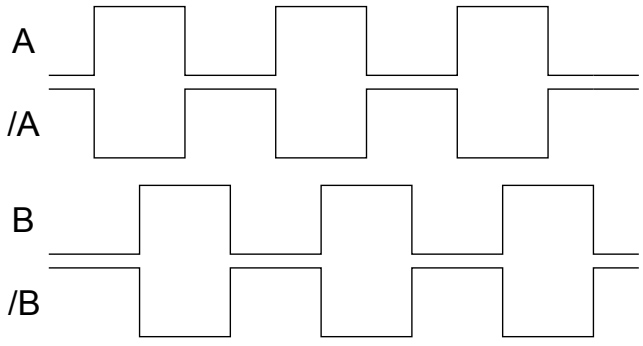



**Abbildung 3.8:** Differenzielles Signal eines Inkremental-Encoders [16]

Bei diesem Signaltyp werden zwei Signale „A“ und „B“ übertragen und zusätzlich jeweils das invertiert Signal „/A“ und „/B“. Die beiden Signale („A“ und „B“) sind um 90° zueinander Phasenverschoben [15].

Die Bewegungsrichtung, im oder gegen den Uhrzeigersinn, linear positiv oder negativ wird durch die Anordnung der Signale übermittelt. Entweder eilt „A“ neunzig Grad „B“ voraus oder umgekehrt. Durch diese Unterscheidung kann die übergeordnete Steuerung zusätzlich zum zurückgelegten Weg auch die Bewegungsrichtung erkennen.

Dieser Signaltyp ist sehr störunempfindlich. Die Steuerung erfasst die Differenz zwischen „A“ und „/A“ bzw. „B“ und „/B“. Sollte durch äußere Einflüsse ein Störsignal die Leitung beeinflussen, wirkt sich dieses auf beide Signale gleichermaßen aus und die Differenz bleibt gleich. Dieser Vierleiter Anschluss wird als RS422 bezeichnet [16].

**Masse bezogenes Signal:**

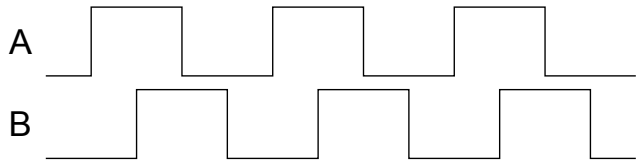



**Abbildung 3.9:** Masse bezogenes Signal eines Inkremental-Encoders

Bei diesem Signaltyp werden nur drei Leitungen benötigt („A“, „B“ und „GND“) und wird auch TTL-Signal genannt. Die Bewegungsrichtung wird wieder über die Anordnung erkannt, also welcher der beiden Signale voreilt. Allerdings erfasst die Steuerung den Spannungspegel zwischen Signal („A“, „B“) und „GND“, aufgrund dessen ist dieser Signaltyp sehr viel störungsanfälliger als der Differenzielle.

# 4 Hardware und Software

In diesem Kapitel sind die eingesetzten Hardware- und Softwarekomponenten sowie deren korrekte Konfiguration beschrieben.
Voraussetzung hierfür ist nicht nur die korrekte Installation der Hardware sondern auch der mitgelieferten Software und benötigten Treiber.

# 4.1 Hardware

Der lineare Teststand ist bereits in einer fertiggestellten Bachelorarbeit geplant worden. Der Schaltplan der Anlage und die Verdrahtung der Hardware waren bereits vorhanden. Allerdings mussten die einzelnen Komponenten an das NI System angeschlossen werden. Diese Erweiterung der Verdrahtung und des Schaltplans ist im Folgenden beschrieben.

## 4.1.1 CAN-Bus

Die CAN-Bus Karte des NI Systems ist mit dem Servoverstärker verbunden. Die Pinbelegung wurde bereits in den Grundlagen erläutert (vgl. Abschnitt 3.1.3) und ist in Abbildung 4.1 abgebildet.
Der Bus wurde im Schaltschrank auf Klemmen gelegt. Über diese Klemmen können jederzeit zusätzliche Komponenten, mit Stichleitungen, an den Bus angeschlossen werden. Mit Hilfe des NI Systems kann über den CAN-Bus ein Datenaustausch mit diesen Komponenten realisiert werden.

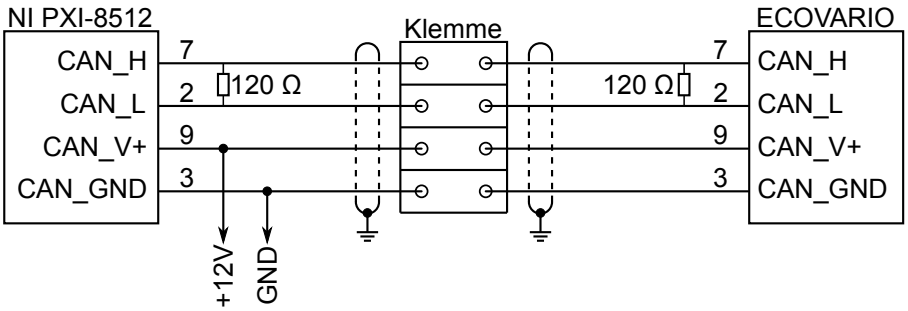



**Abbildung 4.1:** CAN-Bus Verdrahtung

## 4.1.2 Connector Box

Die Connector Box bietet vielseitige Funktionen für verschiedene Anschlüsse [2]. Diese Funktionen können in der Software MAX konfiguriert werden (vgl. Abschnitt 4.2.1).
Acht digitale Anschlüsse wurden im Schaltschrank auf Klemmen gelegt. Somit können jederzeit verschiedenste Bauteile angeschlossen und die Anschlüsse über die Software konfiguriert werden.

Über die Connector Box wird sowohl der Zustand des Überbrückungsschalters der Lichtschranken (ANF_04), als auch die beiden Encoder eingelesen. Der Zustand des Schalters wird über den digitalen Eingang „P0.0“ erfasst. Wird der Schalter geschlossen, schaltet ein Relais. Ein Schließerkontakt des Relais liegt zwischen dem „+5 V“ Anschluss und dem „P0.0“. Beim Schalten des Eingangs und anliegen von +5 V wird die Betätigung des Überbrückungsschalters softwaretechnisch erkannt.

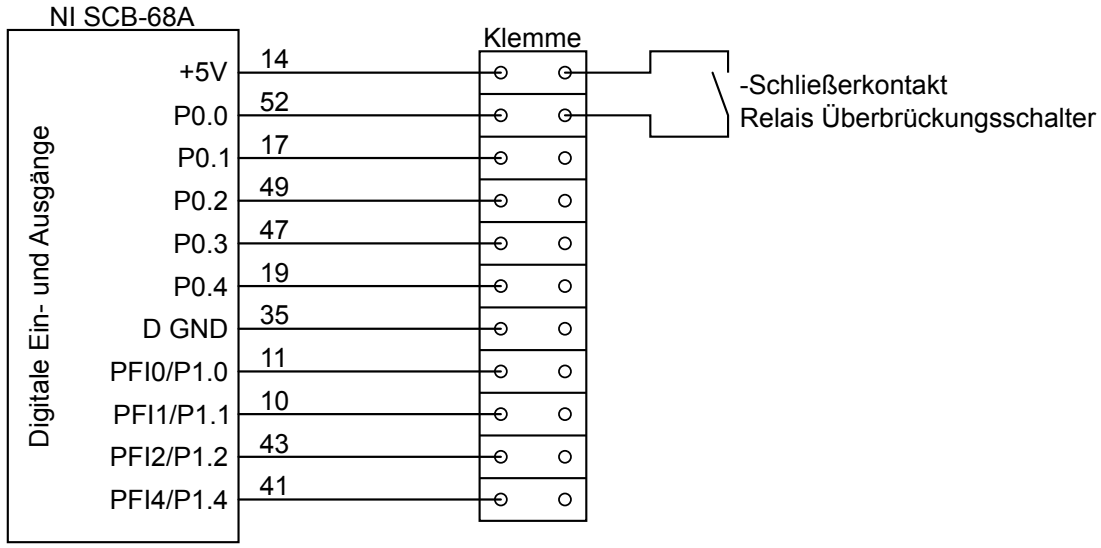



**Abbildung 4.2:** Verdrahtung der digitalen Anschlüsse im Schaltschrank

Sowohl der Referenz-Encoder (Magnetsensor), als auch das zurzeit angeschlossene DUT (Intacton GmbH - CVD-A1-300-RS-S4-M00) verwenden die Anschlussart RS422 (vgl. Abschnitt 3.2). Das NI System arbeitet allerdings mit einem Masse bezogenen Signal.

Um die beiden Encodersignale einlesen zu können, werden Signal Converter (LEG SU3-2) verwendet. Diese wandeln das RS422 Signal in ein 5V TTL Signal um. Die Versorgung der beiden Encoder und der Signalwandler sind über eine 24 V Spannungsquelle realisiert worden. Der Anschluss des DUT ist in Abbildung 4.3 dargestellt.

Einer fehlerhafte Datenübertragung wurde mit Hilfe von verdrillter und abgeschirmter Leitung entgegengewirkt. Wichtig ist hierbei, dass die Schirmung auf der gemeinsamen Potentialausgleichsschiene der Anlage anliegt.

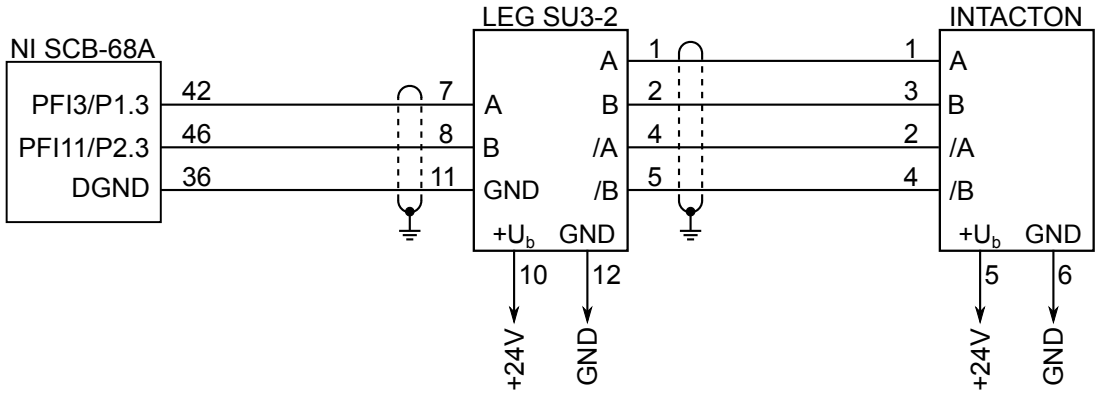



**Abbildung 4.3:** Verdrahtung des DUT (Intacton GmbH - CVD-A1-300-RS-S4-M00)

In dieser Arbeit verwendet der Servoverstärker als Referenz für die Fahrprogramme den Motorencoder. Da als alternative Referenz der Fahrprogramme der Magnetsensor genutzt werden könnte, wurde dieser zusätzlich an den Ecovario angeschlossen.

Laut Norm RS422 wird am Ende einer Leitung ein 120 Ω Abschlusswiederstand zwischen Signal und invertiertem Gegenstück geschaltet [16]. Im Signal Converter sind diese bereits integriert [17], in der Steckverbindung zum Ecovario wurden diese zusätzlich eingelötet.
Der Anschluss des Magnetsensors ist in Abbildung 4.4 dargestellt. Hierbei ist wiederum auf die passende Leitungswahl (verdrillt, abgeschirmt) und auf den korrekten Anschluss der Schirmung geachtet worden.

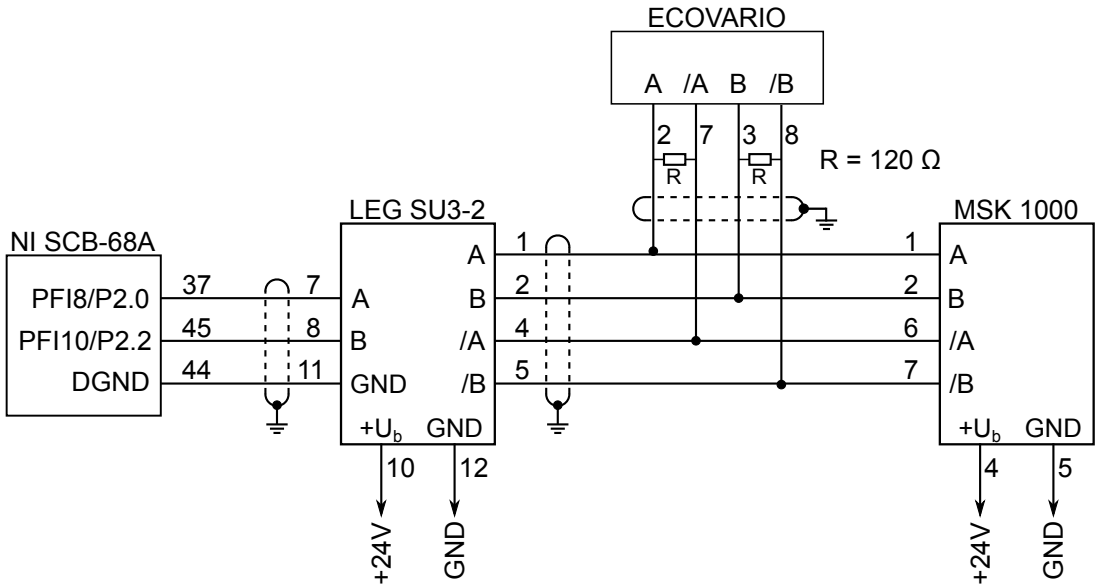



**Abbildung 4.4:** Verdrahtung des Magnetsensors (Siko MSK 1000)

## 4.2 Software

### 4.2.1 National Instruments System

In diesem Abschnitt werden verschiedene Softwares, die von NI zur Verfügung gestellt werden, beschrieben.

**Measurement & Automation Explorer (MAX):** In den ersten Arbeitsschritten wurde diese Software zur Konfiguration der einzelnen Hardwarekomponenten verwendet.

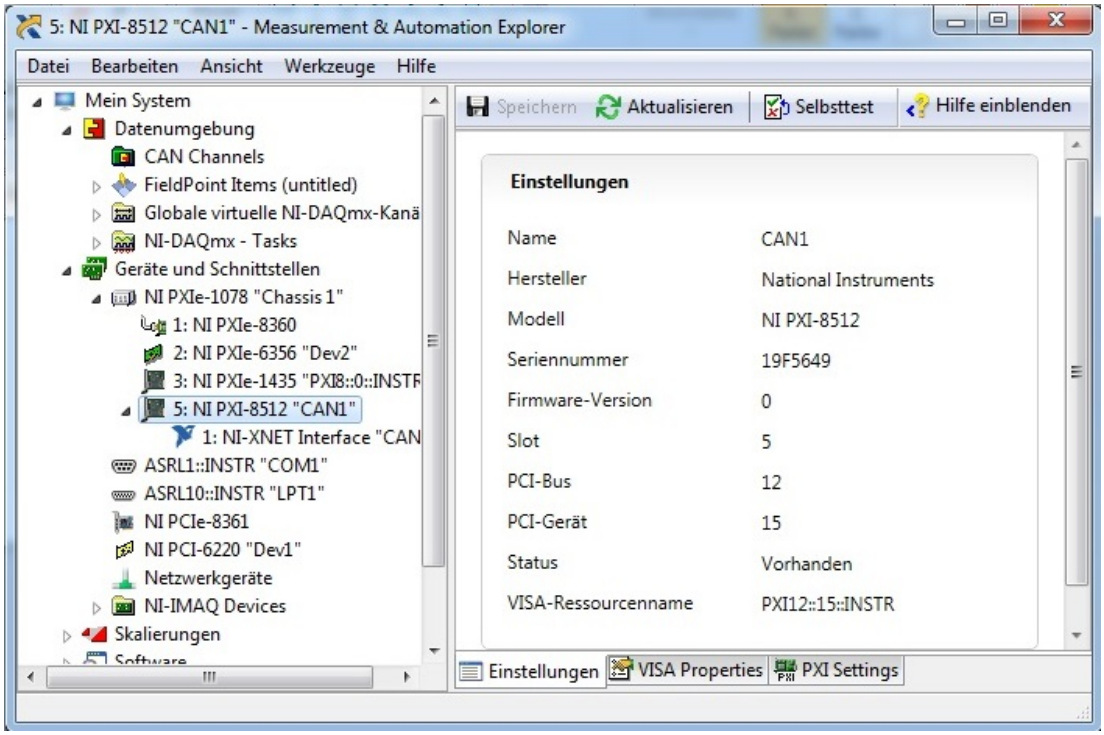



**Abbildung 4.5:** Measurement & Automation Explorer

Auf der linken Seite befindet sich eine Baumstruktur in der sich die beiden Themen „Datenumgebung“ und „Geräte und Schnittstellen“ befinden.

Unter dem Thema „Geräte und Schnittstellen“ verbirgt sich das angeschlossene Chassis und darunter die einzelnen Karten. Zu beachten ist hierbei, dass das Chassis vor dem Hochfahren des Computers gestartet werden muss.
Jede Karte hat verschiedene Funktionen und Einstellmöglichkeiten. Eine Hilfe befindet sich oben rechts unter „Hilfe einblenden“. Diese Software ermöglicht das umbenennen von Karten und Ports. Zusätzlich können über die Funktion „Selbsttest“ einzelne Karten auf ihre korrekte Funktion getestet werden.

