Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Jan Sachse

Online-Monitoring - Zustandsautomat der Sicherheitstopologie für kooperative Industrieroboter

Fakultät Technik und Informatik *Faculty of Engineering and Computer Science*

Jan Sachse

Online-Monitoring - Zustandsautomat der Sicherheitstopologie für kooperative Industrieroboter

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

Betreuender Prüfer: Prof. Dr.-Ing. Thomas Frischgesell
Zweitgutachter: Dipl.-Ing. Carolina Bohnert

Abgegeben am 15. Juli 2016

Jan Sachse

Thema der Bachelorthesis

Online-Monitoring - Zustandsautomat der Sicherheitstopologie für kooperative Industrieroboter

Stichworte

Steuerung, KUKA, dSPACE, Industrieroboter, Sicherheitstopologie, Zustandsautomat

Kurzzusammenfassung

Das Ziel dieser Bachelorarbeit ist die Entwicklung eines Zustandsautomaten für KUKA KR C4 compact Industrierobotersteuerungen mit Simulink Stateflow. Im Rahmen dieser Bachelorarbeit war es daher notwendig, die sicherheitsrelevanten Signale über Schnittstellen auszulesen, diese elektrisch zu koppeln und mit einer Visualisierung die Zustände und Schaltbedingungen sichtbar zu machen.

Jan Sachse

Title of the paper

Online-Monitoring - State machine of security topology for cooperative industrial robots

Keywords

control unit, state machine, KUKA, dSPACE, robots, security topology

Abstract

The aim of this thesis is to create a state machine for KUKA KR C4 compact industrial robot controllers with Simulink Stateflow. For this task it was necessary to read the security signals via interfaces, to connect these electrically and to realise a visualization of the states and switching conditions.

Inhaltsverzeichnis

Abkürzungsverzeichnis

C	Programmiersprache
DVI	Digital Visual Interface; Monitoranschluss
EN	Europäische Norm
EtherCAT	Ethernet for Control Automation Technology; Feldbussystem auf Ethernet-Basis
GND	Ground; englisch für Masse
GNDA	Ground analog; Massefläche mit getrenntem Potential
GUI	Graphical User Interface; Bedienoberfläche der Visualisierung
HIL	Hardware In The Loop; Testsystem für Hardware
IEC	International Electrotechnical Commission; Norm
I/O	Input/Output; Digitale Ports, sind als Eingang oder Ausgang verwendbar
KRL	KUKA Robot Language; KUKAs eigene Programmiersprache
LED	Light Emitting Diode; Leuchtdiode
PCI	Peripheral Component Interconnect; Schnittstellentyp
PLC	Programmable Logic Controller; englisch für SPS
RCP	Rapid Control Prototyping; System für Regelungs-und Steuerungsentwicklung
SAK-Fahrt	Satzkoinzidenz-Fahrt; Koordinatenabgleich von Roboter & Bewegung
SmartPAD	KUKA Programmierhandgerät
SPS	Speicherprogrammierbare Steuerung
VEOS	Computer-basierte Simulationsplattform von dSPACE
VCC	Voltage at the Common Collector; positive Versorgungsspannung

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

In diesem Kapitel wird die Aufgabenstellung und Motivation vorgestellt und ein kurzer Überblick über den Aufbau der Arbeit wiedergegeben.

1.1 Aufgabenstellung

Im Zentrum für industrielle Robotik werden KUKA Agilus KR6 Roboter mit KR C4 Steuerungen verwendet. Der Arbeitsraum für beide Roboter wird über Not-Aus Schalter, Lichtgitter, Trittschutzmatte und ein Safety Eye gesichert. Die Simulation und Programmierung erfolgt über die entsprechenden Handbediengeräte sowie angeschlossene PCs. Für beide Roboter soll im Rahmen dieser Arbeit ein echtzeitfähiger Zustandsautomat erstellt werden, der die wichtigsten Zustände und Schalterstellungen der Robotersteuerung KR C4 emuliert, nach außen sichtbar und teilweise steuerbar macht. Dabei soll zum einen der Einzelbetrieb, der duale Betrieb und auch der kooperative eruiert werden. Die zur Verfügung stehenden elektrischen Schnittstellen sind mit dem Echtzeitboard signalkompatibel zu verdrahten. Mit Hilfe von Simulink soll eine Software für ein Echtzeitboard von dSpace entwickelt werden, die die Schnittstellen der Roboter ausliest, die entsprechenden Zustandsänderungen wahrnimmt und über eine Bedienoberfläche auch visualisiert. Dem Bediener werden damit die Einzelzustände und Schaltbedingungen und damit entsprechende Handlungsoptionen offensichtlich. Nicht sicherheitsrelevante Zustände lassen sich gegebenenfalls auch über die Bedienoberfläche schalten bzw. Quittierungsmeldungen auslösen. Die Dokumentation beschreibt die Zustände und die Schaltbedingungen bedienergerecht.

1.2 Motivation

In den Zeiten von Industrie 4.0 gewinnen Industrieroboter vermehrt an Bedeutung. Die Produktionen sollen an Flexibilität zunehmen und der Einsatz in direkter Zusammenarbeit zwischen Roboter und Werksarbeiter soll ermöglicht werden. Ein starker Fokus wird hierbei oft auf die Arbeitssicherheit gelegt und muss bei der Verwendung von automatisierten Anlagen und Robotern berücksichtigt werden. Das hat nicht selten eine entscheidende Auswirkung

auf mögliche Anschaffungen und Neuentwicklungen. Starre Schutzzäune haben hier eine eher störende Funktion, so dass sensorische Sicherheitseinrichtungen und kollaborierende Industrieroboter immer mehr an Bedeutung erlangen. In einigen Bereichen werden kollaborierende Roboter noch mit Skepsis betrachtet. Der Grund dafür sind Szenarien, die weiterhin ein potentielles Risiko mit sich bringen. Ein Beispiel hierfür sind Arbeiten mit scharfkantigen oder spitzen Werkstücken, bei denen Kollisionen mit Arbeitern Verletzungen hervorrufen können. Viele Industrieunternehmen greifen daher auf sensorische Sicherheitseinrichtungen wie Lichtschranken und optische Bilderkennungseinrichtungen zurück [22]. Zu Schulungszwecken hat die Hochschule für Angewandte Wissenschaften Hamburg zwei KUKA Agilus KR6 R900 sixx Industrieroboter mit KR C4 Steuerung angeschafft. Die Industrieroboter verfügen über Sicherheitszustände, die die Roboter bei entsprechenden externen Signalen oder internen Sicherheitsauswertungen einnehmen. Durch den Zustandsautomaten und der Visualisierung lässt sich für den Bediener oder in der Umgebung befindende Personen gut erkennen, in welchem Zustand der Roboter sich befindet und vereinfacht somit sowohl die Inbetriebnahme nach einem Not-Aus, einem Systemfehler oder anderen Ursachen, als auch das Verständnis und die Übersichtlichkeit des Systems.

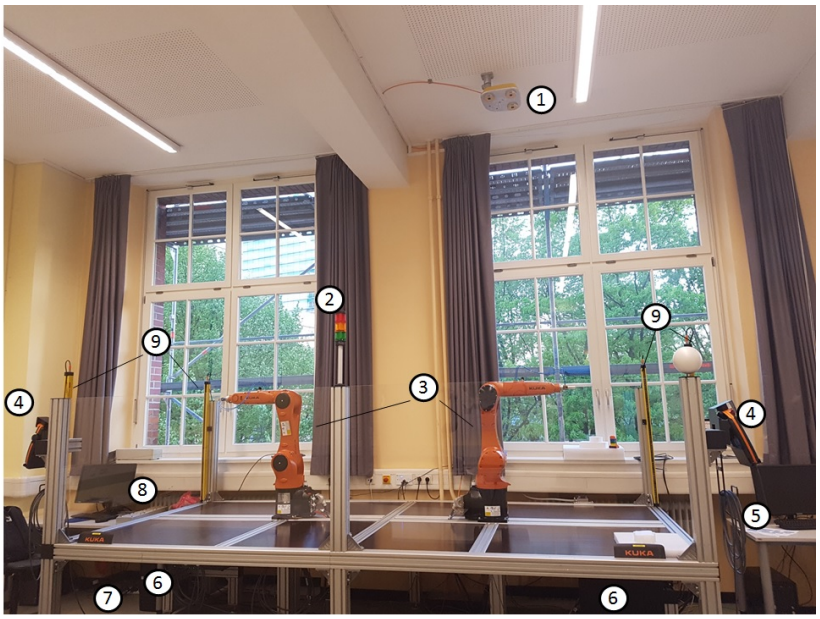
1.3 Aufbau der Arbeit

Die Bachelorarbeit ist in 5 Hauptkapitel gegliedert. Das Kapitel 1 gibt eine kurze Einführung in die Bachelorarbeit. Darin werden die Aufgabenstellung und die Motivation beschrieben. Zum Verständnis wird am Anfang in Kapitel 2 eine Übersicht des aufgebauten Systems und die einzelnen Komponenten vorgestellt. Im darauffolgenden Kapitel 3 werden die Anforderungen erläutert und mögliche Lösungen entworfen. Die tatsächliche Realisierung und Umsetzung des Zustandsautomaten und der damit verbundenen Visualisierung bilden das Kapitel 4. Hier werden ebenfalls wichtige Details für die umgesetzte Hardwareverbindung und Inbetriebnahme erläutert und getestet. Das 5. Kapitel dieser Arbeit fasst alle Ergebnisse zusammen und zeigt einen Ausblick auf das weitere Vorgehen oder mögliche Optimierungen.

2 Grundlagen

In dem Kapitel 2.1 werden die einzelnen Komponenten des Systems vorgestellt und Zusammenhänge erklärt. Die verwendete Software und nötigen Versionen sind in Kapitel 2.2 erläutert. Der Einfluss der Betriebsmodi und die einzelnen Zustände für den Zustandsautomaten werden in Kapitel 2.3 kurz dargelegt.

2.1 Systemaufbau und Systemkomponenten



1. PILZ Safety Eye
2. Kontrollampel
3. KUKA Agilus KR6
4. smartPAD
5. Nothalt- & Quittierungsschalter
6. KUKA KR C4 compact
7. Trittschutzmatte
8. dSPACE Station
9. PILZ Lichtschranken

Abbildung 2.1: Aufbau der Roboterzelle

Um einen Überblick über alle Komponenten zu bekommen, ist der komplette Aufbau in Abbildung 2.1 zu sehen. Aus Abbildung 2.2 sind die Verbindungen der einzelnen Komponenten zu entnehmen. Dabei sind alle Geräte mit der KR C4 compact Steuerung über Schnittstellen verbunden. Für die Kommunikation zwischen Echtzeitboard und Steuerung wird eine elektrische Schaltung entwickelt, die die Spannungspegel anpasst, um eine Kommunikation zu

ermöglichen. Das Echtzeitboard ist über eine PCI-Karte mit einem Computer verbunden, auf dem das Programm ControlDesk für die Visualisierung des Zustandsautomaten installiert ist. Der Zustandsautomat wird mit Simulink Stateflow programmiert, in ein C-Programm übersetzt und auf das Echtzeitboard gespielt.