Um die Arbeit mit den verschiedenen Anschlüssen der Connector Box zu vereinfachen, können verschiedene Kanäle (Tasks) unter dem Thema „Datenumgebung“ erstellt werden.
In diesen Tasks werden die einzelnen Anschlüsse für die entsprechende Funktion konfiguriert. Jeder Anschluss kann über diese Software und über den entsprechenden erstellten Tasks, ohne Programmcode, auf die korrekte Funktion getestet werden.
Somit wurde die Funktion des Überbrückungsschalters der Lichtschranken und der beiden Encoder vor der eigentlichen Implementierung in LabVIEW mit Hilfe dieser Software getestet. Diese erstellten Tasks können in LabVIEW aufgerufen werden. Zusätzlich besteht die Möglichkeit veränderbare Tasks erst zur Laufzeit, mit Hilfe der DAQmx-API, zu erstellen. Im Folgenden wird die Erstellung des Tasks für den Überbrückungsschalter, der an Port P0.0 angeschlossen wurde, Schritt für Schritt beschrieben.

*Rechtsklick auf Datenumgebung → NI-DAQmx-Task → Weiter*

Im darauffolgendem Fenster wird angegeben, welche Art von Signal erfasst oder erzeugt werden soll. Für diese Anwendung wird lediglich ein digitaler Eingang benötigt (Port 0/ Line 0). Der Anschluss dieses Ports ist in Abschnitt 4.1.2 beschrieben.

*Signal erfassen → Digitale Erfassung → Erfassung über Leitung → PXIe-6356 → port0/line0*

Die beiden Tasks für die beiden Encoder sind „Zählergestützte → Positions (Weg) Erfassungen“ und wurden auf die Zähleranschlüsse des PXIe-6356 Port 0 (Referenz) und Port 1 (DUT) gelegt. Es gibt verschiedene Möglichkeiten die Impulse zu zählen. In dieser Arbeit wurde für die Kodierung „X4“ verwendet, diese bietet die höchste Empfindlichkeit [Onl4].
Um Positionsangaben zu erhalten, kann die Einstellung „Abstand pro Impuls“ mit Hilfe der Auflösung angepasst werden. Zusätzlich kann auch noch eine Anfangsposition vorgegeben werden. In den in dieser Arbeit erstellten Tasks werden nur die Inkremente gezählt. Die Kodierung „X4“ zählt an jeder Flanke des Signal A und B, also werden vier Zähl-Schritte pro Inkrement getätigt. Durch die Vorgabe „4 = Abstand pro Impuls“ erhält man die Anzahl der Inkremente.
Eine Ausführliche Beschreibung der Einstellungen bietet *„Anschließen von Quadratur-Encodern an ein DAQ-Gerät“* [Onl4]

**NI-XNET Datenbank Editor:** NI-XNET Treiber und API werden zur Entwicklung von Frame- und Signalanwendungen in LabVIEW verwendet.
Um mit dem Servoverstärker über CANopen zu kommunizieren müssen einzelne Frames mit Objekten übertragen werden. In der XNET-API sind bereits Unterprogramme (Sub-VIs) vorgesehen um Frames zu senden und zu empfangen.
Diese benötigen Informationen wie die Schnittstelle (Interface), das Protokoll und die zu verwendende Baudrate [Onl5]. Die Informationen werden in Form von XNET-Datenbanken und Sessions vorgegeben. XNET Datenbanken sind sehr vielfältig und können das Einlesen von physikalischen Signalen stark vereinfachen [18].

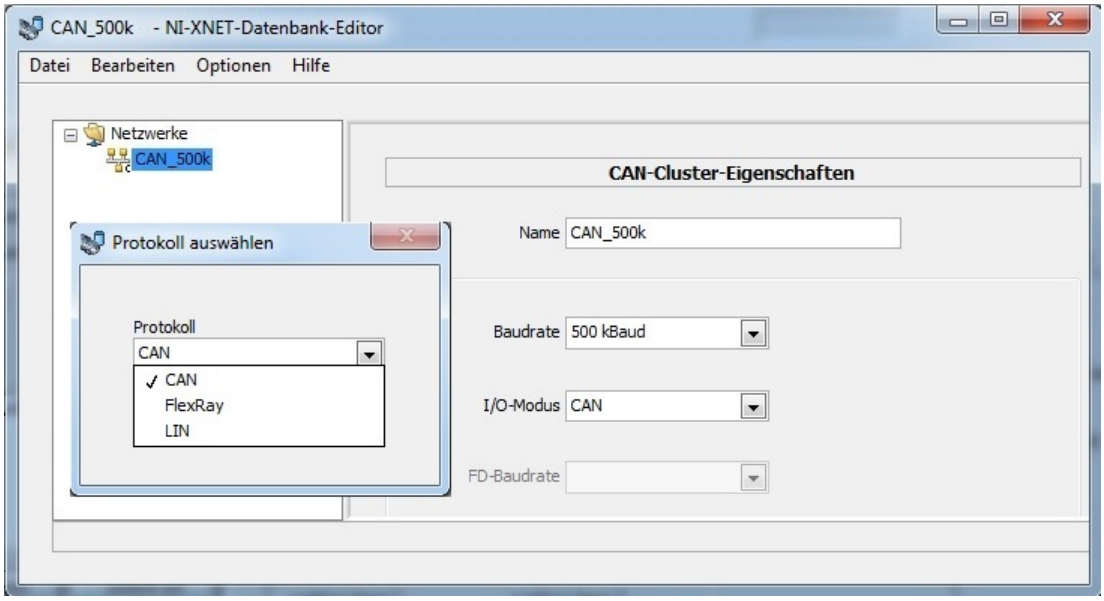



**Abbildung 4.6:** NI-XNET Datenbank Editor

Nach dem Öffnen der Software wird eine neue Datenbank angelegt und gespeichert. Um die oben genannten Einstellungen zu tätigen, wird ein Cluster erstellt.

*Rechtsklick auf Netzwerke → Cluster erstellen*

Im neu geöffneten Fenster wird dem Cluster ein Name gegeben, das Protokoll (CAN, FlexRay, LIN [9]) und der I/O-Modus auf CAN eingestellt und die Baudrate vorgegeben. Die Baudrate ist abhängig von der Bus-Datenleitungs- und Stichleitungslänge und ist bei diesem Teststand auf 500 Kilo Bits pro Sekunde eingestellt [9, Tabelle 6.1-2].

**LabVIEW:** LabVIEW ist eine grafische Systemdesignsoftware der Firma National Instruments.
In LabVIEW wird mit Projekten gearbeitet, in denen sich verschiedene VIs befinden. Jedes VI enthält Programmcode und/oder Sub-VIs (Unterprogramme). Um mit der bereits erwähnten NI-XNET API arbeiten zu können muss das Projekt zusätzlich Sessions enthalten.
Dazu wird aus dem LabVIEW Startfenster heraus ein neues Projekt geöffnet, gespeichert und eine Session erstellt.

*Rechtsklick auf mein Computer → Neu → NI-XNET Session*

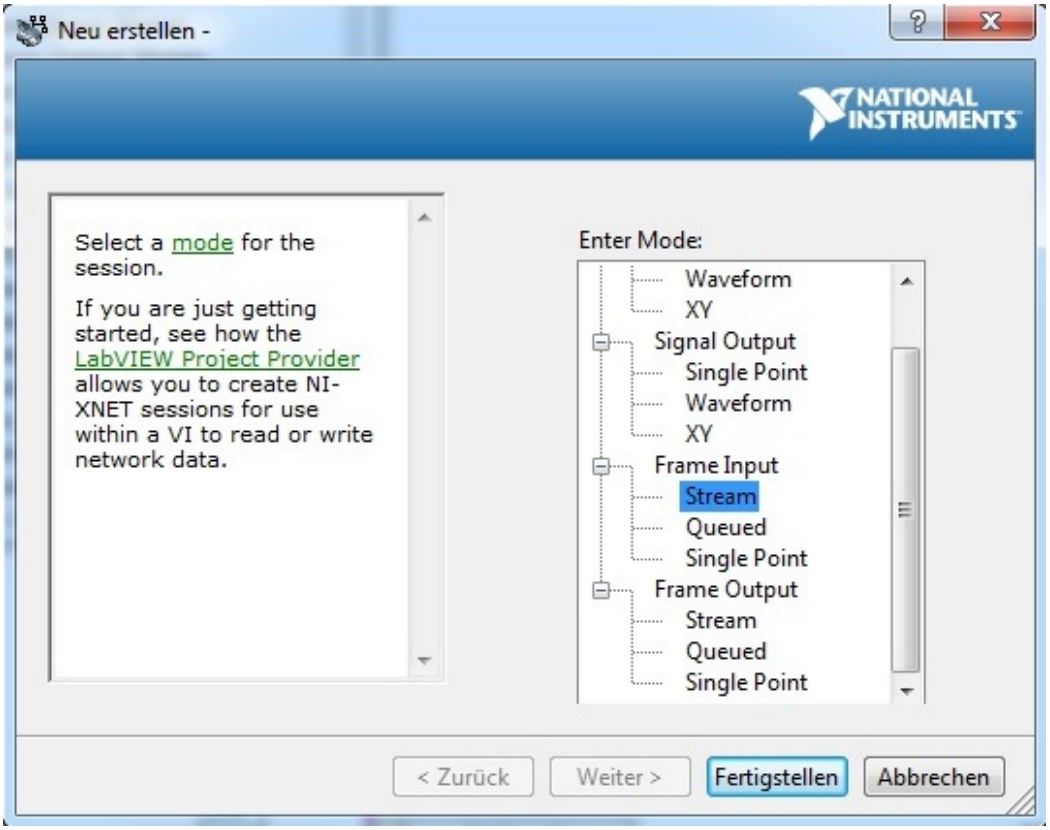



**Abbildung 4.7:** Erstellen einer XNET-Session

In dem erscheinenden Fenster wird der Modus ausgewählt, d.h. die Richtung und die Darstellung der Daten. Da die CANopen Kommunikation aus reinen Frames besteht, wird sowohl eine Frame Output- als auch eine Frame Input Stream Session benötigt.
In dem darauffolgenden Session Setup wird der Session ein Name gegeben, das in der Software MAX eingestellte Interface für die verwendete Hardware angegeben, sowie die Datenbank und das erstellte Cluster eingestellt. Die erstellten Sessions können aus der Projekt-Baumstruktur direkt per drag & drop in das Blockdiagramm des VIs gezogen und der Programmcode angelegt werden (Kapitel 5).

### 4.2.2 Jenaer Antriebstechnik

Um den Servoverstärker (Ecovario 414) in Betrieb zu nehmen wird die Software „ECO Studio“ benötigt. Eine Verbindung zwischen Servoverstärker und ECO Studio kann über RS232, RS485, CAN über USB-Dongle, CAN über PEAK-Dongle, USB direkt, Ethernet oder EtherCAT hergestellt werden. Die Durchführung der Inbetriebnahme und das Einstellen der verschiedenen Parameter ist im Bedienhandbuch *„ECO Studio“* [19] ausführlich beschrieben.
Nachdem die Inbetriebnahme abgeschlossen wurde, wird die Software nicht mehr benötigt, da die Kommunikation mit dem Ecovario über LabVIEW implementiert wird.

# 5 Implementierung in LabVIEW

In diesem Kapitel wird veranschaulicht, wie die einzelnen Anforderungen in LabVIEW gelöst wurden. Grundgedanke des Programmes ist es eine Haupt-Whileschleife und für die einzelnen Fahrprogramme und die Datenerfassung Zustandsautomaten (engl. State Machines) zu implementieren. Alle in Abbildung 5.1 aufgeführten Funktionen, sind in diesem Kapitel beschrieben.
Eine State Machines ist eine grundlegende LabVIEW Architektur in der bestimmte Zustände programmiert werden und durch Bedingungen zwischen diesen beliebig gewechselt werden kann [Onl6].

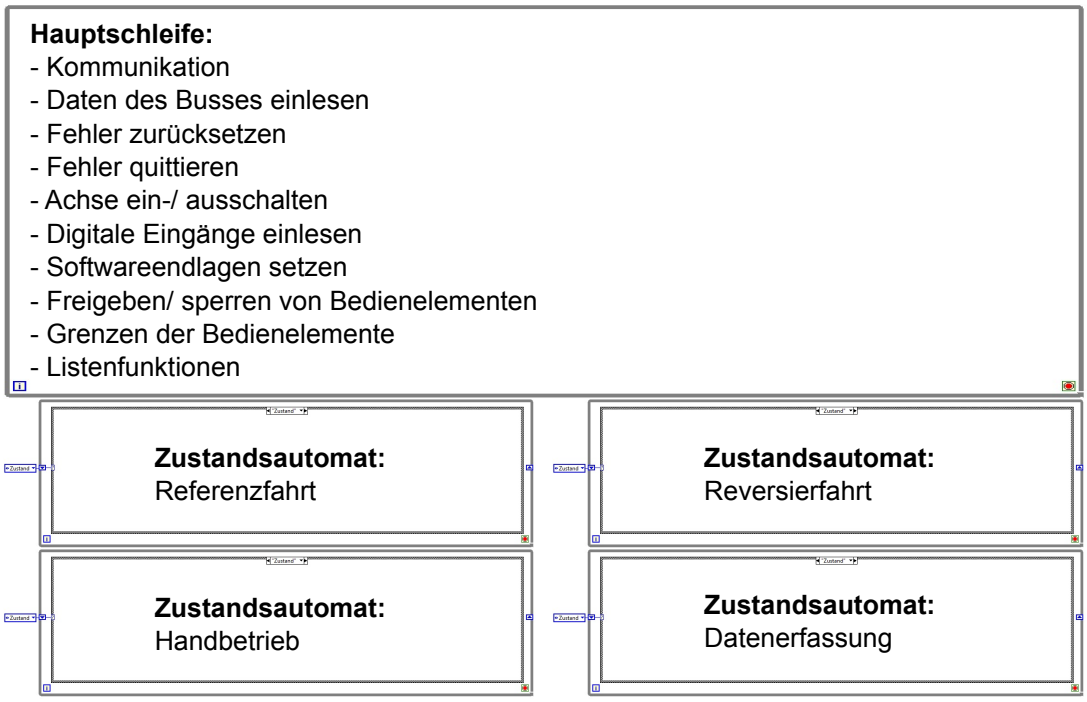



**Abbildung 5.1:** Programmarchitektur

Wie bereits in Abschnitt 4.2.1 erwähnt wird für den Informationsaustausch mit dem CAN-Bus die XNET-API verwendet. Diese beinhaltet ein „XNET write (Frame CAN).vi“ und ein „XNET read (Frame CAN).vi“ zum beschreiben bzw. lesen des Busses. Diesen VIs wird pro Frame der Identifier und der Payload Data übergeben.

Der Payload Data wird in die acht Datenbytes des CAN-Frames geschrieben und nach den CANopen Spezifikationen (vgl. Abschnitt 3.1.3) gebildet. Im Lastenheft wurde in ANF_07 festgelegt, dass die GUI in der englischen Sprache ausgeführt sein soll. Dies wurde berücksichtigt und umgesetzt. Eine Abbildung der GUI befindet sich im Anhang.

## 5.1 Kommunikation

Die Anwendung soll sowohl den Lineareteststand, als auch einen rotativen Teststand steuern (ANF_06). Um dies übersichtlich in der GUI umzusetzen, ist diese in drei Registerkarten unterteilt. Die Erste ist für die Erkennung des angeschlossenen Teststands zuständig und wird bei Programmstart aufgerufen (Connection).

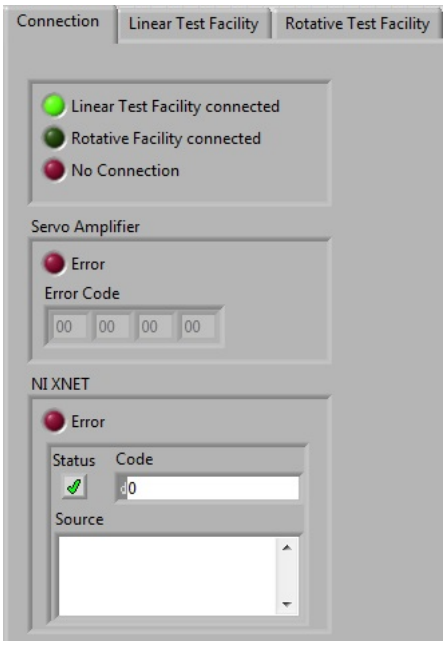



**Abbildung 5.2:** Kommunikationsbereich der GUI

Nachdem die Verbindung hergestellt wurde, wird entweder die Registerkarte „Linear Test Facility“ oder „Rotative Test Facility“ freigeschaltet und initialisiert. Ist zu keinem der beiden Teststände eine Verbindung vorhanden, wird dies in der GUI angezeigt (No Connection). Zusätzlich müssen Kommunikationsfehler abgefragt und angezeigt werden.
Es werden zwei verschiedene Kommunikationsfehler unterschieden. Zum einen kann ein Fehler im Bezug mit dem CANopen Protokoll auftreten (Servo Amplifier), zum anderen kann es zu einem Fehler bei der Kommunikation mit dem NI System kommen (NI XNET).

Um diese beiden Fehlertypen abzufangen, wird sowohl der übertragene Fehlercode der SDO-Kommunikation dargestellt, als auch der Fehlerausgang der verwendeten XNET-VIs zur Anzeige gebracht.
Kommt es durch Verlust der Versorgungsspannung des Servoverstärkers oder des Busses zu einem Verbindungsabbruch wird dies erkannt und in der GUI angezeigt. Bei Wiedereinschalten könnte die NI PXI-8512 Karte in den Fehlerzustand „controller transitioned to bus off“ gehen [20]. Dieser Fehler wird durch einen Neustart behoben. Mit dem Typ „CAN Comm State“ des „XNET read.vi“ wird der aktuelle Status der NI-Karte ausgelesen. Ist ein Fehler vorhanden wird mit Hilfe des „XNET Start.vi“ der Neustart vollzogen.

Die Initialisierung des Lineareteststands sieht das PDO-Mapping vor (vgl. 3.1.3.3 PDO). Die benötigten Objekte werden per SDO in das Mappingobjekt geschrieben.
In dem Tx-PDO1 ist das aktuelle Statuswort und der Zustand der digitalen Eingänge des Servoverstäkers gemappt. Die beiden Objekte werden alle 35 ms vom Servoverstärker versendet.
Die digitalen Eingänge dienen zur Kontrolle der mechanischen Lageendschalter. Ist einer der Endschalter betätigt kann über das PDO1 dieser Zustand in der GUI angezeigt werden (ANF_02). Der Servoverstärker unterbricht selbständig das Fahrprogramm bei Betätigung einer der beiden Endschalter.
Um die Anforderung „Geschwindigkeits- und Positionsanzeige“ umzusetzen, werden die beiden Objekte Ist-Position und Ist-Geschwindigkeit in das Tx-PDO2 gemappt (ANF_05). Dieses PDO wird bei Wertänderung in einem der beiden Objekte mit einer Inhibit Time von 100 ms gesendet.
Alle anderen Parameter sind im Servoverstärker abgespeichert und müssen nicht bei der Initialisierung definiert werden. Außer die Polarität (Öffner/Schließer) der digitalen Eingänge, da sich diese beim Öffnen des ECO Studios ändern könnten. Eine Sicherungskopie (Backup) dient zur Wiederherstellung bei ungewollten, nicht mehr nachvollziehbaren Änderungen der Objekte im Servoverstärker.

Der Programmcode für die Verbindungsherstellung und für die Initialisierung wurden in Unterprogramme aufgeteilt, um die Übersicht zu steigern und Änderungen leichter im jeweiligen Programm umsetzten zu können.
Da die Kommunikation und Initialisierung nur einmal am Anfang ausgeführt werden müssen, wurde in der Hauptschleife eine Case-Struktur eingefügt. Diese Struktur besitzt die äußere Registerkarte als Bedingung. Der beschriebene Programmcode befindet sich im Case des „Connection“ Reiters und wird nur abgearbeitet wenn dieser geöffnet ist.

## 5.2 GUI


(a) Normalzustand (b) Fehlerzustand

**Abbildung 5.3:** Statusanzeige der GUI

**Daten des Busses einlesen:**
Um die vom Servoverstärker gesendeten PDOs jederzeit auswerten zu können, werden in der Hauptschleife die Daten des Busses gelesen. Die eingelesenen Frames werden anhand des Identifier sortiert und somit zwischen verschiedenen PDOs, SDOs oder EMCYs unterschieden (vgl. Abschnitt 3.1.3.1).

Der Payload des PDO1 wird aufgesplittet und die entsprechenden Bits der digitalen Eingänge direkt mit der Anzeige für die Endlagenschalter verbunden (ANF_02).
Das Statuswort wird in einzelnen Sub-VIs untersucht. Sub-VIs haben den Vorteil, dass diese unabhängig vom Hauptprogramm getestet und verändert werden können. Dies ermöglicht eine frühzeitige Fehlersuche und -erkennung. In diesen Sub-VIs wird zum einen festgestellt ob es sich bei dem eingelesenen Statuswort um einen Fehlerzustand handelt, woraufhin das Anzeigefeld des Statuswortes entsprechend der ANF_01 rot oder grün gefärbt wird, zum anderen werden die im Normalbetrieb vorkommenden Stati in Klartext übersetzt. Dazu wird die Bedeutung der einzelnen Bits mit Hilfe des *„Objektverzeichnis“* [14] im Anzeigefeld „Status“ angezeigt (siehe Abbildung 5.3 (a)).
Zusätzlich wurde ein Timeout integriert. Sollte der Servoverstärker die PDO1 nicht mehr regelmäßig senden wird ein Verbindungsabbruch erkannt, alle Bedienelemente gesperrt und dies oben links über dem Statuswort in der GUI angezeigt.

Die Bedienelemente werden aus Sicherheitsgründen gesperrt, da nicht mehr sichergestellt ist in welchem Zustand sich der Teststand befindet.

Emergency Messages werden bei Fehlerzuständen vom Servoverstärker auf den Bus gelegt und besitzen einen eigenen Identifier. Der Fehlercode in diesen Frames wird ebenfalls im Anzeigefeld Status angezeigt (siehe Abbildung 5.3 (b)).

Aus dem PDO2 wird die aktuelle Geschwindigkeit und Position entnommen und zur Anzeige gebracht (ANF_05). Die Daten müssen in Dezimalzahlen umgewandelt und in die entsprechende Anzeigeeinheit umgerechnet werden. Zum Beispiel befindet sich die aktuelle Geschwindigkeit in inc/64 s im Objekt [14]. Um diese in mm/s anzuzeigen muss durch 64 und die Motorencoderauflösung (640 inc/mm) geteilt werden.

$$v \cdot \frac{\text{inc}}{64\ \text{s}} \Big/ \frac{640\ \text{inc}}{\text{mm}} = v \cdot \frac{\text{inc} \cdot \text{mm}}{64\ \text{s} \cdot 640\ \text{inc}} = \frac{v}{64 \cdot 640} \cdot \frac{\text{mm}}{\text{s}} \tag{5.1}$$

Um die ANF_05 vollständig zu lösen ist für die Information über das erreichen der Soll-Position ein extra Anzeigeelement vorgesehen worden. Die Information liefert das Statuswort.

**Fehler zurücksetzen und quittieren:**
In Abbildung 5.3 ist der „Reset Error“ Button zu erkennen. Sollte ein Fehler auftreten, kann dieser über den Button zurückgesetzt werden. Dazu wird per SDO im Steuerwort das Bit 7 (Rücksetzten Gerätefehler) gesetzt und daraufhin der Ausgangszustand (Steuerwort = 0x0006) hergestellt. Ist der Fehler behoben sendet der Servoverstärker eine leere Emergency Message und der Fehler wird nicht mehr im Status angezeigt.
Da ein Statusfehler, wie zum Beispiel „Internal Limit Active“ nur kurzzeitig anliegt, könnte die Anzeige dem Benutzer entgehen. Um dies zu vermeiden werden Fehlerzustände dauerhaft angezeigt und erst durch betätigen des „Acknowledge“ Button quittiert.

**Achse ein-/ausschalten und digitale Eingänge einlesen:**
Laut ANF_04 soll es möglich sein die Lichtschranken zu überbrücken. Da es wünschenswert ist die Achse unter diesen Umständen im Betrieb per Hand zu verschieben, wurden Tasten vorgesehen um den Motor manuell aus und ein zu schalten. Dazu werden je nach Tastenbetätigung dem Steuerwort der Wert zum Aus- bzw. Einschalten übergeben (siehe *„Objektverzeichnis“* [14]).
Des Weiteren gibt diese Anforderung vor, bei versuchtem Einschalten während die Anlaufsperre aktiv ist (Lichtschranken nicht zurückgesetzt, Not-Aus-Schalter betätigt), einen Fehler auszugeben, der über die Software zurückgesetzt werden muss. Dies wird über die Kommunikation mit EMCY und dem „Reset Error“ realisiert.

Der Zustand, dass die Lichtschranke überbrückt ist wird über einen digitalen Eingang der NI SCB-68A Connector Box zur Anzeige gebracht (Bypass Switch). Um diesen einzulesen wird das „DAQmx Read.vi“ aus der DAQmx-API benötigt. Dieses VI verwendet den in der Software MAX konfigurierten Task (vgl. Abschnitt 4.2.1).

**Softwareendlagen setzen:**
Unten links in der GUI ist es dem Anwender möglich Softwareendlagen zu setzen (ANF_15).

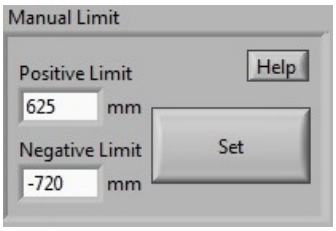



**Abbildung 5.4:** Softwareendlagenbereich der GUI

Diese verhindern, dass der Schlitten in die mechanische Endlage fährt. Die eingestellte Schnellstoppbremsrampe ($a_{\text{Schnellstopp}}$) beträgt 23 000 mm/s$^2$. Bei maximaler Geschwindigkeit ($v_{\text{max}}$) von 1650 mm/s ergibt sich ein Bremsweg von 59,18 mm.

Einschub: Die gleichmäßig beschleunigte Bewegung [21]

$$v = a \cdot t \quad \Rightarrow \quad t = \frac{v}{a} \tag{5.2}$$

$$s = \frac{1}{2} \cdot a \cdot t^2 \tag{5.3}$$

Die Zeit ($t$) die benötigt wird um mit einer bestimmten Beschleunigung ($a$) eine Geschwindigkeit ($v$) zu erreichen (Formel 5.2), wird in die Formel 5.3 eingesetzt. Die entstehende Formel berechnet die Strecke ($s$) die zurückgelegt wird um mit einer Beschleunigung eine Geschwindigkeit zu erreichen oder von einer Geschwindigkeit zum Stillstand zu gelangen.

$$s = \frac{1}{2} \cdot a \cdot \left(\frac{v}{a}\right)^2 = \frac{1}{2} \cdot a \cdot \frac{v^2}{a^2} = \frac{1}{2} \cdot \frac{v^2}{a} \tag{5.4}$$

$$s = \frac{1}{2} \cdot \frac{v_{\text{max}}^2}{a_{\text{Schnellstopp}}} = 59,18 \text{ mm} \tag{5.5}$$

Auf dem Lineareteststand werden verschiedene Materialien getestet und auf dem Schlitten eingespannt. Beim Bremsen wird der Strom aus dem Motor in den Servoverstärker gespeist (generatorischer Betrieb) und in einem Ballastwiderstand geleitet. Die Bremsenergie wird somit im Ballastwiderstand in Wärme umgewandelt. Ist die Last des Schlittens zu groß, könnte die maximale Stromstärke, die der Servoverstärker aufnehmen kann, überschritten werden. Dies hätte zur Folge, dass die Schnellstoppbremsrampe nicht sicher umgesetzt werden kann und der Bremsweg sich verlängert. Um sicherzustellen, dass unter diesen Umständen der Schlitten nicht in die mechanische Endlage fährt, wird mit jedem Material ein Test durchgeführt. Dieser Test sieht eine Fahrt mit maximaler Geschwindigkeit vor, die in der Mitte des Teststandes mit einem Not-Aus-Schalter (Schnellstoppbremsrampe) abgebrochen wird. Der resultierende Bremsweg muss von den mechanischen Endlagen abgezogen werden und diese Werte als Softwareendlage gesetzt werden. Die Endlagen beziehen sich nach erfolgreicher Referenzierung auf den Nullpunkt des Teststands (vgl. Abschnitt 5.3)

positive Softwareendlage = 645 mm - Bremsweg
negative Softwareendlage = -740 mm + Bremsweg

Der Verlauf des Tests ist in der GUI unter dem „Help“ Button beschrieben.

## 5.3 Fahrprogramme

Das Lastenheft sieht drei Fahrprogramme vor. Die Referenzfahrt fährt den mittleren Lageschalter an und setzt dort den Referenzpunkt (0 mm). Der Handbetrieb sieht eine relative oder absolute Positionierung vor, wobei der Fahrschlitten an eine einzustellende Position gefahren wird. Die eigentliche Messung findet im Reversierbetrieb statt. In diesem Fahrprogramm reversiert der Schlitten zwischen zwei Positionen. Währenddessen werden die Daten der Referenz und des DUT aufgenommen.
Die einzelnen Fahrprogramme werden vom Motor auf den Schlitten umgesetzt, diese Positionierungen werden nicht am Schlitten, sondern vom internen Motorencoder bestimmt. Dies bedeutet, dass die Positionseingaben und die erfassten Messdaten aufgrund der Mechanik (Spiel im Getriebe, der Spindel, etc.) von einander abweichen können. Allerdings nicht die Messdaten des DUT und des Magnetsensors, da diese beiden den Schlitten, bzw. das Material erfassen.

Die drei Fahrprogramme wurden mit Hilfe von State Machines in LabVIEW umgesetzt. Diese Methode ermöglicht es leicht Bedingungen oder Zustände einzufügen oder Fehler zu finden und zu beheben. Durch die Aufteilung in drei verschiedene Softwarekomponenten ist eine separate und frühzeitige Verifikation und Analyse von Fehlern möglich.

LabVIEW-Programme werden ohne spezielle Programmierung parallel ausgeführt [18]. Nach der Architektur in Abbildung 5.1 ist zu erkennen, dass die State Machines und die Hauptschleife parallel ausgeführt werden. Dies hat den Vorteil, dass während eines Fahrprogrammes das aktuelle Statuswort, EMCYs, usw. eingelesen und verarbeitet werden. Aus dem Statuswort können alle wichtigen Zustände des Servoverstäkers gelesen werden und dienen als Bedingungen für die Fahrprogramme.

**Freigeben/ sperren von Bedienelementen:**
Durch Freischalten und Sperren von Bedienelementen werden nicht erlaubte Aktionen verhindert. Das Aktivieren bzw. Deaktivieren von Bedienelementen wurde in LabVIEW mit Hilfe von Eigenschaftsknoten umgesetzt. Vor der Referenzfahrt können keine Softwareendlagen gesetzt werden, da diese sich auf den Nullpunkt beziehen. Dieser Nullpunkt ist vor der Referenzfahrt die Position beim Einschalten und nach absolvierter Referenzfahrt der Referenzpunkt. Zusätzlich ist das Bedienelement zur Auswahl der absoluten Positionierung und die Registerkarte für den Reversierbetrieb vor der Referenzfindung gesperrt. Ein sicherer Betrieb dieser Fahrprogramme ist erst nach erfolgreicher Referenzfindung gewährleistet. Nur eine relative Positionierung ist vor der Referenzfahrt möglich (ANF_08). Diese Funktion wurde umgesetzt, da der Schlitten sich bei Arbeiten und Umbauten an der Anlage in einer ungünstigen Position befinden könnte. In solchen Fällen muss nicht erst eine Referenzfahrt durchgeführt werden, sondern es kann sofort eine relative Positionierung in Bezug auf die Ist-Position vorgenommen werden.
Während eines Fahrprogrammes ist es nicht möglich ein anderes Programm zu starten. Dafür werden beim Start die „Start“ Button der anderen Programme gesperrt und erst nach Beendigung wieder freigegeben. Ausnahme bildet hier nur die relative Positionierung und die Referenzfahrt. Während einer relativen Positionierung ist es jederzeit möglich in die Referenzfahrt zu wechseln, umgekehrt nicht. Wird durch falsche Bedienung die relative Positionierung anstatt der Referenzfahrt gestartet, muss nicht erst ein Abbruch dieser Erfolgen.

**Grenzen der Bedienelemente:**
Um der ANF_11 „Grenzwerte Geschwindigkeit, Beschleunigung und Position“ gerecht zu werden, müssen sowohl in den Einstellungen der jeweiligen Bedienelemente die Grenzen gesetzt werden, als auch in der Hauptschleife mit Hilfe von Eigenschaftsknoten die Hintergrundfarbe je nach Wert verändert werden. Dies verhindert eine ungültige Eingabe außerhalb definierter Grenzen. Die Grenzen wurden mit dem Auftraggeber getestet und festgelegt.

| | |
|---|---|
| max Position negativ | -740 mm |
| max Position positiv | +645 mm |
| max Geschwindigkeit | 1650 mm/s |
| max Beschleunigung | 4000 mm/s$^2$ |
| max Bremsverzögerung | 15000 mm/s$^2$ |

**Tabelle 5.1:** Grenzwerte des Systems

### 5.3.1 Referenzfahrt

Die Referenzfahrt ist in der ANF_03 beschrieben. Die Bedienelement befinden sich unter „Homing“.

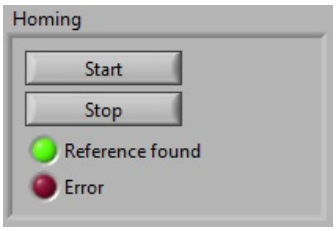



**Abbildung 5.5:** Homing Bereich der GUI

Die Referenzfahrt dient zum Setzten des Nullpunkts am mittleren Lageschalter und ist aus jeder Position nach dem Einschalten möglich. Fährt der Schlitten auf einen Lageendschalter kehrt dieser die Fahrtrichtung um.
Nach der Referenzierung werden die vorher gesperrten Bedienelemente freigeschaltet und die der Referenzfahrt gesperrt. Die Information „Reference found“ liefert das Statuswort und wird in der GUI angezeigt. Bei positiver Flanke dieses Anzeigeelements werden Softwareendlagen gesetzt, die vorher mit dem Auftraggeber vereinbart wurden (positiv: 625 mm, negativ: -720 mm).
In Abbildung 5.6 ist der Ablauf der einzelnen Zustände der Referenzfahrt dargestellt.