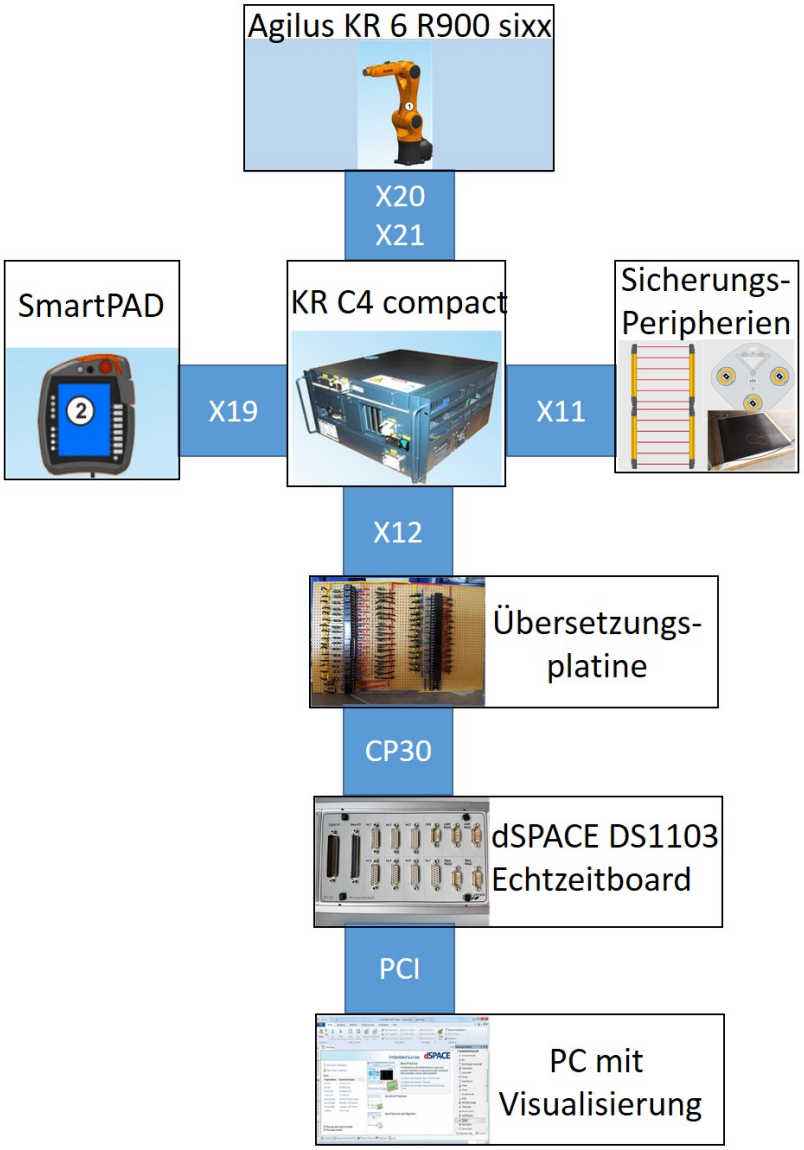


Abbildung 2.2: Zusammenhänge der Komponenten

2.1.1 Der KUKA Agilus KR6 R900 sixx Roboter

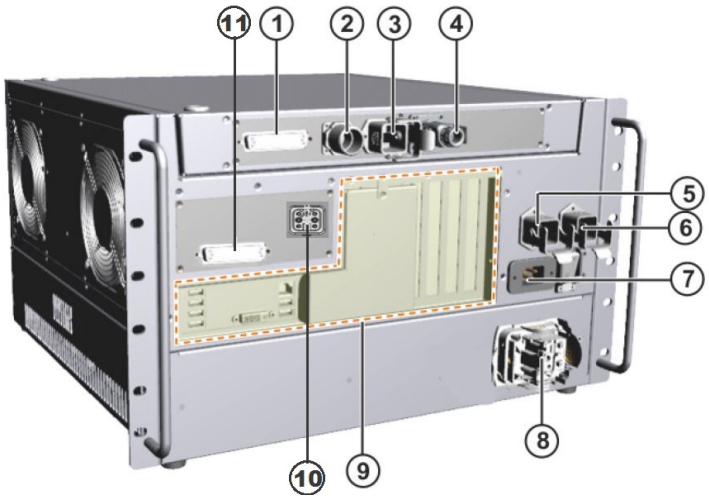
Der KUKA Agilus KR6 R900 sixx ist ein Industrieroboter, mit sechs Kilo Traglast. Damit gehört er zu der Klasse der Kleinrobotik. Er verfügt über 6 Achsen und die Hauptaufgabengebiete bestehen aus Montagezwecken, Handling, Bestückung, Beladungs- und Entladungsaufgaben [8, S.17]. Der Roboter ist als Wand-, Decken- und Boden-Variante verfügbar. Die verwendeten Roboter gehören zu der Boden-Variante.

Abbildung 2.3: Aufgebaute Agilus KR6 Roboter an der HAW Hamburg

2.1.2 Die KUKA KR C4 compact Steuerung

Angesteuert werden die KUKA Roboter Agilus KR6 mit einer KUKA KR C4 Steuerung, die über das SmartPAD bedient wird. Die Steuerung KUKA KR C4 compact ist die kleinste Variante der KR C4 Steuerungen der Firma KUKA, die in verschiedenen Ausführungen und Kundenvarianten erhältlich ist. Die Steuerung KR C4 compact verfügt über eine Reihe von Schnittstellen. Eine Übersicht der Schnittstellen ist in Abbildung 2.4 zu entnehmen. Die Schnittstellen haben unterschiedliche Aufgaben, wie Netzspannungsversorgung, Ansteuerung des Roboters oder Gewährleistung der Sicherheit. Die Datenübertragung kann dabei diskret, anlog oder über Bussysteme wie Ethernet erfolgen. Ebenso sind PC-Schnittstellen vorhanden. Diese beinhalten USB, LAN, DVI-Anschlüsse und PCI-Kartenslots. Für den Zustandsautomaten sind nur die für die Sicherheit übertragbaren Daten relevant. Die verwendete Kundenvariante ist die „KUKA KR C4 compact RP DC1 X11 X12 X55“. Die Variante

verfügt über eine I/O-Schnittstelle, an die für Diagnosezwecke Signale ausgelesen werden können. Die Schnittstelle hat die Bezeichnung X12. Intern ist diese mit einem EtherCAT Master verdrahtet, der über den „KUKA Extension Bus“ mit dem Mainboard verbunden ist. Die Konfiguration des Masters erfolgt mit dem Programm „WorkVisual“ [7, S.56].



1. X11 Sicherheits-Schnittstelle
2. X19 Anschluss SmartPAD
3. X65 Extention-Schnittstelle
4. X69 Service-Schnittstelle
5. X21 Manipulator-Schnittstelle
6. X66 Ethernet-Schnittstelle
7. X1 Netz-Anschluss
8. X20 Motorstecker
9. PC-Schnittstellen
10. X55 Externe Spannungsversorgung
11. X12 Digitale I/O-Schnittstelle

Abbildung 2.4: Schnittstellen der Steuerung [9, S.16] & [19, S.10]

2.1.3 Sicherheitsüberwachungskomponenten

Die aufgebaute Roboterzelle ist teilweise mit einer durchsichtigen Scheibe abgeschirmt und enthält keine Schutztür. Die beiden offenen Bereiche der Roboterzelle sollen mit Lichtschranken des Modells „Tapeswitch Guardscan“ überwacht und über das Signal „Bedienerschutz“ abgefragt werden. Ebenfalls über die diskrete Schnittstelle angebunden sind der Not-Halt- und der Quittierungsschalter. Zusätzlich ist eine Trittschutzmatte von der SSZ-GmbH vorhanden, die sicher stellt, dass der Bediener sich außerhalb der Gefahrenzone befindet. Außerdem ist auch eine Überwachung über Safety Eye installiert. Die Safety Eye Vorrichtung besteht aus der Sensoreinheit „PILZ PSEN se SU AM3 65“ für die Bilderkennung, der Auswerteeinheit „PILZ PSEN se AU AM3“, die zur Auswertung für die Arbeitsraumüberwachung des Safety Eyes dient und das programmierbare Steuerungssystem „PILZ PSS 3047-3 ETH-2 SE“, das bereits mit vorinstallierten Anwenderprogrammen und diverser Schnittstellen ausgestattet ist [21, S.88]. Sollte eine Verletzung des Arbeitsbereiches erkannt werden, wird dieses über die Ampel dargestellt.

Aktuell sind die Sicherheitsperipherien nicht mit der Steuerung des Roboters verbunden, werden aber in der Entwicklung des Zustandsautomaten berücksichtigt, da sie parallel zu dieser Arbeit in Betrieb genommen werden sollen.

2.1.4 Das DS1103 PPC Controller Board von dSPACE

Das für den Aufbau verwendete Echtzeitboard ist das DS1103 PPC Controller Board der Firma dSPACE. Für die vereinfachte Verbindung des Boards mit Anwendungsgeräten ist es mit „Connector LED Combi Panel“ verbunden. Die Programmierung erfolgt dabei direkt in Simulink über die Blockdiagramme. Aus den Blöcken wird beim Aufspielen der Software direkt ein C-Code generiert.

Abbildung 2.5: Connector LED Combi Panel für Verbindung mit Echtzeitboard [4]

2.2 Softwarekomponenten

Für die Kompatibilität der Softwareversionen sind die Programme auf dem operativen System Windows 7 mit Service-Pack 1 Erweiterung installiert. Als dSPACE RCP and HIL Software wird das Release 2015A verwendet. Das ist in Verbindung mit MATLAB/Simulink der Version R2015a und ControlDesk der Version 5.4 lauffähig [3]. Durch die Installation wird die „Simulink Libary“ um die in Abbildung 2.6 angezeigten Blöcke erweitert. Sie ermöglichen die Kommunikation zu den Schnittstellen des Echtzeitboards. Die Einstellungen in MATLAB, wie beispielsweise die Approximationsmethode und der Generator des C-Codes, werden automatisch angepasst.

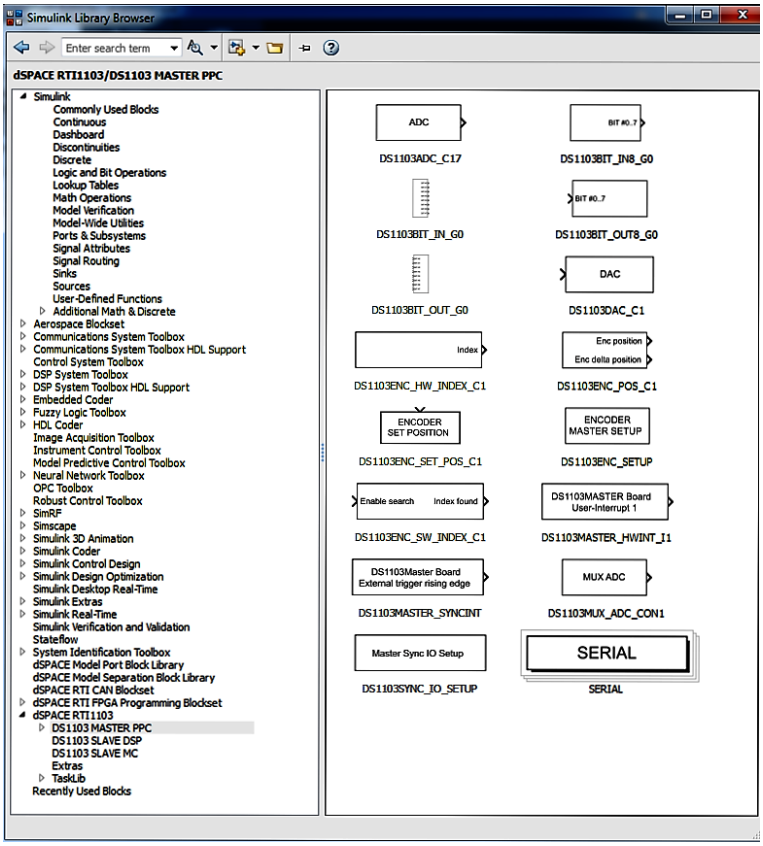


Abbildung 2.6: Ergänzung der Simulink Libary mit dSPACE-Bausteinen

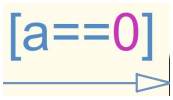
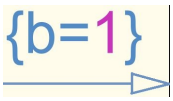
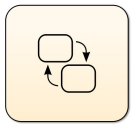
2.2.1 Simulink Stateflow

Um den Zustandsautomaten lesen zu können, erfolgt an dieser Stelle eine kleine Einweisung in die Systematik und Funktionsweise von Simulink Stateflow. Die Zustände werden in einem „Chart“ dargestellt, das einem abgerundeten Viereck entspricht. Oben links befindet sich der Name des Zustandes, der eindeutig in der Chart-Umgebung vergeben sein muss. Ein Chart kann mehrere Charts enthalten, wodurch es auch möglich ist, einen „Zustand im Zustand“ zu realisieren. Außerdem ist es möglich, dem Chart Aktionen zuzuordnen, die auch innerhalb des Charts notiert werden. Durch Schlüsselwörter wird angegeben, ob die Aktion beim Eintreten, Verlassen oder während des Zustandes ausgeführt wird. Die Bedeutung der Schlüsselwörter ist Tabelle 2.1 zu entnehmen. Die Charts können exklusive oder parallele Zustände sein. Welche Art von Zustand vorliegt, ist am Rahmen erkennbar. Der Wechsel zwischen den Zuständen geschieht über Transitionen, die an Bedingungen geknüpft sein können oder ebenfalls Aktionen auslösen. Eine Bedingung wird in eckigen und eine Aktion in geschweiften Klammern dargestellt. Eine Start-Transition gibt an, in welchem Zustand sich der Automat bei der Inbetriebnahme befindet.