**Zustandsautomaten:**
Alle drei Fahrprogramme sind in dieser Arbeit in Zustandsdiagrammen abgebildet. Die Zustände „Fehler“ und „Stopp“ wurden pro Zustandsautomat (Fahrprogramm) nur einmal implementiert, diese sind für die Übersichtlichkeit in den Diagramme (Handbetrieb, Reversierfahrt) mehrfach vorhanden.
Um mit dem Servoverstärker zu kommunizieren, wird in den einzelnen Zuständen die XNET-API verwendet. Diese Sub-Vis haben einen Fehlerausgang, der für eine zusätzliche Abbruchbedienung verwendet wird.

Sollte ein XNET-Fehler auftreten, wird sofort im jeweiligen Zustandsautomat in den Fehler-Zustand gesprungen. Dies ist in den Abbildungen nicht eingetragen, da es die Übersichtlichkeit stark einschränken würde.
Die Bedingungen die für einen Übergang von einem Zustand in einen anderen erfüllt sein müssen, stehen an den entsprechenden Pfeilen. Eine Legende der Abkürzungen ist in Tabelle 5.2 aufgeführt.

| **Kürzel** | **Bedeutung** | **Kürzel** | **Bedeutung** |
|---|---|---|---|
| $\overline{XX}$ | Nicht XX | Fehler | EMCY oder Status Error |
| $\vee$ | ODER-Verknüpfung | Achse | Achse eingeschaltet |
| $\wedge$ | UND-Verknüpfung | Quitt | Sollwert quittiert |
| Start | Start-Button betätigt | Pos | Position erreicht |
| Stopp | Stop-Button betätigt | Hin | Hinfahrt |
| Kom | Kommutierung gefunden | Rück | Rückfahrt |
| Timer | Timer abgelaufen | Wdh | nächste Wiederholung |
| EMCY | Emergency Message empfangen | Liste | Liste abgearbeitet |
| Ref | Referenz gefunden | | |

**Tabelle 5.2:** Legende der Abbildungen der Zustandsautomaten

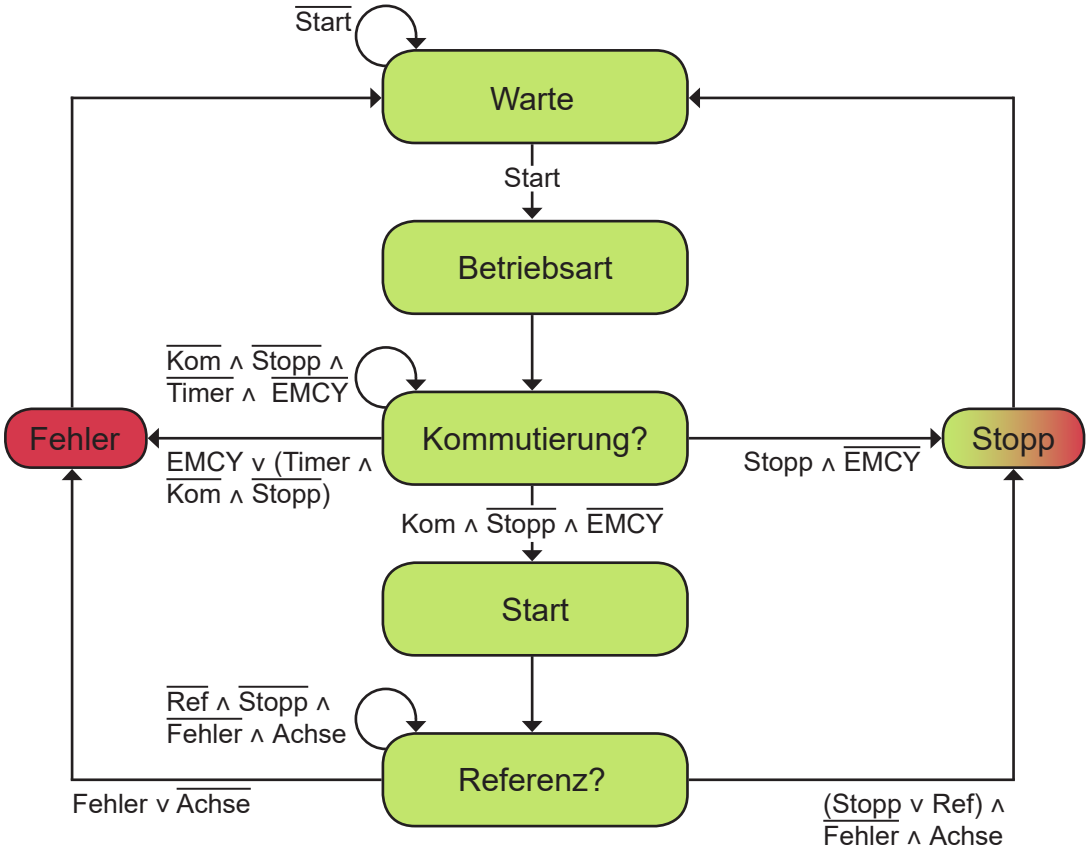



**Abbildung 5.6:** Zustandsautomat: Referenzfahrt

**Warte:** Wie der Name schon sagt wird in diesem Zustand gewartet, bis der „Start“ Button in der GUI betätigt wird. Nach Betätigung begibt sich der Zustandsautomat in den nächsten Schritt.

**Betriebsart:** In diesem Zustand wird dem Servoverstärker über ein SDO die Betriebsart „Referenzfahrt“ mit dem Wert 0x0006 übergeben. Um sicherzustellen, dass der Motor eingeschaltet ist, wird dem Steuerwort der Wert 0x000F zugewiesen. Zusätzlich wird der Timer gestartet der im nächsten Zustand benötigt wird.

**Kommutierung?:** Das Statuswort wird per PDO ständig in der Hauptschleife des Programmes eingelesen. Bevor die Referenzfahrt starten kann, muss der Servoverstärker die Kommutierung erfolgreich gefunden haben. Wurde die Kommutierung gefunden wechselt das Bit 14 im Statuswort zu „high“ und der nächste Zustand beginnt. Der Automat verbleibt in diesem Zustand bis die Kommutierung gefunden wurde.
Kann die Kommutierung durch einen Fehler nicht gefunden werden, wird nach Ablauf des Timers der Fehlerzustand ausgeführt. Sobald der „Stop“ Button betätigt wird, bricht das Programm ab und der Stopp-Zustand wird ausgeführt. Höchste Priorität haben die Fehlerzustände. Sendet der Servoverstärker eine Emergency Message wird sofort in den Fehler-Zustand gewechselt.

Einschub: Kommutierung
Die Kommutierung bezieht sich bei diesem Motor auf die Encoderkommutierung. Der Motorencoder ist auf die Motorwelle aufgeklebt. Bei einem Inkremental-Encoder muss die Encoderausrichtung nach dem Einschalten erst bestimmt werden. Dazu wird eine Drehung des Motors erzeugt und die Inkremente gezählt. Der Motorstrom wird auf die Lage der Motorwelle ausgerichtet.

**Start:** Um die Referenzfahrt zu starten wird in das Steuerwort der Wert 0x001F geschrieben [14, Tabelle 5.57]
**Referenz?:** Der Zustandsautomat verweilt so lange in diesem Zustand bis die Referenz gefunden wurde, dies signalisiert das Bit 15 des Statuswortes. Tritt in diesem Zustand ein Fehler auf, wird der Fehler-Zustand ausgeführt. Fehler werden durch ein fehlerhaftes Statuswort, durch eine EMCY oder durch ein Ausschalten der Achse erkannt. Sobald sich der Schlitten auf dem Referenzpunkt befindet oder der „Stop“ Button betätigt wird geht der Automat in den Stopp-Zustand über.
**Stopp:** Der Wert 0x000F der dem Steuerwort bei Abbruch dieses Fahrprogrammes übergeben wird, ist der gleiche der nach Referenzfindung übergeben werden muss [14]. Dieser „Normalzustand“ des Servoverstärkers wird im Stopp-Zustand ins Steuerwort geschrieben und daraufhin in den Warte-Zustand gewechselt.
**Fehler:** In diesem Zustand wird die Anzeige „Error“ unter dem Bereich „Homing“ in der GUI eingeschaltet und dem Steuerwort der Wert 0x000F und 0x0006 übergeben. Dies bewirkt zuerst ein Schalten in den Normalzustand und dann ein Ausschalten der Achse. Daraufhin wechselt der Zustandsautomat in den Warte-Zustand.
Ist der Fehler behoben und die Referenzfahrt wird erneut gestartet muss diese Anzeige im Betriebsart-Zustand quittiert werden. Ein Homing Error kann durch eine EMCY, einen Statusfehler, ein Ausschalten der Achse während der Referenzsuche oder durch eine nicht gefundene Kommutierung (Timer abgelaufen) verursacht werden.

### 5.3.2 Handbetrieb

Der Handbetrieb beinhaltet zwei verschiedene Positionierungen, die absolute und die relative. Diese beiden Fahrten sind durch die ANF_08 und ANF_09 definiert.

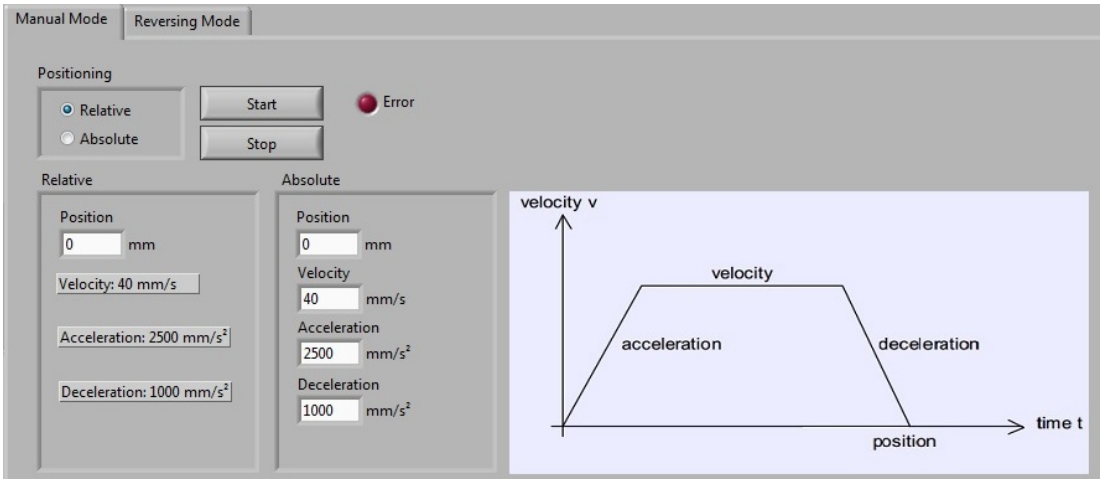



**Abbildung 5.7:** Registerkarte des Handbetriebs in der GUI

Der Handbetrieb kann erst nach erfolgreicher Kommutierung gestartet werden, die nach der Referenzfahrt gegeben ist. Sollte eine relative Positionierung vor der Referenzfahrt durchgeführt werden, muss vorher die Achse eingeschaltet und die Kommutierung gefunden worden sein. Eine Fehlermeldung weißt beim Start der relativen Positionierung, bei nicht erfolgter Kommutierung auf das Einschalten der Achse hin.
Vor der Referenzfahrt sind keine Softwareendlagen aktiv und der Schlitten kann auf die Lageendschalter fahren. Aus diesem Grund sind die Parameter Geschwindigkeit, Beschleunigung und Bremsverzögerung bei der relativen Positionierung auf ungefährliche, mit dem Auftraggeber besprochene, Werte voreingestellt.

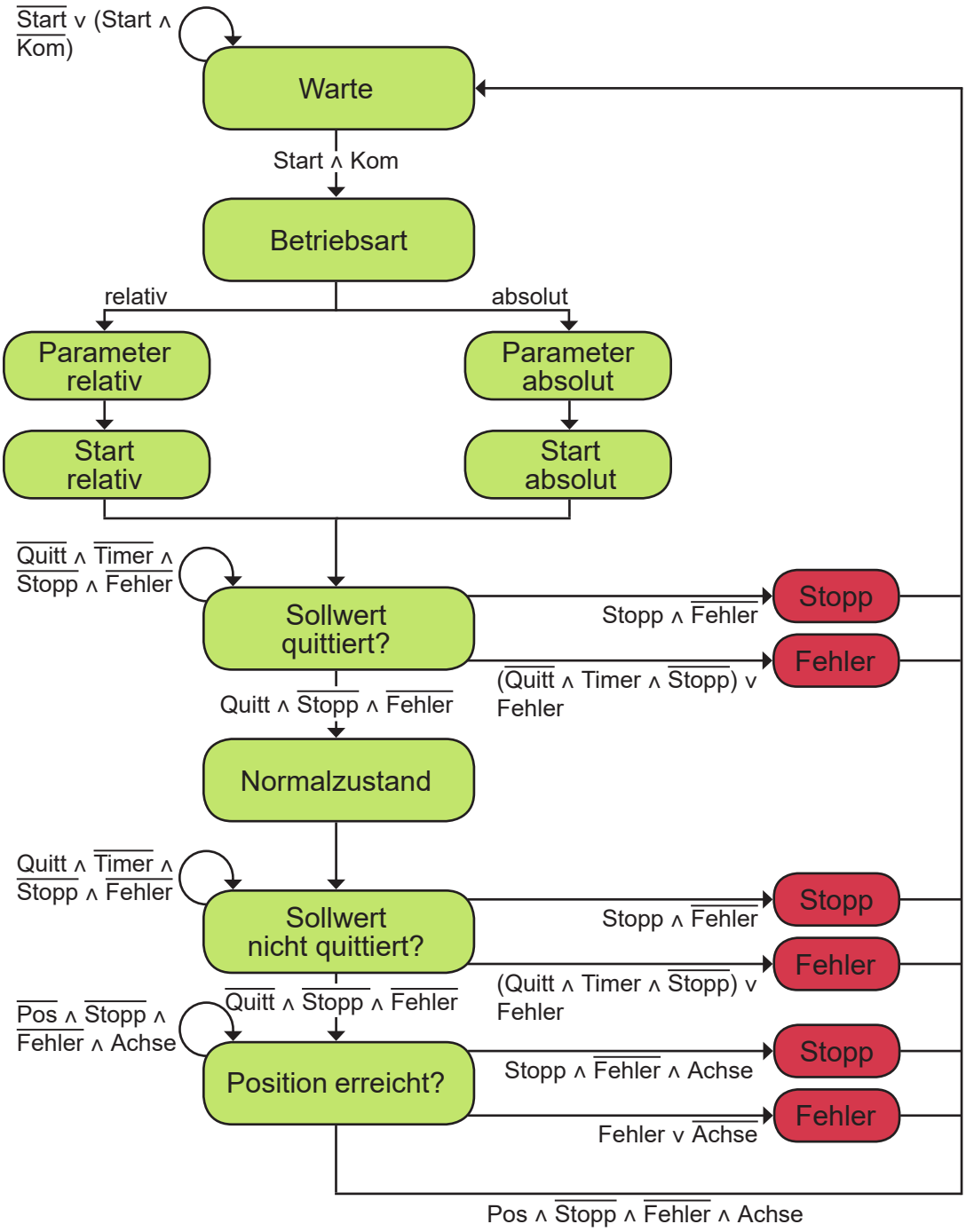



**Abbildung 5.8:** Zustandsautomat: Handbetrieb

**Warte:** Der Warte-Zustand ist so lange aktiv, bis der „Start“ Button vom Benutzer betätigt wird und die Kommutierung gefunden wurde.

**Betriebsart:** Die Betriebsart für den Positioniermodus hat den Wert 0x0001 [14]. Dieser wird gesetzt und die Achse eingeschaltet. Über ein Auswahlfeld in der GUI wird entschieden, ob es sich um eine relative oder absolute Positionierung handelt.