Tabelle 2.1: Schlüsselwörter

Schlüsselwort	Funktion
entry	Aktion beim Eintreten
during	Aktion, während der Zustand aktiv ist
du	Aktion, während der Zustand aktiv ist
exit	Aktion wird ausgeführt, wenn der Zustand verlassen wird

Tabelle 2.2: Stateflow-Legende

Symbol	Funktion
	Transition mit Bedingung
	Start-Transition
	Transition mit Aktion
	Exklusives Chart
	Paralleles Chart
	Subchart
	Chartumgebung

2.2.2 Verbinden über ControlDesk

Die Visualisierung des Zustandsautomaten erfolgt über das Programm ControlDesk 5.4. In dem Programm können virtuelle Schalter und Anzeigen erstellt und in Layouts gespeichert werden. Bei jedem der Schalter und jeder Anzeige sind spezielle Anpassungen möglich. Dadurch lassen sie sich nach eigenen Wünschen gestalten und kreieren. Über den Button „Go-online“ wird eine Verbindung mit dem Echtzeitboard von dSPACE hergestellt. Das Simulink-Modell muss zuvor in Simulink mit dem Befehl „Build“ auf das Board übertragen worden sein. Die Übertragung ist nicht möglich, während ControlDesk auf das Echtzeitboard Zugriff hat. Dazu muss die Verbindung mit dem Befehl „Go-offline“ wieder getrennt werden. Sollte ein neues Programm aufgespielt worden sein, gibt ControlDesk eine Warnung aus, die bestätigt werden muss, um die Variablenliste zu aktualisieren und erneut zu verknüpfen. Wenn das Echtzeitboard nicht verbunden ist, kann das entwickelte Layout nicht getestet werden. Damit eine Simulation auch unabhängig von der Hardware durchführbar ist, wurde der Offline-Simulator „VEOS“ entwickelt [5].

2.3 Einfluss der Betriebsmodi und Zustände

In dem System werden zwei KUKA Agilus KR6 R900 sixx mit jeweils einer zugehörigen KR C4 compact Steuerung betrieben. Die in Abbildung 2.1 aufgelisteten Sicherheitsperipherien sind nicht mit den Robotern verbunden, werden aber in der Zukunft an die Steuerungen angeschlossen und daher im Aufbau berücksichtigt. Sollte der mit den Sicherheitsperipherien überwachte Raum durch einen Sicherheitsverstoß verletzt werden, wird in beiden Robotersteuerungen ein Sicherheitshalt ausgelöst. Daher ist für den Zustandsautomaten nicht relevant, ob die Roboter einzeln, dual oder kooperativ betrieben werden. Innerhalb der Steuerung kann eine interne Überwachung der Roboterposition konfiguriert werden. Dabei muss allerdings berücksichtigt werden, ob eine Überschneidung der Räume, in denen die Manipulatoren der Roboter verfahren, erlaubt ist. Eine Überschneidung ist für einen kooperativen Betrieb notwendig. So können beispielsweise Gegenstände übergeben werden oder eine gemeinsame Arbeit stattfinden. Beim dualen Betrieb sollten die Räume voneinander getrennt sein. Dadurch wird eine Kollision bei parallelen Arbeiten verhindert. Eine sinnbildliche Darstellung ist in Abbildung 2.7 durch ein mit KUKA Sim Pro erstellten Modells veranschaulicht.

Der Zustand, in dem der Roboter verfahren werden kann und kein sicherheitsrelevanter Auslöser den Roboter stoppt, wird als Betrieb bezeichnet. Wird der Betrieb des Roboters gestört, kann dieser abhängig vom Auslöser der Störung in verschiedene Zustände wechseln, durch die ein Sicherheitsstopp ausgelöst wird. Die Zustände können ein Stopp der Kategorie 0 bis 2 entsprechend der EN 602041:2006 sein. Bei STOP0 findet die Bremsung bahnnahe

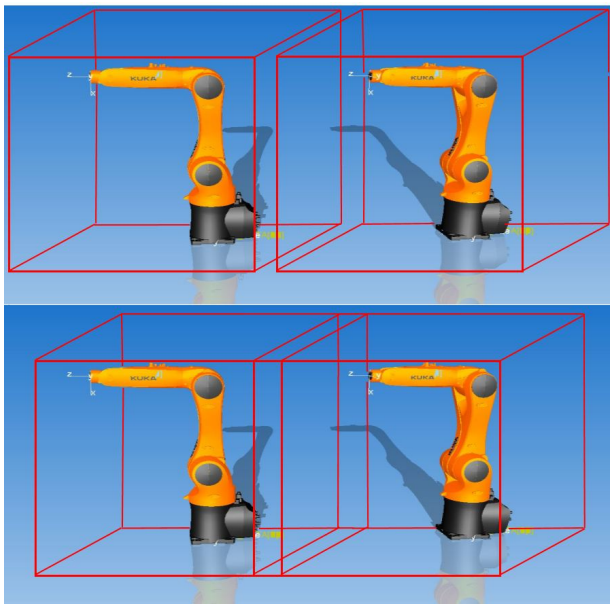


Abbildung 2.7: Getrennte Arbeitsräume (oben) und überschneidende Arbeitsräume (unten)

statt, während die Bremsung in STOP1 und STOP2 bahntreu sind. Ein weiterer wesentlicher Unterschied ist, dass in STOP0 die Antriebe des Roboters sofort abgeschaltet werden. Dagegen bleiben sie bei STOP1 noch eine begrenzte Zeit eingeschaltet, um die bahntreue Bremsung zu ermöglichen und bei STOP2 weiterhin aktiv. Die Dauer bis zur Abschaltung in STOP1 ist abhängig von der Betriebsart. Der Roboter kann in einem Handbetrieb mit reduzierter Geschwindigkeit in gewünschte Positionen gefahren werden. Die Betriebsart wird T1 genannt und ist für das „Teachen“ von Bewegungspunkten vorgesehen. In T1 werden die Antriebe bei einem STOP1 abgeschaltet, sobald der Stillstand erfolgt. Anderenfalls schalteten sie spätestens nach 680 Millisekunden ab. In den anderen Betriebsarten erfolgt die Abschaltung immer nach 1,5 Sekunden. Der Grund für den Zeitunterschied ist, dass der Roboter in T1 nur mit der reduzierten Geschwindigkeit bewegt wird und dadurch ein kürzerer Bremsweg nötig ist. Die anderen Betriebsarten sind der Testbetrieb T2, der ohne reduzierte Geschwindigkeit durchgeführt wird. Der Automatikbetrieb AUT, bei dem das angewählte Programm mit eingestellten Bewegungspunkten und Trajektorien abgefahren wird und der Automatikbetrieb AUT EXT, in dem eine externe übergeordnete Steuerung den Roboter ansteuert. Andere mögliche Zustände des Roboters sind die Stoppzustände, die von einer Sicherheitssteuerung durchgeführt werden. Sie werden im Folgenden als Sicherheitshalt0, Sicherheitshalt1 und Sicherheitshalt2 bezeichnet. Äquivalent zum STOP0 werden bei einem Sicherheitshalt0 auch die Antriebe sofort abgeschaltet. Ebenso stoppt sofort die Spannungsversorgung der Bremsen. Der Vorgang wird von der Sicherheitssteuerung sowohl durchgeführt als auch überwacht. Bei dem Sicherheitshalt1 und Sicherheitshalt2 dagegen, wird der Vorgang von der Sicherheitssteuerung durchgeführt, aber vom nicht-sicherheitsrelevanten

Teil der Steuerung überwacht. Bei dem Sicherheitshalt2 bleiben die Spannungsversorgung der Bremsen und Antriebe aktiv, während im Sicherheitshalt1 beides nach Erreichen des Stillstandes abgeschaltet wird. Diese Zustände stoppen unabhängig von allen vier Betriebsarten gleich [9, S.29]. In welchem Zustand der Roboter sich befindet und welche Ursache der Grund für einen Zustandswechsel ist, soll mit dem Zustandsautomaten verdeutlicht werden. Beim Auslösen eines sicheren Betriebshaltes wird ein STOP0 ausgelöst und anschließend ein Stillstand aller Achsen kontrolliert.

3 Design

In diesem Kapitel werden Ideen zur Lösungsfindung beschrieben und bewertet. Dazu werden zuerst im Abschnitt 3.1 die Anforderungen definiert. Anschließend wird in Abschnitt 3.2 die Möglichkeiten für eine Abfrage und Verbindung der Schnittstellen dargelegt und im Abschnitt 3.3 die Möglichkeiten für einen Zustandsautomaten bewertet.

3.1 Modularisierung der Aufgabe

Für die Entwicklung des Zustandsautomaten werden die Anforderungen in drei Aufgabenkategorien geordnet. Erstens die Verbindung der Echtzeitboards mit der Robotersteuerung, um sicherheitsrelevante Signale auswerten zu können. Zweitens die Erstellung des Zustandsautomaten und drittens die Visualisierung des Zustandsautomaten. Für jede Kategorie werden die Anforderungen einzeln betrachtet.

- **Verbindung Echtzeitboard-Robotersteuerung:** Die sicherheitsrelevanten Signale müssen auf Schnittstellen verfügbar sein oder auf einer Schnittstelle zur Verfügung gestellt werden. Für die Verbindung muss diese elektrisch kompatibel sein oder gegebenenfalls eine Übersetzung entwickelt werden. Die auszulesenden Signale müssen echtzeitfähig übertragen und dabei galvanisch getrennt abgefragt werden. Das Senden von Signalen in die Steuerung sollte gesperrt werden können, um ungewolltes Ändern von Signalen zu vermeiden, da das ein Sicherheitsrisiko für den Anwender sein könnte. Eine Platine zur Übersetzung der Signale muss vor Schäden durch äußere Einflüsse, mit zum Beispiel einer Hülle, geschützt werden. Das Anschließen von Kabeln muss eindeutig sein, um fehlerhaftes Anschließen vorzubeugen. Dazu sind Anschlüsse vorzusehen, die nur in einer fest vorgegebenen Position verbunden werden können oder genau gekennzeichnet sind.
- **Zustandsautomat:** Der Zustandsautomat soll übersichtlich sein. Bei zu hoher Komplexität muss dieser auf einzelne Module heruntergebrochen werden, um anschaulich zu bleiben. Dabei sollte sich auf die wichtigsten Zustände beschränkt, aber alle sicherheitsrelevanten Signale berücksichtigt werden. Zustände, die keine sicherheitsrelevante Bedeutung haben, können vernachlässigt werden. Ein Beispiel hierfür ist der Zustand des Roboter-Interpreters, der auch über das SmartPAD ausgelesen werden

kann. Der Zustandsautomat muss simulierbar sein, um die Funktionen auch testen zu können, wenn die Robotersteuerung nicht angeschlossen ist. Sollten Signale relevant, aber nicht auslesbar sein, müssen sie immer simuliert werden können. Der Zustandsautomat sollte erweiterbar sein, um in der Zukunft weitere Funktionen oder Signale ergänzen zu können.

- **Visualisierung:** Die Visualisierung sollte über ein einheitliches Layout für Simulation und GUI zur Schnittstellenabfrage verfügen. In der Simulation sollten die eingehenden Signale und die Bedingungen auch direkt am Automaten manipulierbar sein, um alle Funktionen zu testen. Es sind simulierte Bedienmöglichkeiten für die Signale vorzusehen, die nicht über die Schnittstelle ausgelesen werden können. Durch die Visualisierung soll der aktuelle Zustand und die Handlungsmöglichkeiten dem Bediener ersichtlich werden.

In der Anleitung der KR C4 compact existiert eine Tabelle, in der einige der möglichen Auslöser für Stopp-Reaktionen in der Abhängigkeit der Betriebsart dokumentiert sind [9, S.32]. Nach der Tabelle löst die Steuerung je nach Fehlerursache einen STOP0 oder STOP1 bei einem internen Fehler des nicht sicherheitsgerichteten Teils der Steuerung aus. Welche Ursachen hierbei welchem Stopp zugeordnet werden, wird in den Betriebsanleitungen jedoch nicht dokumentiert. Allerdings ist es möglich mit der Variable „$ERR“ Fehler abzufragen [16, S.37]. Es existiert für den Submit- und Roboter-Interpreter jeweils eine eigene Variable, die auch nur innerhalb des zugehörigen Interpreters auswertbar ist [16, S.37]. Die Variablen sind vom Typ „ERROR_T“. Der Datentyp enthält eine Liste an Informationen zu dem Fehler. Es ist aber nicht klar, ob es sich bei den Fehlern um die internen Fehler handelt, die eine Stopp-Reaktion auslösen. Da die Liste nicht über die I/O Schnittstelle verschickt werden kann, müssen die internen Fehler als Auslöser simuliert werden.

Einige Zustände sind auch von einer externen übergeordneten Steuerung über die X12 Schnittstelle auslösbar. Ein externes Auslösen für einen Sicherheitshalt ist bei der KR C4 compact, im Gegensatz zu der KR C4, aber nur für den sicheren Betriebshalt (STOP0) und den Sicherheitshalt2 und nicht für Sicherheitshalt1 möglich [13, S.47]. Jedoch setzen die Zustände Sicherheitshalt1 und Sicherheitshalt2 ein Signal auf dem Feldbus, das auf die X12 Schnittstelle gelegt und somit für die Emulation des Zustandsautomaten abgefragt werden kann [9, S.29].

3.2 Umsetzungsmöglichkeiten der Schnittstellenabfrage

Mögliche auslesbare Sicherheitsschnittstellen der KR C4 compact sind über die diskrete Schnittstelle X11 und die Ethernet-Schnittstelle X66 möglich. Die gleichzeitige Verwendung

der Schnittstellen ist nicht möglich [9, S.15]. Daher muss sich für eine Variante entschieden werden. Über die Ethernet Schnittstelle kann die „Profisafe“-Variante als Sicherheitskonfiguration abgefragt werden [17, S.63]. Hierfür wird aber eine externe SPS benötigt. Da im aktuellen Aufbau keine SPS vorhanden ist, wurde die diskrete Schnittstelle als Sicherheitskonfiguration gewählt. Dazu muss in WorkVisual für die KR C4 compact als lokale Sicherheitskonfiguration „Kommuniktionsschnittstelle X11“ ausgewählt werden. Unter anderem wird an der Schnittstelle ein „Bedienerschutz“ als Signal abgefragt [13, S.44]. Das Signal könnte beispielsweise eine geschlossene Schutztür sein. Im aktuellen Zustand sind die Sicherheitsperipherien noch nicht mit der Steuerung verbunden. Da es sich um sicherheitsrelevante Einrichtungen handelt, ist es sinnvoll, sie ebenfalls an dem Signal Bedienerschutz der X11 Schnittstelle einzubinden. Im Zustandsautomaten sollten sie daher als Auslöser des Bedienerschutzes berücksichtigt, jedoch als einzeln simulierbare Schalter implementiert werden.

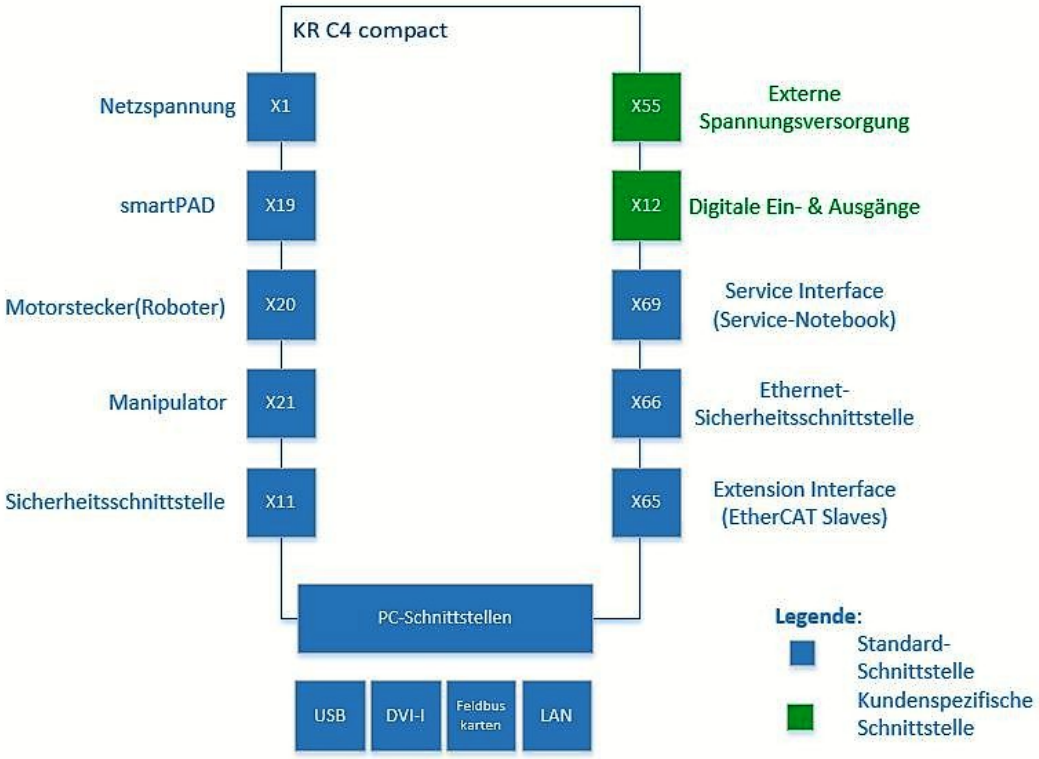


Abbildung 3.1: Schnittstellen

Es gibt verschiedene Möglichkeiten, Daten aus der KR C4 compact Steuerung zu erhalten. Eine Übersicht der Schnittstellen sind in Abbildung 3.1 zu finden. Die einfachste Möglichkeit ist es, die Daten der I/O-Schnittstelle X12 zu entnehmen, die nach kundenspezifischen Konfigurationen angepasst werden kann. Wenn die diskrete Schnittstelle X11 für die Sicherheitskonfiguration verwendet wird, sollten alle Sicherheitssignale an dieser abgegriffen werden

können. Nur die Auslöser innerhalb der Steuerung wären hier nicht abgreifbar. Daher wird der Fokus in Hinsicht zum Zustandsautomat auf die X12 Schnittstelle gelegt. Eine An- und Abwahl der X11 Schnittstelle im Automaten sollte dennoch für Erweiterungszwecke ermöglicht und als Verbindungsmöglichkeit mit berücksichtigt werden. Durch die doppelte Abfrage von Signalen über beide Schnittstellen ist der Betrieb des Automaten auch gewährleistet, wenn die X11 Schnittstelle nicht benutzt wird und eine alternative Sicherheitsoption, wie die Ethernet-Schnittstelle X66, gewählt würde. Ebenso wird durch die redundante Abfrage von Signalen sichergestellt, dass auf jeden Fall Auslöser für einen Zustandswechsel erkannt werden, so dass die Emulation der Robotersteuerung mit beiden Schnittstellen ein besseres Ergebnis erzielen sollte.

Die X12 ist eine I/O-Schnittstelle und besteht aus 16 digitalen Eingängen und 16 digitalen Ausgängen, die mit 24 Volt ein „High“-Potential und mit 0 Volt ein „Low“-Potential überträgt. Die Ausgänge können mit maximal 0,5 Ampere belastet werden. Für höhere Ströme sind vier zusätzliche Ausgänge vorhanden, die mit maximal 2 Ampere bestromt werden können [14, S.21]. In der Steuerung befindet sich ein Buskoppler der Firma Beckhoff. Der Buskoppler besteht aus vier Modulen. Dem EtherCAT Modul EK1100, der Eingangsklemme EL1809 und den Ausgangsklemmen EL2909 und EL2024. Damit die Schnittstelle verwendet werden kann, muss der Buskoppler mit Strom versorgt werden. Eine Stromversorgung ist sowohl mit einer externen Quelle über die X55 möglich, als auch mit der internen Spannungsversorgung durch Überbrücken von Kontakten. Die Überbrückung erfolgt genauso wie die externe Stromversorgung in der X55 Schnittstelle [19, S.21].

Bei der diskreten X11 Schnittstelle werden keine I/Os verwendet, sondern Testsignale zweikanalig ausgegeben und mit den dazugehörigen Eingängen verschaltet. Wenn ein Schalter eine Verbindung unterbricht, werden beide Kanäle unterbrochen. Das dient der Gewährleistung der Sicherheitsfunktionen. Sollte ein Kabelbruch auftreten, wäre nur ein Kanal unterbrochen. Ein solches Vorkommnis wird von der Steuerung ausgewertet und der Bediener mit einer Meldung auf dem SmartPAD informiert. Für den Zustandsautomaten ist es ausreichend, die Eingangssignale nur einkanalig abzufragen, weil dadurch die Hälfte der nötigen Eingänge auf dem dSPACE Echtzeitboard benutzt werden müssen. Wenn von dem Zustandsautomaten nur ein Kanal kontrolliert wird, hat der allerdings nicht die Möglichkeit, einen Kabelbruch zu erkennen und die Meldung des Kabelbruches kann nur von dem SmartPAD aus eingesehen werden. Der Zustand, den der Roboter im Falle eines Kabelbruches einnimmt, hängt von dem Signal ab, das durch den Bruch fehlerhaft auslöst wird. Um die Eingangssignale der X11 Schnittstelle zu prüfen, wäre es normalerweise möglich, die Spannung hinter dem Schalter abzugreifen und zu überprüfen, ob ein Signal anliegt. Sollte ein Signal mit 24 Volt anliegen, ist der Schalter geschlossen. Da die Nothalt-Einrichtung die Funktion eines Öffners hat, wäre das der Zustand, in dem der Schalter nicht betätigt worden ist. Jedoch wird der Stromkreis nicht mit einer durchgehenden anliegenden Spannung versorgt, sondern mit dem Testsignal. Das Testsignal verfügt über einen Ausschaltimpuls von 625 µs [9, S.62]. Für

eine Auswertung der Signale muss der Ausschaltimpuls berücksichtigt werden. Die Abtastrate des Echtzeitboards ist größer als der Ausschaltimpuls gewählt. Es ist trotzdem möglich, dass eine Abtastung während des Impulses stattfindet. Das hätte zur Folge, dass ein fehlerhaft interpretierter Zustandswechsel im Automaten ausgelöst wird. Als präventive Maßnahme wird eine software-technische Lösung bevorzugt, da diese am flexibelsten angepasst werden kann. Daher werden Lösungen mit Glättungskondensatoren oder RS-Flipflops nicht weiter verfolgt. Zur Lösung des Problems könnten mehrere Abtastpunkte auf einen Mittelwert verglichen werden. Dadurch würde im Falle des Ausschaltimpulses das Ergebnis einer logischen 1 zugeordnet werden und nur bei einer Unterbrechung des Signals, wodurch mehr als ein Abtastwert eine 0 ist, sich eine logische 0 ergeben. Alternativ kann auch wie bei einer SPS eine Ausschaltverzögerung nach IEC61131-3 eingefügt werden. Durch die Ausschaltverzögerung wird, wie in Abbildung 3.2 demonstriert, der Pegel noch für eine eingestellte Zeit gehalten. Wird das Eingangssignal innerhalb der Zeitspanne wieder eingeschaltet, bleibt das Signal bestehen und die Verzögerung startet bei der nächsten fallenden Flanke erneut. Erst wenn der Pegel nach der Zeitspanne auf dem Nullpegel verbleibt, wird der Pegel des Ausgangssignals angeglichen.

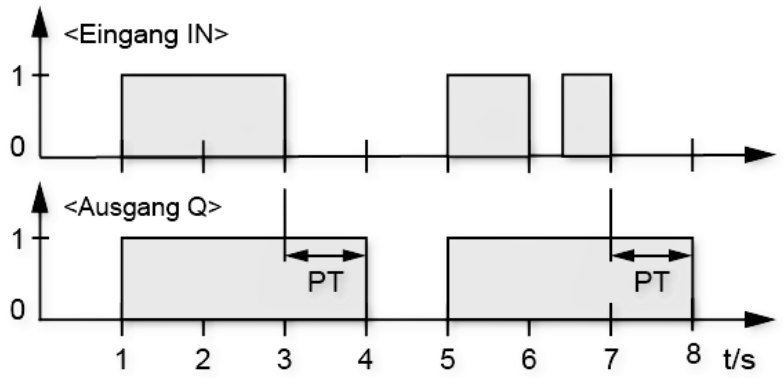


Abbildung 3.2: Zeitlicher Verlauf einer Ausschaltverzögerung [20]

Damit der Zustandsautomat auf die Steuerung des Industrieroboters zugreifen kann, müssen die Schnittstellen der Robotersteuerung mit dem Echtzeitboard signalkompatibel verdrahtet werden. Auf der Seite des dSPACE CP1103 Echtzeitboards wird hierzu die I/O-Schnittstelle CP30 aus Abbildung 3.3 verwendet. Sie verfügt über einen Anschluss für einen 50-poligen Sub-D Steckers und ermöglicht eine Übertragung von bis zu 32 Eingangs- oder Ausgangssignalen, die direkt mit Simulink-Bausteinen des DS1103 Master PPC angesteuert werden können. Für eine Verbindung von dem Echtzeitboard und der Robotersteuerung besteht die Option, die I/Os der CP30 Schnittstelle für die X12 und X11 Schnittstellen aufzuteilen. Das kann sinnvoll sein, weil für das Emulieren der Steuerung hauptsächlich Signale ausgelesen werden müssen.