Je nach Auswahl werden die jeweiligen Zustände ausgeführt (vgl. Abbildung 5.8).
**Parameter:** Die in der GUI eingetragenen Parameter werden umgerechnet, in ein Hexadezimales Array umgewandelt und an die einzelnen Objekte übergeben. Diese Hexadezimale-Umwandlung wurde in einem Sub-VI umgesetzt. Diese Funktion wird mehrfach benötigt und kann somit an mehreren Stellen aufgerufen werden.
Die Einheiten in denen die Werte den einzelnen Objekten übergeben werden müssen sind dem *„Objektverzeichnis“* [14] zu entnehmen, ein Beispiel ist in Formel 5.1 gegeben. Bei der relativen Positionierung werden die vorher definierten Werte übergeben.
**Start:** Abhängig davon welche Positionierart gewählt wurde, muss am Steuerwort ein anderer Zustandswechsel erfolgen (absolut 0x000F $\rightarrow$ 0x001F, relativ 0x000F $\rightarrow$ 0x005F [14, Tabelle 5.57]). Zusätzlich wird der Timer gestartet, der für den nächsten Schritt benötigt wird.
**Sollwert quittiert?:** Dieser Zustand wird für sehr kleine Verfahrwege im Mikrometer Bereich benötigt. Ist der Weg sehr klein könnte im Zustand „Position erreicht?“ die Meldung das die Position erreicht wurde vom Ausgangspunkt stammen. Unter diesen Umständen hätte keine Bewegung stattgefunden. Dieser Effekt wird durch diese Abfrage vermieden.
Solange die eingestellte Timer-Laufzeit nicht abgelaufen ist, das Bit 12 des Statuswortes (Sollwert quittiert) nicht logisch 1 ist, der „Stop“ Button nicht betätigt wurde und kein Fehler vorhanden ist bleibt der Zustandsautomat in diesem Zustand. Läuft der Timer ab oder tritt ein Fehler auf, wird ein Wechsel in den Fehler-Zustand vollzogen. Ein Fehler wird durch ein fehlerhaftes Statuswort oder durch eine EMCY erkannt. Ein Sprung in den Stopp-Zustand ist nur durch betätigen des „Stop“ Buttons möglich.
**Normalzustand:** Für den Normalzustand muss dem Servoverstärker im Steuerwort der Wert 0x000F übergeben werden. Um im nächsten Zustand wieder eine Zeitbedingung zu realisieren, wird wieder ein Timer gestartet.
**Sollwert nicht quittiert?:** Dieser Zustand besitzt den gleichen Aufbau wie der Zustand „Sollwert quittiert?“, außer das hier auf eine logische 0 an Bit 12 des Statuswortes gewartet wird.
**Position erreicht?:** In diesem Zustand wird darauf gewartet, dass im Statuswort das Bit 10 gesetzt wird, also die Position erreicht wurde. Ist die Position erreicht ist das Fahrprogramm beendet und der Zustandsautomat wechselt in den Warte-Zustand. Höchste Priorität der Abbruchbedingungen haben wieder die Fehler EMCY und Statusfehler, zusätzlich wird abgefragt ob die Achse ausgeschaltet ist. Diese Bedingungen bewirken einen Übergang in den Fehler-Zustand. Durch den vorgesehenen Button wird in den Stopp-Zustand gesprungen.
Um sicherzustellen, dass keine Referenzfahrt gestartet wurde, muss abgefragt werden ob der Zustandsautomat der Referenzfahrt gestartet wurde und gegebenenfalls in den Warte-Zustand gewechselt werden.

**Stopp:** Durch den Wert 0x010F im Steuerwort (Bit 8: Halt) wird die Achse angehalten. Bevor der Zustandsautomat wieder in den Warte-Zustand wechseln kann muss der Normalzustand im Servoverstärker eingestellt werden, also das Steuerwort mit 0x000F beschrieben werden.
**Fehler:** Bei einem Fehler wird der Servoverstärker zuerst in den Normalzustand gebracht und daraufhin die Achse ausgeschaltet. Zusätzlich wird der Fehlerzustand in der GUI angezeigt. Beim Neustart wird diese Anzeige im Zustand „Betriebsart“ wieder quittiert.

### 5.3.3 Reversierfahrt

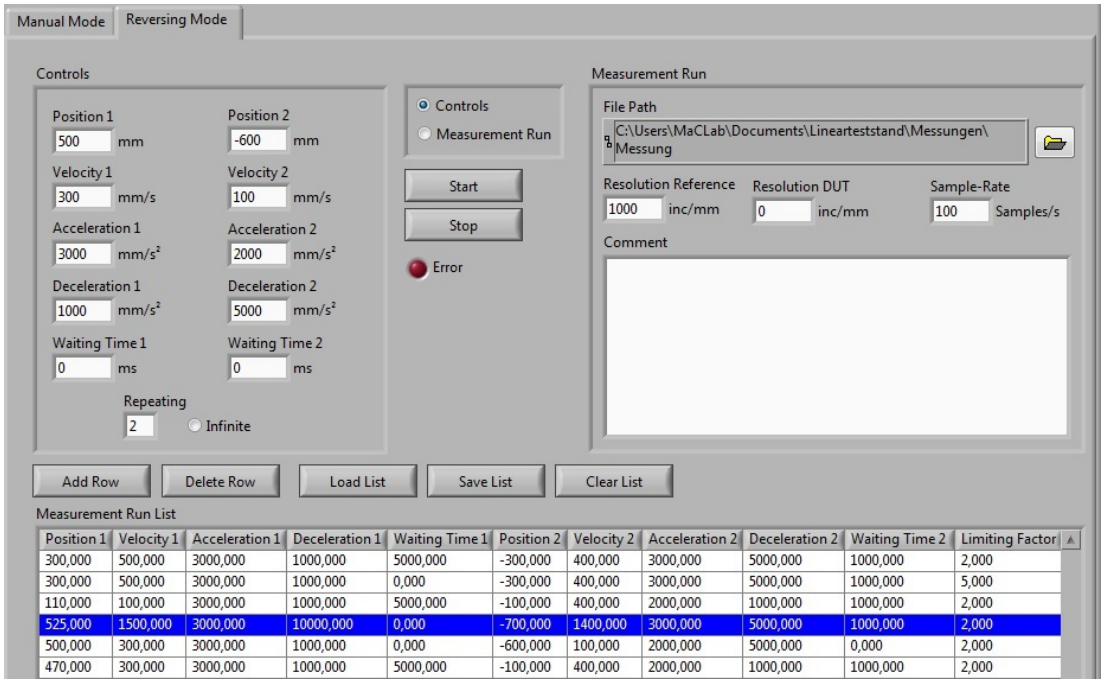
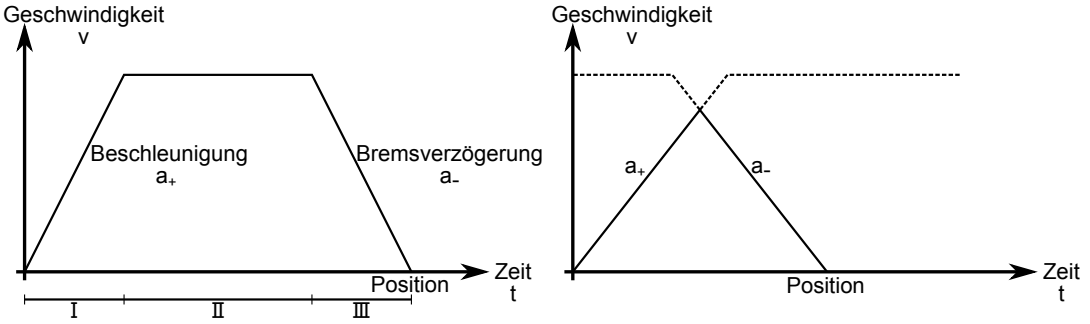



**Abbildung 5.9:** Registerkarte der Reversierfahrt in der GUI

Die Reversierfahrt dient zur eigentlichen Messprotokollaufnahme. In Abbildung 5.9 ist die Registerkarte der Reversierfahrt abgebildet. Diese kann mit den Parametern der Bedienelementen oder mit einer erstellten Liste, bestehend aus Fahrparametern, durchgeführt werden und ist über das Auswahlfeld „Controls“/„Measurement Run“ einstellbar.
Um die Vorgabe der ANF_10 zu realisieren, wird vor dem Start der Fahrt mit den Parametern der Bedienelemente getestet ob die angegebene Geschwindigkeit erreicht wird und gegebenenfalls eine Fehlermeldung ausgegeben.

Dieser Programmcode wurde wegen mehrfachem Aufruf in einem Sub-VI implementiert. Die Berechnung erfolgt nach Formel 5.7

Einschub: Überprüfung der Geschwindigkeit
Eine Fahrt an diesem Teststand zwischen zwei Positionen ist immer Trapezförmig (vgl. Abbildung 5.10 (a)).




(a) Geschwindigkeit wird erreicht (b) Geschwindigkeit wird nicht erreicht

**Abbildung 5.10:** Trapezprofil der Geschwindigkeit

Die Summe der Strecken der Beschleunigung und der Bremsverzögerung muss mindestens so groß sein wie die gesamte Strecke. Also die Strecke zwischen den beiden Positionen. Ist dies nicht der Fall wird, wie in Abbildung 5.10 (b) dargestellt vor erreichen der Geschwindigkeit abgebremst.
Die Kontrolle ob die Geschwindigkeit erreicht wird, wird aus der Formel 5.4 hergeleitet. Diese Formel wird jeweils zur Berechnung der Strecke der Beschleunigung ($a_+$) und zur Berechnung der Strecke der Bremsverzögerung ($a_-$) verwendet.

$$s_{\text{gesamt}} > s_{\text{I}} + s_{\text{III}} \quad \Rightarrow \quad s_{\text{gesamt}} > \left(\frac{1}{2} \cdot \frac{v^2}{a_+}\right) + \left(\frac{1}{2} \cdot \frac{v^2}{a_-}\right) \tag{5.6}$$

$$s_{\text{gesamt}} > \frac{v^2}{2}\left(\frac{1}{a_+} + \frac{1}{a_-}\right) \tag{5.7}$$

**Listenfunktionen:**
Des Weiteren ist es möglich eine Liste aus Fahrparametern zu erstellen. Während der Listen-Reversierfahrt oder auch Messfahrt wird ein Messprotokoll erstellt.
Eine Liste kann durch hinzufügen von bereits abgespeicherten Listen (Load List) und/oder durch die Parameter der Bedienelemente (Add Row) erstellt werden. Mit der Maus ist es möglich eine Zeile zu markieren und somit die Position zum Einfügen der Parameter, bzw. Listen festzulegen. Diese Liste entspricht der ANF_14 *„Testprogramm“*.

Beim Hinzufügen der Parameter der Bedienelemente wird wiederum geprüft ob die angegebene Geschwindigkeit erreicht werden kann, da die Bedienelemente Eingaben enthalten können, die vorher nicht abgefahren und geprüft wurden.
Nach Betätigung des „Load List“ Buttons wird ein Pop-up-Fenster geöffnet, um den Dateipfad der zu ladenden Datei auszuwählen. Das Schreiben und Lesen von Dateien wurde mit den Sub-VIs „Write To Spreadsheet File.vi“ und „Read From Spreadsheet File.vi“ umgesetzt. Beim Abspeichern einer Liste (Save List) wird wiederum durch ein Pop-up-Fenster angegeben unter welchem Pfad und Name die Datei gespeichert werden soll.
Der Startpfad im Fenster ist immer gleich und ist im Programmcode vorgegeben. Um zu vermeiden, dass eine vorhandene Datei überschrieben wird, wird ein Zählerindex inkrementiert.

Um die Überschreibung von vorhanden Dateien zu verhindern, wird kontrolliert ob die ausgewählte Datei vorhanden ist. Ist dies der Fall wird der Pfad mit einer LabVIEW Funktion zerlegt und der Dateiname untersucht.
In den letzten vier Stellen des Strings wird nach einem Unterstrich „_“ gesucht. Liefert die Suche kein Ergebnis wird eine „_1“ an die Datei angehängt. Nachdem ein Unterstrich gefunden wurde, wird die darauffolgende Zahl inkrementiert und im Dateinamen ersetzt. Dieser Programmcode befindet sich in einer While-Schleife und wird erst abgebrochen, wenn der neu erstellte Dateinamen nicht existiert.
Da in den letzten vier Stellen des Strings gesucht wird, kann nur bis 999 hochgezählt werden, zusätzlich darf der eigentliche Name der Datei in den letzten vier Stellen keinen Unterstrich enthalten.

Die Liste in der GUI kann über den Button „Clear List“ vollständig gelöscht werden oder über „Delete Row“ nur einzelne, ausgewählte Zeilen. Der „Clear List“ Button ist mit einem Pop-up-Fenster versehen, in dem erneut das Löschen der Liste bestätigt werden muss. Dies verhindert eine nicht gewollte Ausführung der Gesamtlöschung.
Die Listenfunktionen, können jederzeit ausgeführt werden und sind in der Hauptschleife implementiert worden. Um während der Fahrt die Änderung der grade laufenden Parameter zu verhindern, müssen alle Bedienelemente beim Starten der Reversierfahrt gesperrt und beim Beenden wieder freigegeben werden.

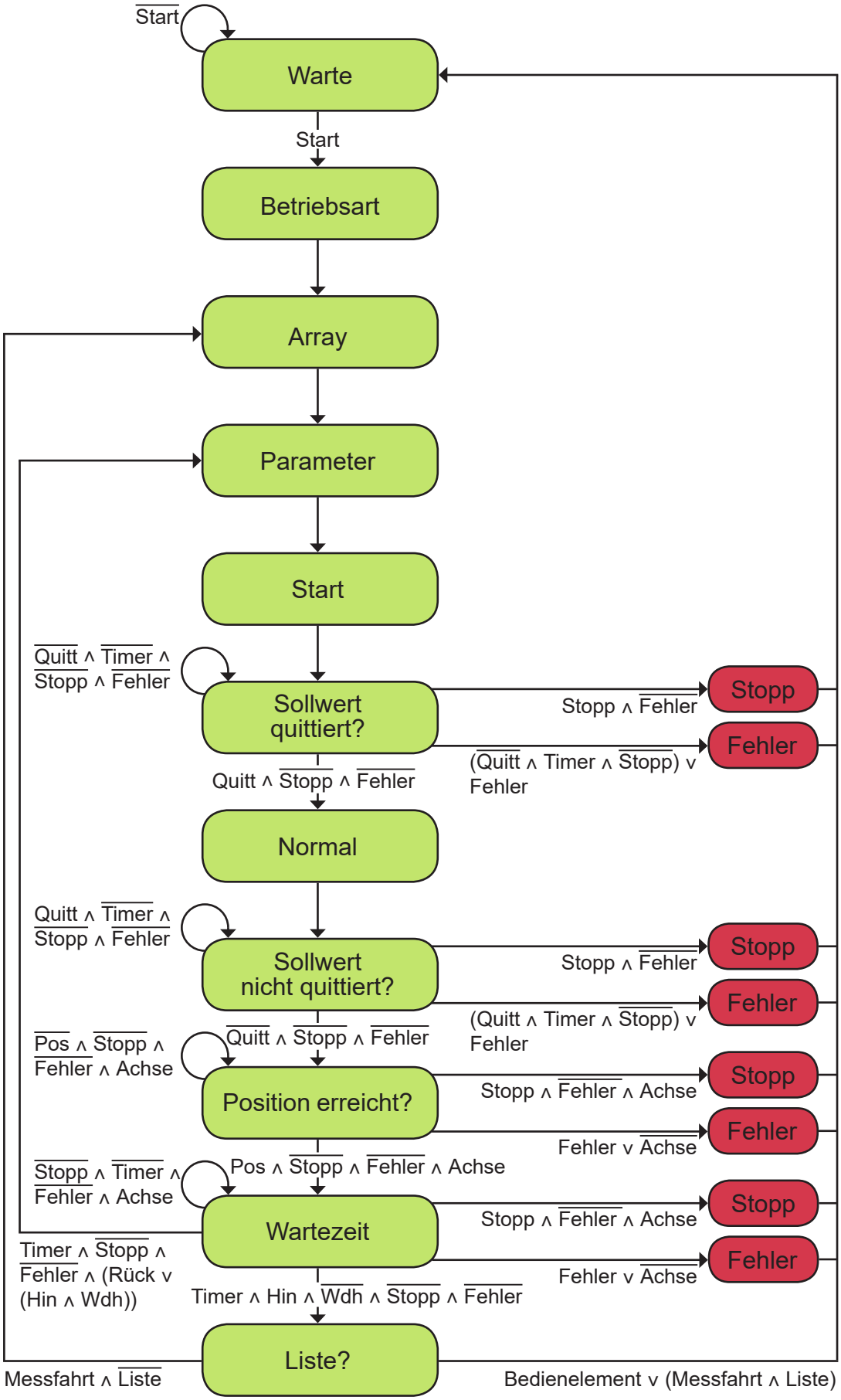



**Abbildung 5.11:** Zustandsautomat: Reversierfahrt

**Warte:** Nach Betätigung des „Start“ Button wird bei Auswahl der Bedienelemente die Überprüfung der Parameter durchgeführt und bei Auswahl der Liste geprüft ob die Liste leer ist und gegebenenfalls Fehlermeldungen ausgegeben. Der Übergang in den Betriebsart-Zustand wird nur bei korrekten Parametern vollzogen.
**Betriebsart:** In diesem Zustand wird wie bereits im Fahrprogrammen Handbetrieb beschrieben, die Betriebsart Positioniermodus eingestellt und die Achse eingeschaltet. Zusätzlich wir eine Hilfsvariable Index auf null gesetzt, die zur Abarbeitung der Liste dient.
**Array:** Die einzelnen Größen werden in diesem Zustand in die passenden Einheiten umgerechnet und in ein Array geschrieben. Ist das Auswahlfeld „Controls“ aktiv werden die Parameter der Bedienelemente eingetragen. Ist die Liste ausgewählt, wird je nach Hilfsvariable „Index“ die Werte der nächsten Zeile übergeben. Falls das Auswahlfeld „Infinite“ ausgewählt ist, wird die Konstante Unendlich ins Array übergeben.
Um die Wiederholungen der Fahrt zu zählen, wird eine Hilfsvariable „Wiederholung“ auf null gesetzt und eine zweite Hilfsvariable für die „Fahrtrichtung“ initialisiert. Mit diesen Informationen wird in späteren Schritten entschieden welcher Zustand als nächstes abgearbeitet wird.
**Parameter:** In diesem Zustand werden je nach dem welche Fahrtrichtung als nächstes abgefahren werden muss, die Parameter aus dem Array, dem Servoverstärker übergeben und die Hilfsvariable „Fahrtrichtung“ geändert.
**Start:** Da es sich um absolute Positionsangaben handelt wird der Wert 0x001F in das Steuerwort übergeben. Bevor der Zustandsautomat in den nächsten Zustand wechselt wird ein Timer gestartet.
**Positionierzustände:** Die nächsten Zustände „Sollwert quittier?“, „Normalzustand“, „Sollwert nicht quittiert?“ und „Position erreicht?“ sind identisch mit den gleichnamigen aus dem Fahrprogramm Handbetrieb. Allerdings bezieht sich die Abbruchbedingung Stopp auf den entsprechenden Button der Reversierfahrt.
**Wartezeit:** Nachdem die Position erreicht ist, wird der Zustand „Wartezeit“ ausgeführt. In diesem Zustand wird, je nach dem welche Fahrtrichtung in der Hilfsvariablen steht, die im Array vermerkte Zeit gewartet. Während dieser Zeit müssen Fehlerzustände, die bereits beschrieben wurden, zu einem Wechsel in den Fehler-Zustand führen und die Betätigung des „Stop“ Buttons zu einem Wechsel in den Stopp-Zustand.
Ist zum ersten Mal die Position 1 erreicht wird eine Variable gesetzt um die Messung zu starten.
Falls die Fahrtrichtung „Rückfahrt“ (Position 2 erreicht) aktiv ist, wird die Hilfsvariable „Wiederholung“ inkrementiert und der Automat springt in den Zustand „Parameter“ und beginnt erneut mit der Hinfahrt.
Ist ein Reversierdurchgang vorbei, Position 1 erneut erreicht, wird der Wert „Wiederholung“ mit der Angabe im Array verglichen.