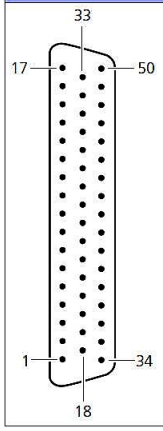
Digital I/O Connector (CP30)	Pin	Signal	Pin	Signal	Pin	Signal
	17	GND			50	VCC (+5 V)
	16	IO30	33	IO31	49	$\overline{\text{INT3}}$
	15	IO28	32	IO29	48	$\overline{\text{INT2}}$
	14	IO26	31	IO27	47	$\overline{\text{INT1}}$
	13	IO24	30	IO25	46	$\overline{\text{INT0}}$
	12	IO22	29	IO23	45	GND
	11	IO20	28	IO21	44	GND
	10	IO18	27	IO19	43	GND
	9	IO16	26	IO17	42	GND
	8	IO14	25	IO15	41	GND
	7	IO12	24	IO13	40	GND
	6	IO10	23	IO11	39	GND
	5	IO8	22	IO9	38	GND
	4	IO6	21	IO7	37	GND
	3	IO4	20	IO5	36	GND
	2	IO2	19	IO3	35	GND
	1	IO0	18	IO1	34	GND

Abbildung 3.3: Aufbau der dSPACE Schnittstelle CP30 [2, S.135]

Andererseits ist es auch konfigurierbar, digitale Signale über den DS1103 Slave DSP einzulesen. Die I/Os sind über die Anschlüsse CP31 oder CP29 erreichbar. Damit nur ein weiterer Stecker angeschlossen werden müsste, werden nur die Simulink-Bausteine der CP31 Schnittstelle für die Erweiterung mit der X11 Schnittstelle im Simulink-Modell eingefügt. Die verwendbaren I/Os der CP31 sind in Abbildung 3.5 aufgelistet. Durch die Erweiterungsmöglichkeit empfiehlt es sich, die 32 I/Os des CP30 auf die 16 Ein- und Ausgänge der X12 Schnittstelle aufzuteilen. Die Verbindungen über die X11 und X12 Schnittstelle funktionieren dadurch über separate Kabelverbindungen.

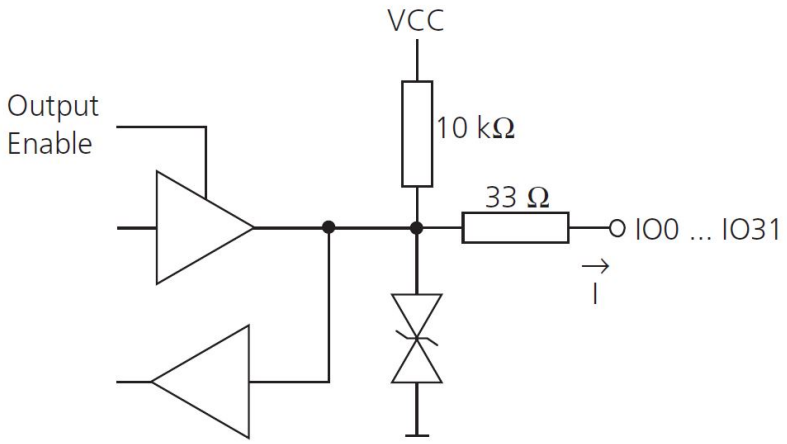


Abbildung 3.4: Elektrische Charakteristik eines I/O-PINs [2, S.171]

Standardmäßig sind die I/Os des Echtzeitboards durch „Pull Up“-Widerstände auf ein „High“-Signal konfiguriert. In der Abbildung 3.4 ist der Widerstand mit $10\,\text{k}\Omega$ angegeben [2, S.135].

Wenn an dem Echtzeitboard ein Anschluss nicht belegt ist, steht dadurch das Eingangssignal auf 1. Im Gegensatz dazu wird von den Blöcken in Simulink eine 0 ausgegeben, wenn sie nicht verendet werden. Das muss bei Tests und Simulationen mit Simulink beachtet werden. Daher empfiehlt es sich, nur Tests in Verbindung mit ControlDesk durchzuführen. Denn hierbei ist das Simulink-Modell auf das Echtzeitboard geladen und verwendet die realen Ausgänge.

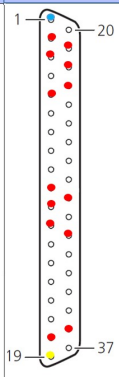
Signal	Channel/Bit Numbers of Related RTI Blocks/RTLib Functions				I/O Pin on ...			Connector (CP31)				
	Related RTI Block(s)	Ch/Bit (RTI)	Related RTLib Functions	Ch/Bit (RTLib)	DS1103	Sub-D Conn.	CP/CLP					
SPWM7 *	DS1103SL_DSP_BIT_IN_Cx/ DS1103SL_DSP_BIT_OUT_Cx	Bit 4	See *Slave DSP Bit I/O Unit* (DS1103 RTLib Reference)	Group 2 bit 0	P2 71	P2B 29	CP31 10		1	GND		
SPWM8 *		Bit 5		Group 2 bit 1	P2 72	P2A 29	CP31 29		2	SCAP1	20	GND
SPWM9 *		Bit 6		Group 2 bit 2	P2 73	P2B 13	CP31 11		3	SCAP3	21	SCAP2
ST2PWM *		Bit 7		Group 2 bit 4	P2 60	P2A 27	CP31 5		4	GND	22	SCAP4
ST3PWM		Bit 8		Group 2 bit 5	P2 61	P2B 11	CP31 24		5	ST2PWM	23	ST1PWM
STMRDIR		Bit 9		Group 2 bit 6	P2 55	P2B 10	CP31 31		6	GND	24	ST3PWM
STMRCLK		Bit 10		Group 2 bit 7	P2 56	P2A 10	CP31 12		7	SPWM1	25	GND
SXF		Bit 12		Group 3 bit 2	P2 83	P2B 31	CP31 18		8	SPWM3	26	SPWM2
SBIO		Bit 13		Group 3 bit 3	P2 84	P2A 31	CP31 36		9	SPWM5	27	SPWM4
SCAP1 *		Bit 14		Group 3 bit 4	P2 77	P2B 30	CP31 2		10	SPWM7	28	SPWM6
SCAP2 *		Bit 15		Group 3 bit 5	P2 78	P2A 30	CP31 21		11	SPWM9	29	SPWM8
SCAP3 *		Bit 16		Group 3 bit 6	P2 79	P2B 14	CP31 3		12	STMRCLK	30	GND
SCAP4 *		Bit 17		Group 3 bit 7	P2 80	P2A 14	CP31 22		13	GND	31	STMRDIR
									14	STINT1	32	SPDPINT
									15	GND	33	STINT2
									16	SSIMO	34	SSOMI
									17	SCLK	35	SSTE
									18	SXF	36	SBIO
									19	VCC (+5 V)	37	GND

Abbildung 3.5: Aufbau und I/O-Liste der CP31 [2, S.151 und S.136]

Da die Signale der Steuerung sowohl bei der X12, als auch bei der X11 Schnittstelle für ein „High“- Signal 24 Volt und für ein „Low“ 0 Volt verwenden und die I/Os des Echtzeitboards für ein „High“-Signal 5 Volt und für ein „Low“-Signal „0“ Volt benutzen, muss eine Übersetzungsplatine für die Spannungsanpassung entwickelt werden. Eine Anpassung der Signale über Spannungsteiler für die signalkompatible Verdrahtung kommt als Lösung nicht in Frage, denn die Kommunikation soll in beide Richtungen ermöglicht werden. Durch die Spannungsteiler könnten lediglich die 24 Volt Spannung auf 5 Volt übersetzt werden, aber anders herum ist eine Übertragung nicht möglich. Zusätzlich müsste die Potentiale der Robotersteuerung und des Echtzeitboards angeglichen werden. Hier empfiehlt es sich eine galvanische Trennung vorzusehen. Genauso ist eine Lösung mit Relais denkbar, die in Abhängigkeit des anliegenden Signals schalten. Solche besitzen aber eine langsamere Schaltgeschwindigkeit, so dass die Echtzeitfähigkeit verloren wäre. Daher empfiehlt es sich, Operationsverstärker oder Optokoppler für die Spannungsanpassung zu verwenden, wobei die Optokoppler wegen der galvanischen Trennung zu bevorzugen sind.

Um Daten von der X12-Schnittstelle abfragen zu können, müssen die Daten zuerst an die Schnittstelle geschickt werden. Dazu bedarf es einer Zuordnung der realen Ein- und Ausgänge zu den Systemvariablen. Die X12-Schnittstelle ist innerhalb der KR C4 compact direkt mit

den Eingangs- und Ausgangsklemmen des Beckhoff-Buskopplers verkabelt. Die Ein- und Ausgänge der Klemmen werden als „Channels“ bezeichnet. Die Einstellung des Buskopplers und damit die Zuweisung der einzelnen Ein- und Ausgänge erfolgt in WorkVisual. Dafür ist der Buskoppler über den „Extension Bus“ an den Hauptplatinen der Steuerung angeschlossen. Der Extension Bus ist ein Bussystem innerhalb der Steuerung. Über den Bus kann der Buskoppler auf boolesche Variablen zugreifen, die für die digitalen Ein- und Ausgänge der Steuerung stehen. Die Variablen heißen $IN[N] oder $OUT[N], wobei N einer natürlichen Zahl zwischen 1 und 4096 entsprechen kann. Die Zuordnung der digitalen Ein- und Ausgänge wird über die EA-Verschaltung in WorkVisual wie in Abbildung 3.6 vorgenommen. Um in WorkVisual Änderungen in der EA-Verschaltung vorzunehmen, muss in der Projektstruktur die Sicherheitssteuerung mit einem Doppelklick angewählt werden. Sie ist in dem Reiter „Geräte“ unter „Cell\Steuerung1\Sicherheitssteuerung“ zu finden. Alternativ funktioniert auch ein Doppelklick auf “Cell\Steuerung 1“. Änderungen sind im Kontextmenü mit der rechten Maustaste einstellbar. Wichtig hierbei ist, dass dazu in Teil 1 des Bildschirms der Reiter „KR C E/As und in Teil 2 der Reiter „Feldbusse“ ausgewählt werden muss, da ansonsten die Verbindungsmöglichkeiten in Bildschirmteil 3, 4 und 5 ausgeblendet werden. Für die Verschaltung von zum Beispiel $OUT[1] auf Pin 17 der X12 Schnittstelle muss nun Bildschirmteil 1 „Digitale Ausgänge“ angewählt werden. Dadurch erscheint in Bildschirmteil 4 eine Liste, bei der $OUT[1] markiert werden kann. Nun muss in Bildschirmteil 2 der „KUKA Extension Bus (SYS-X44)“ geöffnet werden. Durch Anwahl der Ausgangsklemme EL2809 erscheinen in Bildschirmteil 5 die Kanäle. Da die Pins 1 bis 16 der X12 Schnittstelle Eingänge sind, sind Pin 17 bis 32 die Ausgänge. Daher wird in diesem Fall „Channel 1.Output“ ausgewählt und anschließend im Kontextmenü der rechten Maustaste „Verbinden“ angeklickt. Hierbei gilt zu beachten, dass IN[1025] vom System aus immer gesetzt und IN[1026] immer rückgesetzt ist und nicht durch ein Eingangssignal geändert werden kann. Die eingestellten Verbindungen erscheinen in Bildschirmbereich 3 und können hier auch wieder rückgängig gemacht werden.

Um die internen Systemvariablen auf die X12-Schnittstelle zu übertragen, müssen sie noch im KRL-Editor den booleschen Variablen zugewiesen werden. Überwiegend gibt es für die Systemvariablen bereits eine empfohlene Zuordnung, die für die Konfiguration benutzt werden sollte. Denn es wird vom System nicht überprüft, ob ein digitaler Ausgang mehreren Systemvariablen zugeordnet ist. In der Konfigurationsdatei der Steuerung, die unter „KRC:\STEU\Mada\$machine.dat“ zu finden ist, sind standardmäßig bereits einige Aus- und Eingänge bestimmten Systemvariablen zugeordnet. Für die Benutzung bereits zugeordneten $OUT[N] Variablen ist ein einfaches Verbinden über WorkVisual ausreichend. Da aber nicht alle für den Zustandsautomaten nötigen Systemvariablen bereits zugewiesen sind, muss die Zuweisung im Submit-Interpreter erfolgen. Hierfür muss ein KRL Code in der Datei „SPS.SUB“ eingefügt werden. Die SPS.SUB ist in Bereiche eingeteilt, die „Folds“ genannt werden. Die nötigen Einstellungen müssen in dem „Fold USER PLC“ über den KRL Editor eingefügt werden [18, S.479]. Bei den bereits vom System zugeordneten Systemvariablen ist

eine Programmierung in der SPS.SUB nicht möglich, da die Variablen durch die Zuweisung in der „machine.dat“ schreibgeschützt werden. Einige der Systemvariablen sind vom Datentyp „ENUM“ oder nur für jede Achse des Roboters einzeln abfragbar. Für solche Variablen ist eine direkte Zuweisung zu den digitalen Ausgängen nicht möglich. Damit solche Systemvariablen trotzdem abgefragt werden können, muss für jede Abfrage eine IF-Bedingung in der „Fold USER PLC“ geschrieben werden, die die Systemvariablen auswertet und einen logischen Wert zurückgibt. Der Wert kann dann mit den digitalen Ausgängen verknüpft werden.

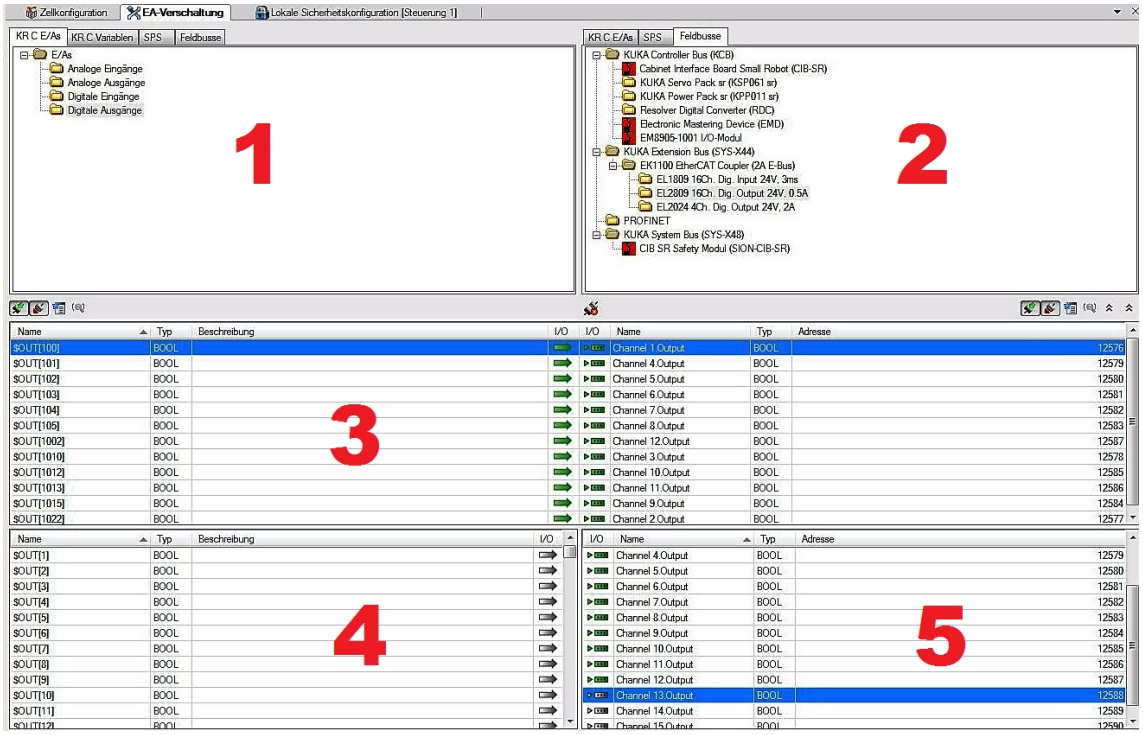


Abbildung 3.6: Verbinden der digitalen Ein- und Ausgänge mit WorkVisual

3.3 Umsetzungsmöglichkeiten für den Zustandsautomaten

Es gibt verschiedene Vorgehen, einen Zustandsautomaten zu realisieren. Wird der Automat ähnlich wie ein Moore-Automat aufgebaut, sind die Ausgabewerte nur von den Zuständen abhängig. In Simulink Stateflow kann ein solcher Automat mit den Schlüsselwörtern und Variablenzuweisungen innerhalb der Zustände aufgebaut werden. Sind eher die Zuweisungen an den Transitionen vorteilhaft für die Funktionen des Automaten, wie bei einem Mealy-Automaten, ist auch diese Variante in Simulink Stateflow mit den geschweiften Klammern

in den Transitionen umsetzbar. Für den Automaten muss bekannt sein, in welchem Zustand gestartet werden soll. Dafür kann ein Startzustand definiert werden, von dem aus in jeden anderen Zustand gewechselt werden kann oder es wird ein Zustand beim Start angenommen und die Anpassung des Zustands erfolgt automatisch nach den Werten der Eingangsvariablen.

Die Aufgabe des Zustandsautomaten ist, die Robotersteuerung zu emulieren. Damit der Zustandsautomat die gleichen Reaktionen ausführt wie die Robotersteuerung, werden die Signale, die die Robotersteuerung erhält, auch für den Automaten verwendet. Da der Automat über eine Schnittstelle mit der Robotersteuerung verbunden ist, kann auch zur Kontrolle von der Steuerung ein „Feedback“ der Zustände eingeholt werden, sofern ein Zustand mit einer Systemvariablen abgefragt werden kann. Wenn der Automat seinen Zustand wechselt und daraufhin sofort wieder ein Wechsel in den vorigen Zustand zurück erfolgt, würde der Zustandswechsel in der Visualisierung gar nicht wahrgenommen werden. Für einen solchen Fall ist es sinnvoll, eine Bestätigung des Zustandswechsels einzuführen. Dadurch kann sichergestellt werden, dass jeder Zustandswechsel auch erkannt wird.

In Simulink Stateflow kann der Zustandsautomat in einer großen Chartumgebung und mehreren Subcharts realisiert werden, die über lokale Variablen untereinander Daten austauschen. Ein anderes Vorgehen ist, den Zustandsautomaten in mehrere kleineren Chartumgebungen zu unterteilen und die für die Kommunikation untereinander verwendeten lokalen Variablen durch „Inputs“ und „Outputs“ zu ersetzen, so dass die einzelnen Chartumgebungen in Simulink über Ports miteinander verbunden sind. Eine Darstellung in mehreren Chartumgebungen hat den Vorteil, dass die Funktionen getrennt voneinander verarbeitet werden und die Verbindungen untereinander besser sichtbar sind. Die große Chartumgebung hat den Vorteil, dass alle Variablen nicht für jedes Chart einzeln definiert werden müssen und die Auswertung der Zustände einfacher gestaltet werden kann. Die Emulation der Zustände und die Verarbeitung der Ursache findet hierbei an der selben Stelle im Simulink-Modell statt.

Der Zustandsautomat soll die Zustände der Stopp-Reaktionen und auch die Ursache ausgeben. Einerseits ist es möglich die Ermittlung der Ursache und die Ermittlung der Stopp-Reaktion voneinander zu trennen. In dem einen Teilbereich des Automaten würden die Stopp-Reaktionen als Zustände und in einem anderen parallelen Teilbereich die Ursache als Zustand ausgewertet werden. Andererseits ist es auch denkbar, die Funktionen in nur einem großen Automaten zusammen darzustellen. In einen solchen Automaten wären die Zustände der Stopp-Reaktionen in jeweils einem Chart untergebracht und die Ursachen wären in untergeordneten Charts innerhalb der Zustände verschachtelt. Mit den untergeordneten Charts würde dann für jede Stopp-Reaktion einzeln die Ursache ausgewertet werden.

Der Wechsel zwischen den Zuständen des Zustandsautomaten findet über Transitionen statt. Dabei kann der Wechsel in einen Stoppzustand verschiedene Auslöser haben. In dem Automaten kann für jeden Auslöser eine eigene Transition eingefügt werden. Der Vorteil dabei

ist, dass für jeden Auslöser eine eigene Transition existiert. Der Nachteil besteht aber darin, dass durch die vielen Transitionen der Automat schnell an Übersichtlichkeit verliert. Ebenso muss berücksichtigt werden, dass die einzelnen Transitionen voneinander abhängig sind. Deswegen muss ausgeschlossen werden, dass eine Transition erfüllt ist und ein Zustandswechsel erfolgt, aber durch andere erfüllte Bedingungen wieder ein Zustandswechsel ausgelöst wird, ohne den Ursprung des vorigen Zustandswechsels zu ändern. Einfacher ist es für jeden Zustandswechsel, nur eine Transition vorzusehen. Die einzelnen Auslöser müssen dafür als logischer Ausdruck „ODER“-verknüpft werden. Dadurch ergeben sich aber schnell lange logische Ausdrücke. In den Unterlagen von KUKA ist angegeben, welcher Auslöser für eine Stopp-Reaktion nötig ist. Zu den Rücktransitionen gibt es aber in den meisten Fällen keine näheren Informationen, so dass sie durch Schlussfolgerungen aus der Logik der Hintransitionen hergeleitet werden müssen. Die langen Ausdrücke nehmen dadurch bei der Erstellung der Rücktransitionen noch an Komplexität zu. Hierfür ist eine Überprüfung der einzelnen logischen Ausdrücke während der Erstellung des Automaten hilfreich.

Eine weitere Variante eines Zustandsautomaten ist nicht bei einer Transition mit vielen erfüllten Bedingungen zwischen den Hauptzuständen zu wechseln, sondern stattdessen Zwischenzustände einzufügen. Bei jeder Signaländerung würde unter den Zwischenzuständen gewechselt werden. Das hat den Vorteil, dass alle Signale eine Änderung und einen abhängigen Ausgabewert erzeugen und nicht nur, wenn die langen Logikketten erfüllt sind. Da aber viele Signale eingelesen werden und für jede Signalkombination ein Zustand erforderlich ist, wären hierfür sehr viele Zwischenzustände nötig.

Für das Simulink-Modell besteht die Möglichkeit, die Schnittstellen der Chartumgebung direkt mit den dSPACE-Bausteinen zu verbinden. Wenn keine direkte Verbindung hergestellt wird, sondern noch eine Schaltebene dazwischen eingefügt wird, kann zwischen den dSPACE-Bausteinen und Konstanten geschaltet werden. Die Signale der Bausteine können dadurch auf einfache Weise simuliert werden. Ebenso ist es sinnvoll, das Simulink-Modell modular aufzubauen. Dadurch sind die einzelnen Module einfacher austauschbar. Eine Erweiterung des Automaten und eine Implementierung für andere Anwendungen ist somit einfacher möglich.

Die Auswertung der Zustimmschalter und Überwachung des programmierten Arbeitsraums erfolgt über eine Logik. Die Logik kann in der Chartumgebung als eigenes Subchart eingebunden oder bereits in Simulink mit Logikbausteinen ausgewertet werden.

Um sich für oder gegen die Umsetzung von Lösungsvarianten des Automaten zu entscheiden, wurde eine Bewertungsmatrix erstellt. Als Kriterien werden ein simpler Aufbau, die Umsetzung der Funktionalität und die durch die Komplexität erhöhte Fehleranfälligkeit einkalkuliert. Weitere berücksichtigte Kriterien sind die Übersichtlichkeit im gesamten Automaten, die zusätzliche Erweiterbarkeit, die Lesefreundlichkeit in Simulink und ob zusätzliche Variablen definiert werden müssen. Die Kriterien sind nach der Notwendigkeit gewichtet. Wie sehr eine

Teillösung ein Kriterium erfüllt, wird mit einem Wert von 0 bis 5 bewertet, wobei 5 der höchste Erfüllungsgrad ist. Die Bewertungsmatrix ist in Abbildung 3.7 dargestellt. Alle Teillösungen, die über 80 Prozent erreicht haben, werden in dem Zustandsautomaten realisiert.