Wurden noch nicht genügend Wiederholungen getätigt, springt der Automat in den Zustand „Parameter“ und beginnt erneut mit der Rückfahrt. Ist die Fahrt beendet, Position 1 erreicht, keine Wiederholungen mehr anstehend, wird in den Zustand Liste? gewechselt.
**Liste?:** Handelt es sich um eine Fahrt mit den Parametern der Bedienelemente wird sofort in den Warte-Zustand gewechselt. Wird allerdings gerade eine Messfahrt abgearbeitet, wird die Hilfsvariable „Index“ inkrementiert und überprüft. Ist die Liste vollständig abgearbeitet wird in den Warte-Zustand gewechselt, ist eine weitere Zeile vorhanden geht der Zustandsautomat in den Array-Zustand um dieses mit den neuen Parametern zu füllen.
**Abbruchzustände:** Die Zustände „Stopp“ und „Fehler“ besitzen den gleichen Aufbau, wie die gleichnamigen Zustände des Handbetriebes. Allerdings wird die Error-Anzeige des Reversiermodus aktiviert.

## 5.4 Datenerfassung

Die eingelesenen Impulse der Referenz und des DUT werden in einer DAT-Datei gespeichert. Diese besitzt einen Header, der in der ANF_12 definiert wurde. Die Datei wird abgespeichert, um später in MATLAB ausgewertet zu werden. Über die Eingabeelemente (Resolution Reference/DUT und Comment) der GUI werden die Parameter des Headers definiert und die ANF_13 gelöst (vgl. Abbildung 5.9).
Zusätzlich ist es möglich, die Sample-Rate der Messung einzustellen. Dieser Wert beschreibt wie viele Messwerte pro Sekunde aufgenommen werden (Samples/s). Wird die Sample-Rate beispielsweise auf den Wert 100 gesetzt, wird alle 10 ms ein Messwert erfasst.

$$1\ \text{s} = 1000\ \text{ms} \quad \Rightarrow \quad \frac{100\ \text{Samples}}{\text{s}} \Big/ \frac{1000\ \text{ms}}{1\ \text{s}} = \frac{1\ \text{Samples}}{10\ \text{ms}} \tag{5.8}$$

Da die Sate Machine ausführlich beschrieben wurde und diese sehr übersichtlich ist, wurde diese LabVIEW Architektur auch für die Datenerfassung gewählt. Zusätzlich können neue Zustände, für die Datenerfassung andere Testsysteme, hinzugefügt und beliebig zwischen diesen gewechselt werden.

Die Datenerfassung setzt sich aus drei aufeinanderfolgenden Zuständen zusammen.
**Warte:** Im Warte-Zustand wird darauf gewartet, dass die Variable, um die Messung zu starten, im Zustandsautomat der Reversierfahrt gesetzt wird.
**Header:** In diesem Zustand wird der in der GUI eingegebene Dateipfad überprüft. Hier wird der Programmcode der Listenfunktion zum Inkrementieren eines Zählerindex, um das Überschreiben einer Datei zu verhindern, erneut verwendet.

Der Programmcode wird bei der Datenerfassung erweitert. Damit die Datei von MATLAB ordnungsgemäß erkannt wird, muss diese die Endung „.dat“ besitzen. Dazu wird diese Endung dem in der GUI eingegebenen Dateinamen angehängt.
Nach der Überprüfung ob die Datei vorhanden ist und vor der Suche nach dem Zählerindex im Namen der Datei, muss diese entfernt werden. Nach dem Inkrementieren des Index und vor der erneuten Überprüfung (Datei vorhanden?) muss die Endung wieder angehängt werden.
Zusätzlich wird in diesem zweiten Zustand der Header in die Datei geschrieben. Dies erfolgt durch ein Zusammensetzen der geforderten Informationen in Strings und das Beschreiben der Datei mit dem bekannten „Write To Spreadsheet File.vi“. Um das aktuelle Datum und die Uhrzeit zu erhalten sind in LabVIEW bestimmte Funktionen vorgesehen [18].
**Datenerfassung:** Im dritten Zustand werden die Impulse der Encoder in die Datei geschrieben. Die Encoder werden über zwei verschiedene Tasks (PXIe-6356 Ctr0/Ctr1) eingelesen, diese müssen mit der angegebenen Sample-Rate synchronisiert werden.
Damit die Synchronisation echtzeitfähig ist, wird eine Sample Clock verwendet, die unabhängig vom Betriebssystem des Computers auf der Karte läuft. Die Clock muss zusätzlich unabhängig von den beiden Tasks der Encoder realisiert werden. Dazu wurde ein dritter Task erstellt, der den ersten anlogen Eingang der Karte enthält. Somit kann die Sample Clock dieses Task für die Synchronisation der anderen beiden verwendet werden.
Die drei Tasks werden zur Laufzeit mit Hilfe des „DAQmx - Kanal erzeugen.vi“ erstellt (siehe Abbildung 5.12). Wie bereits im Abschnitt 4.2.1 unter der Software MAX beschrieben wurde, wird für die Kodierung „X4“ verwendet und der Wert 4 für den Parameter „Abstand pro Impuls“ angegeben. Die eigentliche Synchronisierung erfolgt über das Sub-VI „DAQmx - Timing.vi“. Diesen wird die Sample Clock des analogen Eingangs und die Sample-Rate zugewiesen. Um die drei Tasks zu starten wird das „DAQmx - Tasks starten.vi“ verwendet. Die Synchronisation wurde in ein Sub-VI ausgelagert.

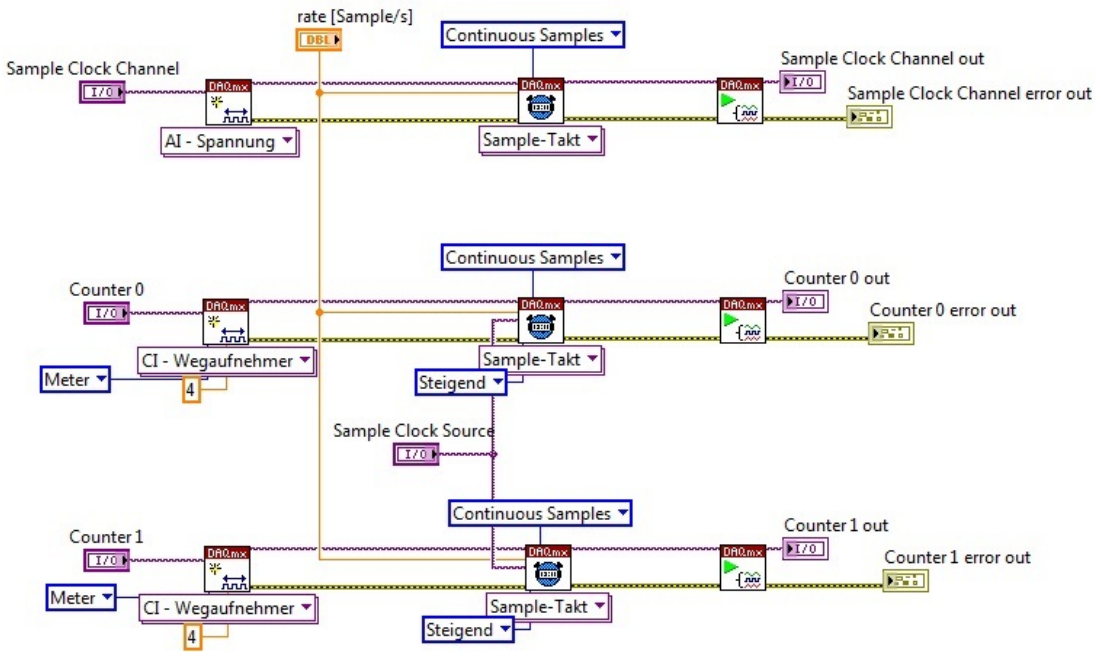



**Abbildung 5.12:** Synchronisation der Encoder-Tasks

Die Messdaten werden in der NI PXI-6356 in ein Buffer geschrieben. Dieser wird in LabVIEW in einer Schleife mit dem „DAQmx - Lesen.vi“ alle 100 ms in die Datei geschrieben. Ein Abbruch der Schleife wird durch eine Logische 0 in der Hilfsvariablen, die auch zum Starten der Messung verwendet wurde, realisiert.
Die Variable im Zustandsautomat der Reversierfahrt wird in jedem Abbruchzustand und beim Beenden der Messfahrt auf „low“ gesetzt und somit die Messung beendet. Zusätzlich wird die Schleife beim Auftreten eines Fehlers der DAQmx VIs abgebrochen. Nachdem die Schleife beendet wurde, werden die einzelnen Tasks geschlossen (DAQmx - Task zurücksetzten.vi) und der Fehlerausgang mit dem „Simple Error Handler.vi“ verbunden. Der Zustandsautomat springt in den Warte-Zustand.

# 6 Verifikation und Validierung

Die Verifikation stellt sicher, dass alle Anforderungen aus dem Lastenheft korrekt implementiert wurden, das heißt die Anforderungen wurden so umgesetzt, wie sie angegeben und formuliert worden sind [6]. Zusätzlich dient die Verifikation zur Fehlerfindung und -behebung. Dazu wurde ein Verifikationsplan erstellt, der sich auf der beigefügten CD befindet. Der Verifikationsplan besteht aus Testfällen, die nach der vollständigen Implementierung abgearbeitet wurden. Falls Veränderungen in der Software vorgenommen werden, muss diese erneut durchgegangen werden.

## 6.1 Verifikationsplan

Ein erster Entwurf des Verifikationsplans wurde bereits während der Projektphase „Design & Architektur“, in der die Realisierbarkeit der Anforderungen geprüft wurde, erstellt. Diese erste Version enthält Testfälle, um die einzelnen Spezifikationen einer Anforderung zu prüfen und am Ende die korrekte Funktion nachzuweisen [6].
In der Projektphase „Umsetzung“ wurde der Verifikationsplan stetig erweitert. In dieser Phase wurden Testfälle dokumentiert, die sich bei der Implementierung der einzelnen Anforderungen aufzeigten. Ein besonders wichtiger Teil war die Niederschrift von Testfällen, in Bezug auf die Wechselwirkung zwischen verschiedenen Anforderungen. So muss es zu jeder denkbaren Fehlbedienung der Software einen Testfall geben. Wie zum Beispiel, dass während eines Fahrprogrammes die anderen nicht gestartet werden können.

**Aufbau:**
Der erstellte Verifikationsplan gliedert sich in fünf Spalten (Abbildung 6.1). Die ersten beiden enthalten die ID und den nichttechnischen Namen der Anforderung. Daraufhin folgt die Spalte Verifikation der Anforderung. In ihr wird die Durchführung der einzelnen Testfälle beschrieben. Das Ende bilden die Spalten Bemerkung und Hilfsmittel. Unter Bemerkung wird der Testfall nach erfolgreicher Durchführung abgehakt und wenn nötig eine Bemerkung verfasst. Die letzte Spalte definiert die Hilfsmittel die für den entsprechenden Testfall benötigt werden.

## 6.2 Ergebnis

Nur durch die Ergebnisse der Verifikation lässt sich eine Aussage über das Gesamtergebnis des Projektes treffen. Bei der Verifikation sind Probleme aufgetreten, die sich vorher nicht gezeigt haben. Diese wurden behoben und die Testfälle erneut geprüft.
Letztendlich wurden alle Testfälle erfolgreich abgearbeitet und lieferten somit die Beweisführung, dass das Produkt den Spezifikationen entspricht [6]. Lediglich kleinere Ausnahmen kristallisierten sich heraus, die im originalen Verifikationsplan unter Bemerkung beschrieben sind. Beispielsweise wurde festgestellt, dass obwohl dem Servoverstärker als Geschwindigkeit 10 mm/s übergeben wird, schwankt auf Grund der Mechanik, die tatsächliche Geschwindigkeit zwischen 9,6 und 11,2 mm/s. Dieser Auszug ist in Abbildung 6.1 dargestellt.

| Nr. / ID | Nichttechnischer Titel | Verifikation der Anforderung | Bemerkung | Hilfsmittel |
|---|---|---|---|---|
| ANF_03 | Systemstart und Referenzierung | Die gesperrten Bedienmöglichkeiten >Handbetrieb absolut >Modus Reversieren >Softwareendlagen können vor der Referenzfahrt nicht bedient werden | ☑ | |
| | | Erst nach Betätigung der vorgesehenen Taste der GUI setzt der Schlitten sich mit max. 10 mm/s in Bewegung | Die Geschwindigkeit schwankt aufgrund der Mechanik ☑ zwischen 9,6 und 11,2 mm/s | |

**Abbildung 6.1:** Auszug aus dem Verifikationsplan

Des Weiteren ist zu erwähnen, dass diese Verifikation nicht alle möglichen Fehler abdeckt. Die Anzahl der möglichen Testfälle ist im Rahmen dieses Projekts nicht klar definierbar. Bei dieser Verifikation handelt es sich um Testfälle die vom Projektleiter erkannt und als relevant erachtet wurden.

## 6.3 Validierung

Die Endabnahme erfolgte in Zusammenarbeit mit dem Auftraggeber. Dazu wurde ein Abnahmeprotokoll erstellt, welches sich auf der beigefügten CD befindet. Dem Auftraggeber wurden die Funktionen der einzelnen Anforderungen vor Ort demonstriert und Abnahmekriterien aufgezeigt.
Abnahmekriterien sind implementierte Funktionen, die bei der Bedienung und Handhabung des Produkts zu beachten sind. So wird zum Beispiel zur Vermeidung, dass eine Datei überschrieben wird, in den letzten vier Stellen des Dateinamens nach einem Unterstrich gesucht (vgl. Abschnitt 5.3.3/ Listenfunktionen). Folglich darf der eigentliche Dateiname in den letzten vier Stellen keinen Unterstrich enthalten.
Für eine erfolgreiche Validierung müssen diese Abnahmekriterien vom Auftraggeber akzeptiert werden. Des Weiteren muss vom Auftraggeber bestätigt werden, dass das Produkt die verfassten Anforderungen erfüllt.
Diese Kriterien wurden erfüllt und somit die Beweisführung für den Erfolg dieses Projektes geliefert [6].

# 7 Fazit

## 7.1 Zusammenfassung

Die Grundlegende Frage zum Ende dieses Projektes ist: „Wurde die Datenerfassung und Ansteuerung des Linearteststands mit Hilfe des NI Systems verbessert?“
Nach erfolgreicher Verifikation und Validierung lautet die Antwort „Ja!“. Die typischen Fehler, die mit dem alten System auftraten, wurden während der Testphase nicht festgestellt. Diese Fehler waren unter anderem, dass der Teststand im Stillstand schwingt und Reversierfahrten nicht vollständig abgearbeitet wurden. Allerdings lässt sich eine eindeutige Aussage darüber erst nach einem Langzeittest treffen.

Die im Vorfeld strategisch durchdachte Vorgehensweise mit Hilfe des V-Modells, ist ein grundlegender Bestandteil für den Erfolg dieses Projektes. Auswirkungen und Fehlerquellen konnten durch die im Vorfeld geleistete „Spezifikation“ früh erkannt und dem zeitlichen Aufwand angepasst werden.

Außerplanmäßige Probleme, die während der „Umsetzung“ aufgetreten sind, wie zum Beispiel die Behebung des Fehlerzustands der NI Karte nach Wiedereinschalten des Servoverstärkers (vgl. Abschnitt 5.1), wurden analysiert und gelöst. Diese Probleme führten jedoch zu einem größeren Zeitaufwand der einzelnen Meilensteine. Durch den vorher konstruierten Meilensteinplan konnte in den anderen Unterprojektzielen entsprechend darauf reagiert und ein nicht erreichen des Projektendtermins verhindert werden.