Zustandsautomat		Transitionen zusammengefasst		Transition pro Auslöser		Chartumgebung + Subcharts		Mehrere Chartumgebungen	
Gewichtung G	Kriterium	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B
20%	Simpler Aufbau	4	0,16	5	0,2	5	0,2	3	0,12
15%	Funktionalität	5	0,15	5	0,15	5	0,15	5	0,15
15%	Fehleranfälligkeit	2	0,06	2	0,06	4	0,12	4	0,12
15%	Übersichtlichkeit	4	0,12	1	0,03	5	0,15	4	0,12
10%	Erweiterbarkeit	5	0,1	3	0,06	5	0,1	3	0,06
15%	Lesefreundlichkeit	4	0,12	5	0,15	4	0,12	5	0,15
10%	Variablenzahl	5	0,1	2	0,04	3	0,06	2	0,04
100%	**Summen**		81%		69%		90%		76%

Zustandsautomat		Festgelegter Startzustand		Angenomener Startzustand		Meldungen bestätigen		Steuerungsabfrage "Feedback"	
Gewichtung G	Kriterium	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B
20%	Simpler Aufbau	4	0,16	4	0,16	3	0,12	2	0,08
15%	Funktionalität	4	0,12	5	0,15	5	0,15	5	0,15
15%	Fehleranfälligkeit	5	0,15	4	0,12	4	0,12	5	0,15
15%	Übersichtlichkeit	4	0,12	3	0,09	5	0,15	4	0,12
10%	Erweiterbarkeit	4	0,08	5	0,1	4	0,08	5	0,1
15%	Lesefreundlichkeit	5	0,15	3	0,09	4	0,12	4	0,12
10%	Variablenzahl	4	0,08	5	0,1	3	0,06	4	0,08
100%	**Summen**		86%		81%		80%		80%

Zustandsautomat		Parallel Ursachen Ermitteln		Ursachen unter Charts Ermitteln		Zwischenzustände einfügen		Schaltbare Ebene einfügen	
Gewichtung G	Kriterium	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B
20%	Simpler Aufbau	4	0,16	2	0,08	3	0,12	5	0,2
15%	Funktionalität	5	0,15	5	0,15	3	0,09	5	0,15
15%	Fehleranfälligkeit	4	0,12	3	0,09	4	0,12	5	0,15
15%	Übersichtlichkeit	4	0,12	4	0,12	2	0,06	4	0,12
10%	Erweiterbarkeit	5	0,1	2	0,04	4	0,08	5	0,1
15%	Lesefreundlichkeit	5	0,15	3	0,09	2	0,06	4	0,12
10%	Variablenzahl	3	0,06	5	0,1	3	0,06	5	0,1
100%	**Summen**		86%		67%		59%		94%

Zustandsautomat		Logik in Simulink-Subsystem		Logik in Chartumgebung		Modularer Systemaufbau		An- und Abwahl Erweiterungen	
Gewichtung G	Kriterium	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B	Erfüllung E	(G*E)/B
20%	Simpler Aufbau	3	0,12	3	0,12	4	0,16	3	0,12
15%	Funktionalität	5	0,15	4	0,12	5	0,15	5	0,15
15%	Fehleranfälligkeit	4	0,12	3	0,09	4	0,12	5	0,15
15%	Übersichtlichkeit	5	0,15	3	0,09	5	0,15	4	0,12
10%	Erweiterbarkeit	3	0,06	5	0,1	5	0,1	5	0,1
15%	Lesefreundlichkeit	5	0,15	4	0,12	5	0,15	4	0,12
10%	Variablenzahl	5	0,1	3	0,06	4	0,08	3	0,06
100%	**Summen**		85%		70%		91%		82%

Abbildung 3.7: Bewertungsmatrix des Zustandsautomaten

4 Realisierung

In diesem Kapitel wird die Umsetzung, der Aufbau und die Visualisierung des Zustandsautomaten dokumentiert. Ebenfalls wird die Verbindung zwischen der Robotersteuerung KR C4 compact und der dSPACE Box betrachtet.

4.1 Signalkompatible Verdrahtung

Für die Kommunikation zwischen dem dSPACE Echtzeitboard und der Robotersteuerung sind die CP30 und X12 Schnittstelle verbunden. Als Lösung für die Spannungsanpassung zwischen den Schnittstellen werden die Optokoppler PC817 der Firma SHARP verwendet [23]. Die Funktionsweise des Optokopplers besteht darin, dass eine Photodiode auf der einen Seite aufleuchtet und ein Phototransistor auf der anderen Seite das Licht erkennt und damit leitend wird. Für die Photodiode muss daher der Stromfluss mit einem Vorwiderstand begrenzt werden. Der maximale Strom der Diode beträgt 20 Milliampere. Damit dieser Wert unterschritten bleibt, wird der Strom an der Diode auf die Größenordnung von 10 Milliampere eingestellt und der Kollektorstrom des Phototransistors soll 5 Milliampere betragen. Nach dem ohmschen Gesetz ergeben sich die Widerstände wie folgt:

$$R_{ges} = \frac{U_{VCC}}{I_R} = \frac{5V}{0,01A} = 500\Omega \tag{4.1}$$

$$R_{DiodeIn} = R_{ges} - R_{IO} = 500\Omega - 33\Omega = 467\Omega \Rightarrow 330\Omega$$

$$R_{Diode2Aus} = \frac{U}{I_R} = \frac{24V}{0,01A} = 2400\Omega \Rightarrow 2,2k\Omega \tag{4.2}$$

$$R_{Kollektor} = \frac{U}{I_F} = \frac{24V}{5mA} = 4800k\Omega \Rightarrow 4,7k\Omega \tag{4.3}$$

Mit den errechneten Werten für die Widerstände wurde das Schaltbild von einer „CP30-X12“-Übersetzung und einer „X12-CP30“-Übersetzung mit dem Programm „EAGLE“ erstellt. Das Schaltbild ist in Abbildung 4.1 dargelegt. Nach dem Schaltbild wurde eine Europlatine mit entsprechender Bestückung erstellt, die 16 mal die „CP30-X12“-Übersetzung und 16 mal die „X12-CP30“-Übersetzung enthält. Der Aufbau des Platinenlayouts ist in Abbildung 4.2

zu erkennen. Dabei werden GNDA Potential und VCC von dem Echtzeitboard versorgt. Die Übersetzungsplatine braucht eine zusätzliche Spannungsversorgung von 24 Volt, damit die 5 Volt Pegel auf 24 Volt verstärkt werden können. Außerdem hat die Platine ein Kabel für die Masse GND. Durch das Kabel wir das 0 Volt Potenzial von der KR C4 compact bezogen und auf Seite der X12 für die Platine festgelegt. Mit der X55 Schnittstelle sind die Verbindungen für die 24 Volt und GND möglich [19, S.26]. Ansonsten kann die 24 Volt Spannungsversorgung auch mit einem externen Spannungsgenerator erfolgen. Das GND Potenzial kann dafür an dem Erdungsanschluss der Steuerung abgegriffen werden.

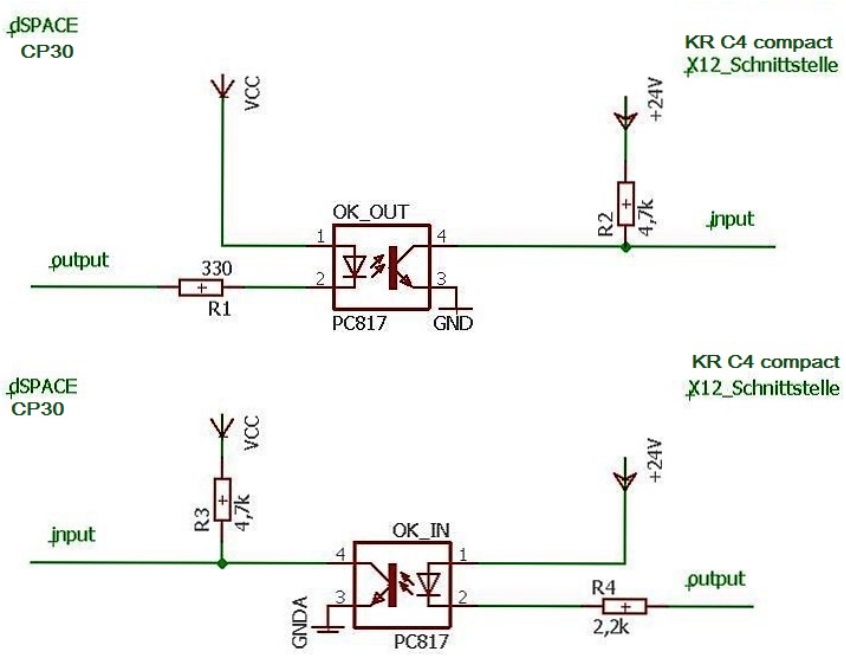


Abbildung 4.1: Schaltbild von Input und Output

Um die Europlatine vor äußeren Einflüssen und Beschädigungen zu schützen, ist sie in einer Hülle untergebracht. Über vier Wannenstecker ist die Platine mit Flachbandkabeln verbunden. Die Wannenstecker sind für die richtige Zuordnung farbig gekennzeichnet und nur in eine Richtung anschließbar. Die Flachbandkabel führen jeweils zu einem 50-poligen Sub-D-Stecker. Die „male“-Variante wird mit der X12 Schnittstelle der Robotersteuerung verbunden und die „female“-Variante an die CP30 Schnittstelle des Echtzeitboards angeschlossen.

Um die Signale auf die X12 Schnittstelle zu legen, muss der Buskoppler, wie in Kapitel 3.2 beschrieben, mit Strom versorgt werden. Dafür sind die entsprechenden Signale in der X55 überbrückt. Weil der Buskoppler aktiv ist, konnten in WorkVisual unter EA-Verschaltung die digitalen Ausgänge zugeordnet werden. Welche digitalen Ausgänge bestimmten „Channels“ zugeordnet wurden, kann den Tabellen 4.1 und 4.2 entnommen werden.

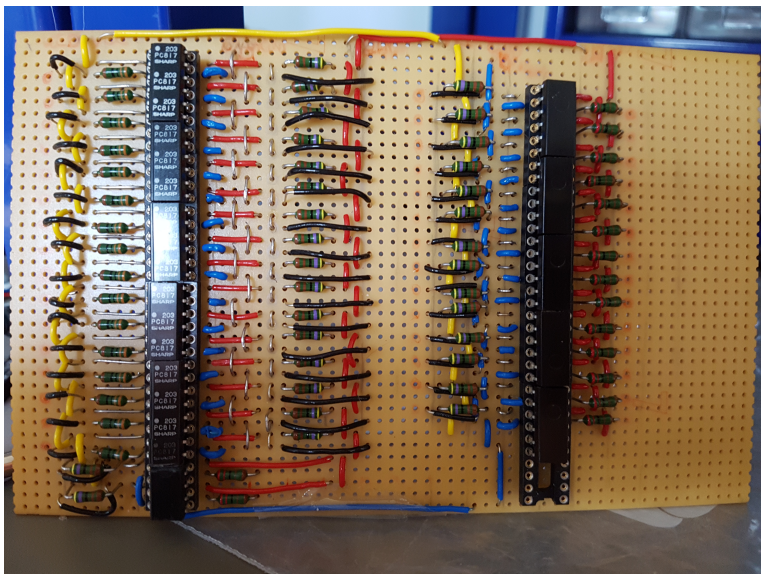


Abbildung 4.2: Spannungswandlerplatine

Um die passenden Systemvariablen für die Abfragen des Automaten herauszusuchen, können in der E/A-Verschaltung von WorkVisual unter dem Reiter „KR C Variablen“ die Systemvariablen angezeigt werden. Hier sind wenige Variablen bereits mit einer Kurzbeschreibung versehen. Die Variablen zur Überwachung des Arbeitsraumes und der Geschwindigkeiten sind in der Bedienungsanleitung „SafeOperation“ [13] zu finden. Für die weiteren ist eine Variablenübersicht vom KUKA Support verfügbar [16]. Informationen zu den Systemvariablen, die in beiden Unterlagen nicht enthalten sind, können direkt beim KUKA Support angefragt werden. Der in der „SPS.SUB“ programmierte Code ist im Anhang A beigefügt. Die durch das Semikolon auskommentierten Zuweisungen, sind die bereits in der „machine.dat“ zugewiesenen Systemvariablen. Die „IF-Abfragen“ werden zur Auswertung der Zustimmschalter benötigt, da die zugehörige Systemvariable vom Typ „ENUM „ ist. Ebenso wird für die Auswertung der Geschwindigkeiten eine „IF-Abfrage“ eingesetzt, weil für alle Achsen des Roboters eigene Variablen existieren und die mit der Abfrage zu einer zusammengefasst werden können.