Alternativen, die sich zu einzelnen Funktionen während der „Umsetzung“ aufzeigten, wurden vom Projektleiter analysiert, Vor- und Nachteile ausgearbeitet und in Absprache mit dem Auftraggeber Entscheidungen getroffen. So fiel zum Beispiel die Wahl für die Implementierung der Fahrprogramme auf die State Machine. Alternativ kann eine Sequenzprogrammierung im Servorverstärker implementiert werden. Diese verfügt über Vorteile im Bezug auf die Laufzeit, allerdings hätte man keinen Einfluss auf die einzelnen im Servoverstärker ablaufenden Zustände. Die Wahl der Softwarearchitektur (State Machine) hat sich bis zum Schluss bewährt.

Die Aufteilung der Software in Teilkomponenten, die bereits in der Planung für sinnvoll erachtet wurde, brachte einen großen Vorteil im Hinblick auf den Zeitaufwand in der letzten Projektphase. Durch die einzelnen Sub-VIs und Zustandsautomaten konnte ein frühzeitiger Funktionstest und somit eine Fehlererkennung und -behebung durchgeführt werden. Eine mögliche Fehlersuche in einem einzelnen Gesamtsystem wäre zu Zeitintensiv.

## 7.2 Ausblick

Zur Realisierung eines Folgeprojektes ist eine Einarbeitung in die implementierte Software und eine anforderungsspezifische Erweiterung notwendig. Mit Hilfe dieses Dokuments und dem auf der beigefügten CD befindlichen Programmcodes ist diese Software rekonstruierbar. Im Programmcode selbst sind zahlreiche Kommentare mit Erklärungen des Codes vermerkt.
Im Labor für elektrische Antriebe der Hochschule Bochum befindet sich ein rotativer Teststand, dessen Motoren über Ecosteps, der Firma Jenaer Antriebstechnik GmbH, angesteuert werden. Die Ansteuerung und Datenerfassung für diesen rotativen Teststand, soll laut ANF_06 mit Hilfe des gleichen GUI realisiert werden. Eine Unterscheidung der Teststände wird Anhand eines Objektes im CANopen Protokoll getroffen. Dieses liegt in beiden Servoverstärkern (Ecovario und Ecostep) mit unterschiedlichen Werten vor. Durch den Wert des Objektes, kann die Software selbständig die Bedienelemente für den entsprechenden Teststand freischalten. Der vorhandene Programmcode muss durch die Funktionen des rotativen Teststands erweitert werden.

Der Lineartteststand dient zur Verifikation von Velocimetern, dazu sind wie beschrieben verschiedene Schnittstellen im Schaltschrank vorgesehen. Dennoch muss für jede Verbindungsart eine Schnittstelle, in Form einer Karte, zum NI System hinzugefügt werden. Bei Verwendung des CAN-Bus geschieht dies über die NI PXI-5812, zusätzlich bietet die NI PXIe-1435 die Möglichkeit Bildmaterial eines Kamera unterstützten Testsystems einzulesen.
Um die Datenerfassung für verschiedene DUT in LabVIEW zu implementieren, kann der verwendete Zustandsautomat der Datenerfassung genutzt werden. Dieser bietet die Möglichkeit, für verschiedene Testsysteme jeweils einen neuen Zustand zu definieren. Zwischen diesen kann beliebig gewechselt werden und der Programmcode früherer Systeme kann bestehen bleiben. Die daraus resultierenden Vorteile waren ausschlaggebend für die Auswahl einer State Machine für die Datenerfassung.

Wie bereits in Abschnitt 4.1.2 beschrieben wurde, ist der Magnetsensor am Servoverstärker angeschlossen. Zurzeit wird als Referenz für die Fahrprogramme der Motorencoder verwendet. Es ist möglich den vorhanden Programmcode zu erweitern und eine Auswahlmöglichkeit zu schaffen, um auch den Magnetsensor als Referenz für die Fahrprogramme auszuwählen. Dazu muss dem Servoverstärker der Wechsel der Schnittstelle des Encoders mitgeteilt und alle Umrechnungsfaktoren, die auf die Auflösung des Motorencoders angepasst sind, auf die entsprechende Auswahl angepasst werden.

Die Regelparameter des Ecovarios sind auf die Spezifikationen, die für das Intacton System (aktuelles DUT) überprüft werden sollen, abgestimmt. Dies bedeutet, dass der Servoverstärker auf sehr hohe Geschwindigkeiten ausgelegt ist.
Die Vorgabe für das vorhandene DUT ist, dass eine Messfahrt über einen längeren Zeitraum mit sehr hohen Geschwindigkeiten gefahren werden soll. Da diese hohen Geschwindigkeiten dazu führen können, dass eine exakte Positionierung nicht umsetzbar ist, wurde der Schleppfehler angepasst. Wird der Schleppfehler zu niedrig gewählt, wird bei Überschreiten des Schleppfehlers, in Bezug auf die Zielposition, die Messfahrt abgebrochen und der Servoverstärker geht in einen Fehlerzustand. Das Bestreben liegt hierbei auf einer hohen Geschwindigkeit, unter Vernachlässigung einer genauen Positionierung. Diesbezüglich wurde der Schleppfehler auf 100 mm festgelegt. Dies impliziert eine maximale Abweichung von der Zielposition um 100 mm, allerdings wird kein Fehler erkannt und die Messfahrt fortgesetzt. Für den Vergleich der Messwerte ist dies unerheblich, da das DUT und die Referenz beide diese Abweichung erfassen.
Wird es notwendig, dass der Schlitten eine sehr genaue Positionierung, mit sehr geringer Fehlerabweichung, vornehmen muss, müssen die Regelparameter angepasst werden. Eine ausführliche Beschreibung zur Parametrierung, bietet das Kapitel Inbetriebnahme im Bedienhandbuch des Ecovarios [19].

Da der Programmcode auf dem angeschlossenen Computer läuft, ist keine Echtzeiterfassung möglich. Eine starke Beanspruchung der CPU des Computers kann zur Verlangsamung der Software führen. Aus diesem Grund ist nur die Datenerfassung, die über eine Sample Clock auf der NI Karte läuft, echtzeitfähig. Um die in der GUI angezeigten Stati, Ist-Geschwindigkeit, -Position, etc. echtzeitfähig zu machen, könnte eine externe CPU im NI System eingesetzt werden.

# Abkürzungsverzeichnis

| | |
|---|---|
| **ANF** | Anforderung |
| **API** | Application Programming Interface |
| **CAN** | Controller Area Network |
| **CiA** | CAN in Automation |
| **COB** | Communication Object |
| **DUT** | Device under Test |
| **EMCY** | Emergency Message |
| **GUI** | Graphical User Interface, Benutzeroberfläche |
| **ISO** | International Standardization Organization |
| **LabVIEW** | Laboratory Virtual Instumentation Engineering Workbench |
| **MAX** | Measurement & Automation Explorer |
| **NI** | National Instruments |
| **NMT** | Network Management |
| **OSI** | Open-Systems-Interconnection |
| **PDO** | Process Data Object |
| **Rx** | Receive |
| **SDO** | Service Data Object |
| **Tx** | Transfer |
| **VI** | Virtual Instrument |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quellenverzeichnis

## Literatur

[1] *Mess- und Prüftechnik Allgemeines, Arbeitsgebiete, Projektbeispiele*. VDEh-Betriebsforschungsinstitut (BFI). Düsseldorf, 2011.

[2] National Instruments. *DAQ NI SCB-68A User Manual*. 2012.

[3] A. W. Koch. *Optische Messtechnik an technischen Oberflächen: praxisorientierte lasergestützte Verfahren zur Untersuchung technischer Objekte hinsichtlich Form, Oberflächenstruktur und Beschichtung*. Expert Verlag, 1998.

[4] W. Jakoby. *Projektmanagement für Ingenieure Ein praxisnahes Lehrbuch für den systematischen Projekterfolg*. 3. Auflage. Wiesbaden: Springer Vieweg, 2014.

[5] D. Prof. Dr. Feldmüller u. a. *Einführung in das Projektmanagement*. Bochum: IBKN-Lehrveranstaltung, WS 2013/2014.

[6] M. Grande. *100 Minuten für Anforderungsmanagement Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. 2. Auflage. Wiesbaden: Springer Vieweg, 2014.

[7] A. Bergmann. *Lastenheft: Linearteststand optische Velocimeter*. V1.0. Bochum, 2015.

[8] B. Scherff u. a. *Feldbussysteme in der Praxis: Ein Leitfaden für den Anwender*. Berlin: Springer-Verlag, 2013.

[9] W. Lawrenz und N. Obermöller. *CAN Controller Area Network: Grundlagen, Design, Anwendungen, Testtechnik*. 5. Auflage. Berlin: VDE Verlag, 2011.

[10] K. Etschberger. *Controller-Area-Network: Grundlagen, Protokolle, Bausteine, Anwendungen*. 3. Auflage. München, Wien: Carl Hanser Verlag, 2002. ISBN: 9783446217768.

[11] Dipl.-Ing. Stefan Jonker. *Prozessleittechnik-Teil 1: ISO/OSI und TCP-Schichtenmodell*. Bochum: Bochum University of Applied Sciences, 2015.

[12] W. Kriesel u. a. *Bustechnologien für die Automation: Vernetzung, Auswahl und Anwendung von Kommunikationssystemen*. Heidelberg: Hüthig Verlag, 1998.

[13] U. Koppe. *AN1201 Einführung in CANopen*. MicroControl GmbH & Co. KG. Revision 02. Troisdorf, 2014.

[14] Jenaer Antriebstechnik GmbH. *Objektverzeichnis: ECOVARIO, ECOSTEP, ECOMPACT, ECOMiniDual*. Jena, März 2014.

[15] G. Schnell. *Sensoren in der Automatisierungstechnik*. 2. Auflage. Braunschweig: Vieweg Verlag, 1993.

[16] SIKO GmbH. *Originalmontageanleitung Magnetsensor MSK1000*. Buchenbach, 2015.

[17] LEG Industrie-Elektronik GmbH. *Signal Converter 3 channels RS422 into 24V (HTL) / 5V (TTL)*. Viersen-Dülken, 2015.

[18] W. Georgi und E. Metin. *Einführung in LabVIEW*. 5. Auflage. München: Carl Hanser Verlag, 2012.

[19] Jenaer Antriebstechnik GmbH. *ECO Studio Bedienhandbuch*. Jena, November 2015.

[20] National Instruments. *NI-XNET Hardware and Software Manual*. Juli 2015.

[21] J. Rybach. *Physik für Bachelors*. 3. Auflage. Leipzig: Carl Hanser Verlag, 2013.

# Standards

[DIN 69901] *Projektmanagement - Projektmanagementsysteme*. DIN. 2009.

[ISO 7498-1] *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. ICS 35.100.00. ISO, 1994.

[IS0 11898] *Road vehicles - Controller area network (CAN)*. ICS 43.040.15. ISO.

[CiA 301] *CANopen: Application Layer and Communication Profile*. Version 4.01. CAN in Automation e.V., 2000.

[CiA 303] *CANopen: Cabling and Connector Pin Assignment*. Version 1.0. CAN in Automation e.V., 1999.

# Online-Quellen

[Onl1] Franz Graser. »Mess- und Prüfmaschinen schneller und einfacher automatisieren«. In: *Elektronik Praxis* (26.10.15). Stand: 07.12.2015. URL: http://www.elektronikpraxis.vogel.de/iot/industrie40/articles/509181/.

[Onl2] ME-Meßsysteme GmbH. *CAN Bus Grundlagen*. Stand: 22.11.2015. URL: http://www.me-systeme.de/canbus.html.

[Onl3] Vector Academy. *CAN: Controller Area Network*. Stand: 07.11.2015. URL: https://elearning.vector.com/vl_can_introduction_de.html.

[Onl4] National Instruments. *Anschließen von Quadratur-Encodern an ein DAQ-Gerät*. Stand: 19.01.2016. URL: https://www.ni.com/getting-started/set-up-hardware/data-acquisition/d/quadrature-encoders.

[Onl5] National Instruments. *Getting Started with the NI-XNET API for LabVIEW*. Stand: 21.12.2015. URL: http://www.ni.com/product-documentation/12375/en/.

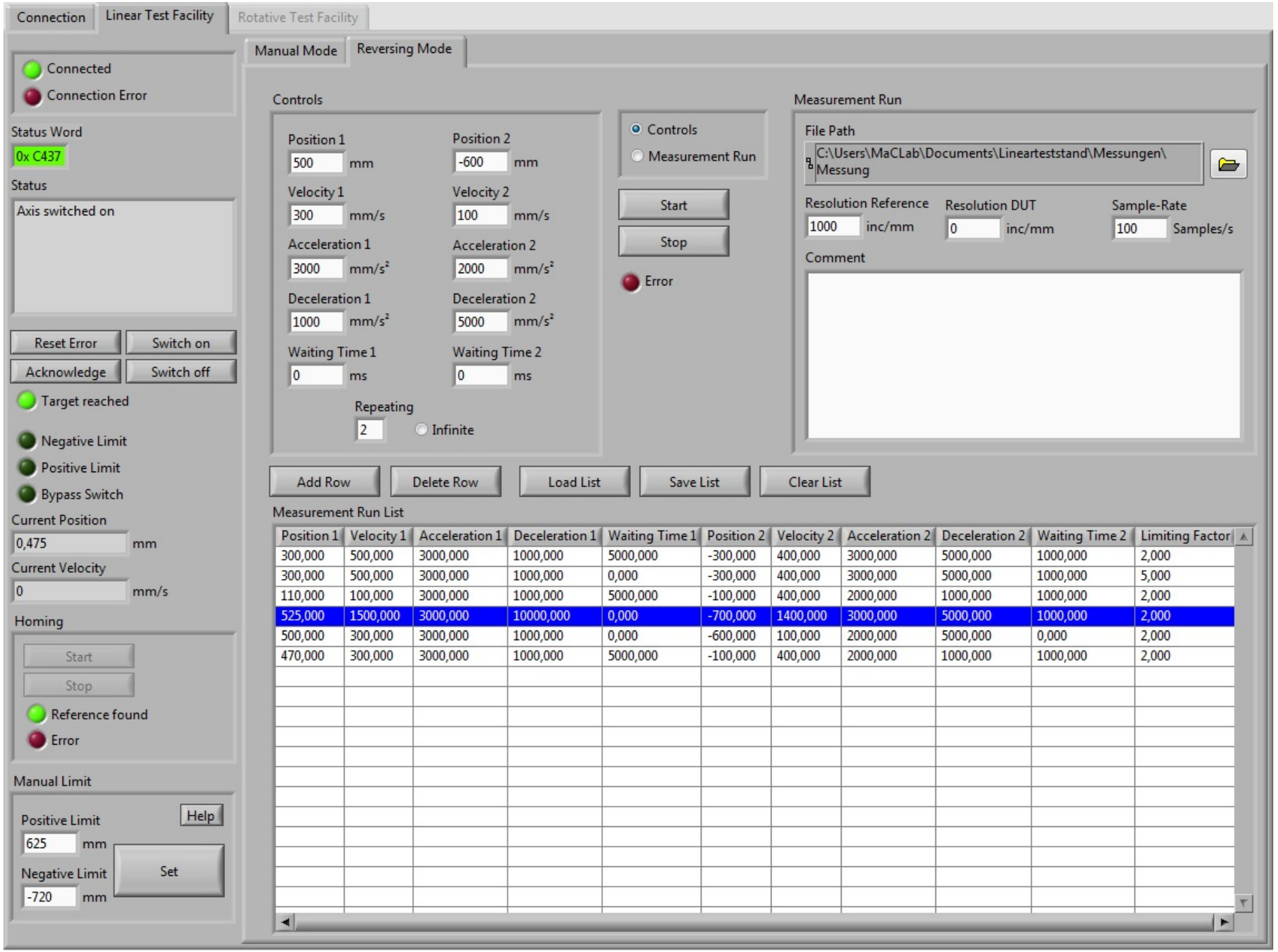
[Onl6] National Instruments. *Tutorial: State Machines*. Stand: 22.12.2015. URL: http://www.ni.com/tutorial/7595/en/.

# Anhang

## Inhalt: CD-ROM

| Ordner/Datei | Kurzbeschreibung |
|---|---|
| Linearteststand Applikation (EXE) | Zum Verwenden der Applikation ohne die Software LabVIEW. |
| Linearteststand_LabVIEW-Projekt | Das LabVIEW-Projekt enthält den implementierten Programmcode. |
| Ecovario 414 Backups | Drei Zustände des Servoverstärkers können wiederhergestellt werden.<br>- Werkseinstellungen<br>- Zustand vor diesem Projekt<br>- Zustand nach diesem Projekt |
| Literatur & Datasheets | Digital verfügbare Literatur und Datenblätter der Betriebsmittel. |
| Lastenheft | Entwurf und digitalisiertes Original*. |
| Verifikationsplan | Entwurf und digitalisiertes Original*. |
| Abnahmeprotokoll | Entwurf und digitalisiertes Original*. |
| Bachelorarbeit | Bachelorarbeit als PDF. |

*Papierausdruck befindet sich im Projektordner




Benutzeroberfläche/ GUI

Hochschule Bochum
Bochum University
of Applied Sciences

**Lastenheft**

# Lineartestand optische Velocimeter

**Version 1.0**

| | | | |
|---|---|---|---|
| Autor des Dokuments | Arno Bergmann | Erstellt am | 17.11.2016 |
| Dateiname | Lastenheft_Lineartestand | | |
| Seitenanzahl | |  | **Vertraulich!** |

# Historie der Dokumentversionen

| Version | Datum | Autor | Änderungsgrund / Bemerkungen |
|---|---|---|---|
| 0.1 | 17.11.15 | Arno Bergmann | Ersterstellung |
| 0.2 | 23.11.15 | Lukas Gehrmann | Überarbeitung der Anforderungen und des Verifikationsplan |
| 0.3 | 26.11.15 | Arno Bergmann<br>Thorsten Bartsch<br>Lukas Gehrmann | Überarbeitung und Erweiterung durch die ANF_15 |
| 1.0 | 09.12.15 | Lukas Gehrmann | Erweiterung der ANF_15 |
| | | | |
| | | | |
| | | | |

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Allgemeines

### 1.1.1 Zweck und Ziel dieses Dokuments

Dieses Lastenheft führt alle Anforderungen gegen den im Labor für elektrische Antriebe befindlichen Lineareteststand zur Verifikation von optischen Velocimetern.

Damit wird dieses Dokument vollständig lösungsneutral gehalten und beinhaltet keine Vorgaben zu Terminen, Budgets und Systementwürfen.

### 1.1.2 Abkürzungen

| | |
|---|---|
| FMEA | Failure Mode and Effects Analysis |
| DAQ | Data Acquisition |
| DUT | Device under Test |
| GUI | Graphical User Interface, Benutzeroberfläche |

## 1.2 Verteiler und Freigabe

### 1.2.1 Verteiler für dieses Lastenheft

| Rolle / Rollen | Name | Telefon | E-Mail | Bemerkungen |
|---|---|---|---|---|
| Projektleiter | Lukas Gehrmann | | lukas.gehrmann@HS-Bochum.de | |
| Auftraggeber | Thorsten Bartsch | | thorsten.bartsch @HS-Bochum.de | |
| Betreuender Professor | Arno Bergmann | 0234 / 32 103 50 | arno.bergmann@HS-Bochum.de | |

## 1.3 Reviewvermerke

## 1.4 Projektziele

Ziel ist der Austausch der Datenerfassung (DAQ) und Ansteuerung des Lineareteststands der Firma FRABA durch ein National Instruments System.

Das fertige National Instruments System kann den Teststand gemäß der Anforderungen in diesem Lastenheft ansteuern und die Positionsdaten des zu testenden Systems (DUT) mit denen des Referenzgebers zur weiteren Auswertung in einem csv-Format abspeichern.

Der gesamte Teststand wird nachträglich sehr wahrscheinlich durch noch unbekannte Anforderungen ergänzt werden, bspw. redundantes Referenzsystem, weitere DUT-Schnittstellen, etc.

## 1.5 Ziele und Nutzen des Anwenders

Anwender des Systems sind die Entwickler von optischen Velocimeter, die Spezifikationen dieser Systeme an diesem Teststand überprüfen.

# 2 Liste der Anforderungen

## 2.1 Statuswortanzeige

| **Nr. / ID** | ANF_01 | **Nichttechnischer Titel** | Statuswortanzeige | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.1.1 Beschreibung

Während des gesamten Betriebs der Testanlage ist das aktuelle Statuswort des Frequenzumrichters in der Benutzeroberfläche sichtbar, mindestens die im Normalbetrieb vorkommenden Stati werden in Klartext übersetzt. Diese Anzeige ist nicht echtzeitfähig.

Normalbetrieb wird durch ein grünes, Fehlerzustände durch ein rotes Statuswort kenntlich gemacht.

### 2.1.2 Wechselwirkungen

### 2.1.3 Risiken

Die Statuswortabfrage ist nicht echtzeitfähig, wodurch dem Benutzer ein möglicherweise kritischer Zustand der Anlage entgeht.

### 2.1.4 Testhinweise

Stati die im Normalbetrieb vorkommen sind dem Objektverzeichnis des Servoverstärkers zu entnehmen.

### 2.1.5 Grobschätzung des Aufwands

Mittel.

## 2.2 Einhalten der Lageendschalterposition

| **Nr. / ID** | ANF_02 | **Nichttechnischer Titel** | Einhalten der Lageendschalterposition | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.2.1 Beschreibung

Der Fahrschlitten verlässt nie den Positionsbereich zwischen den Lageendschaltern. Fährt der Fahrschlitten auf eine der beiden Lageendschalter wird dieser gestoppt und das Fahrprogramm abgebrochen. In der GUI wird angezeigt, dass der Fahrschlitten sich auf einem der beiden Lageendschalter befindet.

### 2.2.2 Wechselwirkungen

Mit allen Anforderungen.

### 2.2.3 Risiken

Personenschaden, mechanische Zerstörung des Testsands.

### 2.2.4 Testhinweise

### 2.2.5 Grobschätzung des Aufwands

Gering.

## 2.3 Systemstart und Referenzierung

| **Nr. / ID** | ANF_03 | **Nichttechnischer Titel** | | Systemstart und Referenzierung | | |
|---|---|---|---|---|---|---|
| **Quelle** | | | **Verweise** | | **Priorität** | muss |

### 2.3.1 Beschreibung

Der gesamte Teststand befindet sich im sicheren Zustand (Antrieb ist gesperrt). Über die GUI kann ausschließlich eine Referenzfahrt mit max. 10 mm/s angefordert werden. Die Referenzfahrt wird durch den Benutzer in der GUI gestartet.

Über die Referenzfahrt wird der mittlere Lageschalter angefahren und die Ist-Position auf den Wert 0 mm gesetzt.

Nach der Referenzfahrt werden die weiteren Bedienmöglichkeiten der GUI freigeschaltet und die Referenzierung optisch in der GUI kenntlich gemacht.

### 2.3.2 Wechselwirkungen

Freischalten aller weiteren Bedienmöglichkeiten mit begründeten Ausnahmen (ggf. Statusanfragen zum Umrichter, Handbetrieb, etc.)

### 2.3.3 Risiken

Keine bekannt.

### 2.3.4 Testhinweise

Die Referenzfahr ist aus jeder Position durchzuführen.

### 2.3.5 Grobschätzung des Aufwands

Mittel.

## 2.4 Sicherer Betrieb durch Lichtschranken und Not-Aus

| **Nr. / ID** | ANF_04 | **Nichttechnischer Titel** | | Sicherer Betrieb durch Lichtschranken und Not-Aus | | |
|---|---|---|---|---|---|---|
| **Quelle** | | | **Verweise** | | **Priorität** | muss |

### 2.4.1 Beschreibung

Mindestens vor der ersten Referenzfahrt wird der Fahrbetrieb durch manuelle Freigabe der Lichtschranken durch den Bediener am Schaltschrank geleistet.

Diese Funktion kann ausschließlich durch einen Überbrückungsschalter im Schaltschrank umgangen werden. Eine aktive Überbrückung wird in der GUI angezeigt.

Wird versucht die Achse einzuschalten ohne dass die Lichtschranken am Schaltschrank freigegeben wurden oder ein Not-Aus-Schalter betätigt ist, befindet sich der Frequenzumrichter im Fehlerzustand. Dieser wird in der GUI angezeigt und muss dort zurückgesetzt werden.

### 2.4.2 Wechselwirkungen

Jede Fahrbetriebsfunktion.

### 2.4.3 Risiken

Personenschaden!

### 2.4.4 Testhinweise

Ein sicherer Betrieb muss auch beim Auslösen der Lichtschranken bzw. Not-Aus-Schalter während einer Achsenbewegung gewährleistet sein.

### 2.4.5 Grobschätzung des Aufwands

Mittel.

## 2.5 Geschwindigkeits- und Positionsanzeige

| **Nr. / ID** | ANF_05 | **Nichttechnischer Titel** | Geschwindigkeits- und Positionsanzeige | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | hoch |

### 2.5.1 Beschreibung

Aktuelle Position relativ zum Lagemittelschalter des Systems und aktuelle Geschwindigkeit werden jederzeit in der GUI angezeigt. Zusätzlich wird in der GUI angezeigt ob die Zielposition erreicht wurde.

### 2.5.2 Wechselwirkungen

### 2.5.3 Risiken

Keine bekannt.

### 2.5.4 Testhinweise

### 2.5.5 Grobschätzung des Aufwands

Gering.

## 2.6 Erweiterung rotativer Teststand

| **Nr. / ID** | ANF_06 | **Nichttechnischer Titel** | Erweiterung rotativer Teststand | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | hoch |

### 2.6.1 Beschreibung

Die GUI umfasst eine leere Seite zur Steuerung des rotativen Stands.

Die GUI erkennt eigenständig die Art des angeschlossenen Teststands und schaltet eigenständig zugehörige Bedienelemente frei.

### 2.6.2 Wechselwirkungen

### 2.6.3 Risiken

Keine bekannt.

### 2.6.4 Testhinweise

Ein Test mit Anschluss des rotativen Teststandes wird nicht realisierbar sein.

### 2.6.5 Grobschätzung des Aufwands

Gering.

## 2.7 Anzeigesprache Englisch

| Nr. / ID | ANF_07 | Nichttechnischer Titel | | Anzeigesprache Englisch | | |
|---|---|---|---|---|---|---|
| Quelle | | | Verweise | | Priorität | muss |

### 2.7.1 Beschreibung

GUI ist in Englisch ausgeführt.

### 2.7.2 Wechselwirkungen

### 2.7.3 Risiken

Keine bekannt.

### 2.7.4 Testhinweise

### 2.7.5 Grobschätzung des Aufwands

Mittel.

## 2.8 Handbetrieb ohne Referenzierung

| Nr. / ID | ANF_08 | Nichttechnischer Titel | | Handbetrieb ohne Referenzierung | | |
|---|---|---|---|---|---|---|
| Quelle | | | Verweise | | Priorität | niedrig |

### 2.8.1 Beschreibung

Ohne die Referenzfahrt durchzuführen ist es möglich den Teststand im Handbetrieb zu verfahren. Dieser sieht eine relative Positionierung in Bezug auf die derzeitige Position vor. Einzustellen ist die relative Position. Geschwindigkeit und Beschleunigung (positiv und negativ) werden auf zu definierende, ungefährliche Werte vorgewählt.

### 2.8.2 Wechselwirkungen

### 2.8.3 Risiken

Achse ist nicht referenziert. Die Position des Magnetsensors und die des Motorencoders weichen voneinander ab. Fehlerhafte Messvergleichswerte.

Keine Softwarelageendschalter vorhanden.

### 2.8.4 Testhinweise

Ein umschalten in den absoluten Positioniermodus darf nicht möglich sein.

### 2.8.5 Grobschätzung des Aufwands

Mittel.

## 2.9 Handbetrieb mit Referenzierung

| **Nr. / ID** | ANF_09 | **Nichttechnischer Titel** | Handbetrieb mit Referenzierung | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | hoch |

### 2.9.1 Beschreibung

Im Handbetrieb mit Referenzierung ist es möglich, eine absolute sowie eine relative Positionierung durchzuführen.

Im absoluten Positioniermodus ist die absolute Position, die Geschwindigkeit, die Beschleunigung (positiv und negativ) einzustellen, im relativen nur die relative Position, die anderen Parameter werden vorgewählt.

### 2.9.2 Wechselwirkungen

### 2.9.3 Risiken

Keine bekannt.

### 2.9.4 Testhinweise

### 2.9.5 Grobschätzung des Aufwands

Gering.

## 2.10 Modus Reversierfahrt

| **Nr. / ID** | ANF_10 | **Nichttechnischer Titel** | Modus Reversierfahrt | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.10.1 Beschreibung

Der Benutzer kann in der GUI eine Reversierfahrt anfordern mit folgenden Parametern:

- Geschwindigkeit Hinfahrt
- Geschwindigkeit Rückfahrt
- Beschleunigung (positiv und negativ) Hinfahrt
- Beschleunigung (positiv und negativ) Rückfahrt
- Startposition
- Endposition
- Stillstandsdauer nach Hinfahrt
- Stillstandsdauer nach Rückfahrt
- Anzahl der Reversierfahrten oder Fahren ohne Begrenzung der Testdauer

Die GUI überprüft, ob die angegebenen Geschwindigkeitswerte innerhalb der zu fahrenden Strecke bei der angegebenen Beschleunigung erreicht werden können und weist den Benutzer im Falle nicht erreichbarer Geschwindigkeiten darauf hin.

Komplette, gültige Parametrierungen können durch den Benutzer abgespeichert und wieder geladen werden. Der Benutzer kann den Speicherort der Parametrierung angeben. Lässt der Benutzer diese Angaben frei, werden Defaults verwendet. Das Programm erkennt bereits vorhandene Dateinamen und vermeidet überschreiben durch inkrementieren eines Zählerwerts im Namen.

Am Ende der Testfahrt wird ein Messprotokoll gemäß ANF_12 gespeichert.

### 2.10.2 Wechselwirkungen

### 2.10.3 Risiken

Keine bekannt.

### 2.10.4 Testhinweise

### 2.10.5 Grobschätzung des Aufwands

Hoch.

## 2.11 Grenzwerte Geschwindigkeit, Beschleunigung und Position

| **Nr. / ID** | ANF_11 | **Nichttechnischer Titel** | | Grenzwerte Geschwindigkeit, Beschleunigung und Position | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.11.1 Beschreibung

Die GUI erlaubt keine Eingabe von Geschwindigkeits- und Beschleunigungswerten oberhalb festzulegender Grenzwerte. Eine absolute Positionsangabe (Handbetrieb absolute Positionierung und Modus Reversierfahrt) außerhalb der Grenzen ist nicht möglich.

Das entsprechende Feld wird rot gefärbt und der Grenzwert erscheint.

### 2.11.2 Wechselwirkungen

### 2.11.3 Risiken

Keine bekannt.

### 2.11.4 Testhinweise

### 2.11.5 Grobschätzung des Aufwands

Mittel.

## 2.12 Messprotokoll als csv

| **Nr. / ID** | ANF_12 | **Nichttechnischer Titel** | | Messprotokoll als csv | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.12.1 Beschreibung

Am Ende jeder Messfahrt wird ein Messprotokoll in einem noch zu definierenden csv-Format abgespeichert.

Das csv-File enthält einen Header, umfassend:

- Auflösung des Referenzencoders
- Auflösung des DUT
- Datum
- Bemerkungen des Benutzers (Ein Eingabefeld in der GUI muss vorhanden sein)
- Parameter des Tests

Das csv-File ist in Matlab importierbar.

Der Name und der Speicherort der csv-Datei wird durch den Benutzer vorgegeben. Lässt der Benutzer diese Angaben frei, werden Defaults verwendet. Das Programm erkennt bereits vorhandene Dateinamen und vermeidet überschreiben durch inkrementieren eines Zählerwerts im Namen.

### 2.12.2 Wechselwirkungen

### 2.12.3 Risiken

Keine bekannt.

### 2.12.4 Testhinweise

Ein Importskript zum Test wird mit Bergmann erstellt.

### 2.12.5 Grobschätzung des Aufwands

Hoch.

## 2.13 Festlegen der Auflösungen

| **Nr. / ID** | ANF_13 | **Nichttechnischer Titel** | Festlegen der Auflösungen | | |
|---|---|---|---|---|---|
| **Quelle** | | **Verweise** | | **Priorität** | muss |

### 2.13.1 Beschreibung

Der Benutzer kann die Auflösungen von Referenzencoder und DUT eingeben.

### 2.13.2 Wechselwirkungen

### 2.13.3 Risiken

Keine bekannt.

### 2.13.4 Testhinweise

Die Auflösung der DUT wird im Messprotokoll Header übergeben.

### 2.13.5 Grobschätzung des Aufwands

Gering.

## 2.14 Testprogramme

| Nr. / ID | ANF_14 | Nichttechnischer Titel | Testprogramme | | |
|---|---|---|---|---|---|
| Quelle | | Verweise | | Priorität | mittel |

### 2.14.1 Beschreibung

Der Benutzer kann eine Liste, bestehend aus bereits abgespeicherten Reversierfahrten erstellen. Diese Liste wird im Anschluss abgefahren.

### 2.14.2 Wechselwirkungen

### 2.14.3 Risiken

Keine bekannt.

### 2.14.4 Testhinweise

### 2.14.5 Grobschätzung des Aufwands

Hoch.

## 2.15 Softwareendlagen

| Nr. / ID | ANF_15 | Nichttechnischer Titel | Softwareendlagen | | |
|---|---|---|---|---|---|
| Quelle | | Verweise | | Priorität | mittel |

### 2.15.1 Beschreibung

Die Softwareendlagenfunktion des Umrichters verhindert ein Überfahren der Lageendschalter bis zum mechanischen Anschlag.

Die Softwareendlagen sind mit einem Bedienelement veränderbar, da der Bremsweg je nach Last getestet werden muss.

### 2.15.2 Wechselwirkungen

### 2.15.3 Risiken

Keine bekannt.

### 2.15.4 Testhinweise

Bei maximaler Geschwindigkeit und Last, darf der mechanische Anschlag nicht erreicht werden.

### 2.15.5 Grobschätzung des Aufwands

Mittel