Eine Übersicht über die komplette Verbindungskette ist in den Tabellen 4.1 und 4.2 zu finden. Eine Zeile spiegelt den Fluss eines kompletten Signals wieder. Dabei ist in der linken Spalte angegeben, welcher I/O der Simulink-Bausteine, auf welchem Pin der CP30 Schnittstelle zu finden ist. Die Pins der CP30 Schnittstelle sind mit den angegebenen Pin der X12 Schnittstelle über die Übersetzungsplatine verbunden und in der Steuerung den Kanälen des Beckhoff Buskopplers zugeordnet. Weiteren Spalten lässt sich entnehmen, welche digitalen Aus- und Eingänge mit WorkVisual vernetzt und in der PLC des Submit-Interpreters oder der „machine.dat“- Datei einer Systemvariablen zugewiesen sind.

Tabelle 4.1: Pinbelegung

dSPACE CP30		KUKA X12		KR C4 compact	
Bit	PIN	PIN	Kanal	Digitaler Ausgang	Systemvariable
I/O 0	1	17	Channel1.Output	OUT[100]	$SAFE_FS_STATE
I/O 1	18	18	Channel2.Output	OUT[1022]	$PRO_MOVE
I/O 2	2	19	Channel3.Output	OUT[1010]	$STOP_MESS
I/O 3	19	20	Channel4.Output	OUT[101]	$POWER_FAIL
I/O 4	3	21	Channel5.Output	OUT[1015]	$SPOC_MOTION_ENABLE
I/O 5	20	22	Channel6.Output	OUT[102]	$SAFETY_DRIVE_ENABLE
I/O 6	4	23	Channel7.Output	OUT[1012]	$PERI_RDY
I/O 7	21	24	Channel8.Output	OUT[1002]	$ALARM_STOP_INTERN
I/O 8	5	25	Channel9.Output	OUT[103]	$SR_SAFEMON_ACTIVE
I/O 9	22	26	Channel10.Output	OUT[104]	$SR_AXISSPEED_OK & SR_CARTSPEED_OK
I/O 10	6	27	Channel11.Output	OUT[105]	$SR_RANGE_OK
I/O 11	23	28	Channel12.Output	OUT[1011]	$USER_SAF
I/O 12	7	29	Channel13.Output	OUT[106]	$SAFETY_SW
I/O 13	24	30	Channel14.Output	OUT[107]	$SR_SAFETYSTOP1_ACTIVE
I/O 14	8	31	Channel15.Output	OUT[108]	$SR_SAFETYSTOP2_ACTIVE
I/O 15	25	32	Channel16.Output	OUT[1013]	$ALARM_STOP
GNDA	17	mit GNDA der Platine verbunden			

Tabelle 4.2: Pinbelegung Fortsetzung

dSPACE CP30		KUKA X12		KR C4 compact	
Bit	PIN	PIN	Kanal	Digitaler Eingang	Systemvariable
I/O 16	9	1	Channel1.Input	IN[100]	$CONF_MESS
I/O 17	26	2	Channel2.Input	IN[101]	$IMM_STOP
I/O 18	10	3	Channel3.Input	IN[102]	$SAFEGATE_OP
I/O 19	27	4	Channel4.Input	IN[103]	$MOVE_ENABLE
I/O 20	11	5	Channel5.Input	IN[104]	$LOOP_CONT
I/O 21	28	6	Channel6.Input	IN[105]	$DRIVES_ENABLE
I/O 22	12	7	Channel7.Input		
I/O 23	29	8	Channel8.Input		
I/O 24	13	9	Channel9.Input		
I/O 25	30	10	Channel10.Input		
I/O 26	14	11	Channel11.Input		
I/O 27	31	12	Channel12.Input		
I/O 28	15	13	Channel13.Input		
I/O 29	32	14	Channel14.Input		
I/O 30	16	15	Channel15.Input		
I/O 31	33	16	Channel16.Input		
VCC (+5V)	50	mit VCC der Platine verbunden			
24V der Platine extern Stromquelle oder X55 GND der Platine an Masse der Steuerung					

4.2 Aufbau des Zustandsautomaten

Um den Umfang des Zustandsautomaten festzulegen, wurde die Systemgrenze für den Automaten so definiert, dass die Stopp-Reaktionen und alle daran beteiligten Sicherheitsfunktionen berücksichtigt werden. Das schließt beispielsweise die Darstellung der Interpreter aus, da sie keine direkte Sicherheitsrelevanz haben, sondern nur für die Ausführung von Programmen zuständig sind. Dabei werden vom Roboter-Interpreter die Roboterbewegungen und vom Submit-Interpreter kleinere Programme, wie die Signalausgabe für Diagnose- oder Überwachungsaufgaben, bearbeitet [18, S.477]. Der Zustand der Interpreter wird durchgängig auf dem SmartPAD angezeigt.

Das Simulink Modell

Eine schematische Darstellung in Abbildung 4.3 zeigt den Aufbau des Simulink-Modells. Das vollständige Modell ist im Anhang in Abbildung B.1 einzusehen. Es befindet sich in den Abbildungen in der Startkonfiguration. In der Konfiguration ist die Simulation aktiviert und es wird keine Stopp-Reaktion ausgelöst. Die Auslöser werden direkt am Automaten manipuliert und das Setzen von Signalen auf der X12-Schnittstelle und Empfangen von Signalen der X11-Schnittstelle ist deaktiviert. Wenn der Automat erneut auf das Echtzeitboard geladen wird, wird der Automat immer auf diese Ausgangssituation zurückgesetzt. Das Simulink-Modell, über das der Zustandsautomat mit dem Echtzeitboard von dSPACE kommuniziert, ist für einen modularen Aufbau in mehrere Subsysteme unterteilt. Dadurch soll es vereinfacht werden, die Verbindungen der Signale zu überblicken. Für detailreichere Informationen kann ein Einblick in die Subsysteme erfolgen. Die DS1103-Eingangsblöcke werden mit Signalen von der X12 Schnittstelle versorgt. Ob die X11 Schnittstelle verwendet werden kann und von der X12 Schnittstelle nicht nur Signale empfangen, sondern auch gesendet werden können, lässt sich hier einstellen. Ebenso wählbar ist, ob bei einer Simulation die simulierten Ausgangswerte der X12 Schnittstelle oder die Werte direkt an den Eingängen des Automaten verwendet werden sollen. Dadurch muss beim anschließenden Simulieren in der Visualisierung, nicht immer an alle Logiken gedacht werden, sondern es können einfache direkte Tests vorgenommen werden. Durch das Anschließen und Zuschalten der X11 Schnittstelle werden die gleichen Signale redundant von beiden Schnittstellen abgefragt. Es wäre auch möglich, den Automaten um weitere Eingänge zu erweitern. Dazu müssen die von der X11 Schnittstelle eingelesen Signale, bei der X12 Schnittstelle durch andere Variablen ersetzt werden und der logische ODER Baustein vor dem entsprechenden Automateneingang entfernt werden.

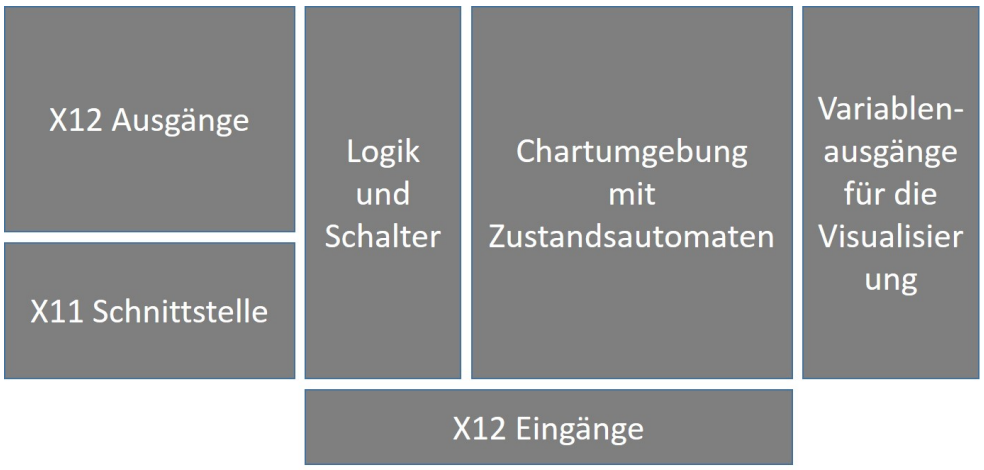


Abbildung 4.3: Schematische Darstellung des Simulink-Modells

Die Subsysteme der X12 Ein- und Ausgänge

Die aus den Ausgängen der X12 Schnittstelle ausgelesen Signale werden hier in das Simulink-Modell eingespeist oder ausgegeben. Die 16 Ein-und Ausgänge sind jeweils in ein Subsystem zusammengefasst. Die Eingänge sind im Anhang unter B.2 und die Ausgänge unter B.3 zu finden. Die im Zustandsautomaten verwendeten Variablennamen entsprechen nicht den Systemvariablen der Steuerung, da sie teilweise nicht selbsterklärend sind oder eine Systemvariable in verschiedenen Betriebsarten unterschiedliche Verwendungszwecke haben kann. Dadurch müssen in der Steuerung teilweise mehrere Variablen ausgewertet werden, um eine Information zu bekommen, die im Zustandsautomaten nach Möglichkeit mit nur einer Variablen abgedeckt wurde. Die verwendeten Systemvariablen sind in Tabelle 4.1 und Tabelle 4.2 den einzelnen Ports zugeordnet und auch an den Simulink-Blöcken des DS1103 notiert. Die Eingänge des Automaten sind so gewählt, dass in der Ausgangssituation alle Eingänge auf 0 gestellt sind und sobald eine 1 anliegt, eine Fehlerursache ausgelöst wird. Als Folge davon müssen einige Signale, bei denen die 0 der auslösende Wert ist, in dem Subsystem invertiert werden. Die Schaltung zwischen der Simulation und den realen Werten ist mit in dem Subsystem integriert. Einige Signale sind keine Auslöser einer Stopp-Reaktion, sondern Statusabfragen von Variablen. Die werden benötigt, um zwischen den Auslösern zu unterscheiden oder Funktionen ein- oder abzuschalten. Wenn eine solche Variable alleine geändert wird, löst das meist keinen Zustandswechsel aus. Für die Simulation ist der Eingang „Betrieb an“ als einziger bereits auf 1 geschaltet. Der Grund dafür ist, dass der Roboter in der Simulation hauptsächlich in Bewegung getestet wird und das hin und her schalten dadurch gespart werden kann. Das eingelesene Peri-Signal ist an mehrere Bedingungen geknüpft, die alle erfüllt sind, wenn das Signal gesetzt ist [10, S.59]. Wird ein Sicherungshalt ausgelöst und das Peri-Signal zurückgesetzt, lässt sich aber nicht

explizit unterscheiden, welche der Bedingungen ausschlaggebend für das Auslösen des Sicherungshalts ist. Daher wird das Peri-Signal nur eingelesen, um in der Simulation angezeigt zu werden, aber wird nicht für eine Überprüfung der Zustände verwendet.

Logik der Zustimmschalterstellungen

In der Logik wird berücksichtigt, ob ein Zustimmschalter sich in der Mittelstellung befindet und dadurch eine Fahrt des Roboters freigegeben ist oder ob die Zustimmung nicht erfolgt. Wenn die Zustimmung nicht erfolgt, kann der Zustimmschalter entweder in der gelösten oder durchgedrückten Position sein. Da das Lösen bei der normalen Bedienung erfolgt und das Durchdrücken als Panikstellung bezeichnet wird, wird bei beiden Stellungen eine andere Stopp-Reaktion durchgeführt. Aus den Signalen wird ausgewertet, in welcher Stellung sich die Schalter befinden. Außerdem können die Zustimmschalter am SmartPAD bis zu 15 s gleichzeitig in Mittelstellung gehalten werden, um ein Handwechsel zu ermöglichen. Befinden sie sich länger als die 15 s in dieser Stellung, wird ein Sicherheitshalt1 ausgelöst [13, S.47]. Mit den Variablen der Steuerung kann die Dauer der gleichzeitigen Mittelstellung nicht überprüft werden. Es wird von der SmartPAD-Verbindung das gleiche Signal gesetzt, als wenn der Zustimmschalter durchgedrückt wurde. Daher folgt der Zustandswechsel in den Sicherheitshalt1. Die Zustimmschalterlogik ist nur in den Betriebsarten T1 und T2 aktiv. In den anderen Betriebsarten wird über die gleichen Signale der Bedienerschutz ausgewertet und daher ebenfalls in der Logik berücksichtigt. Das aufgebaute Simulink-Modell ist in Abbildung 4.4 zu sehen.

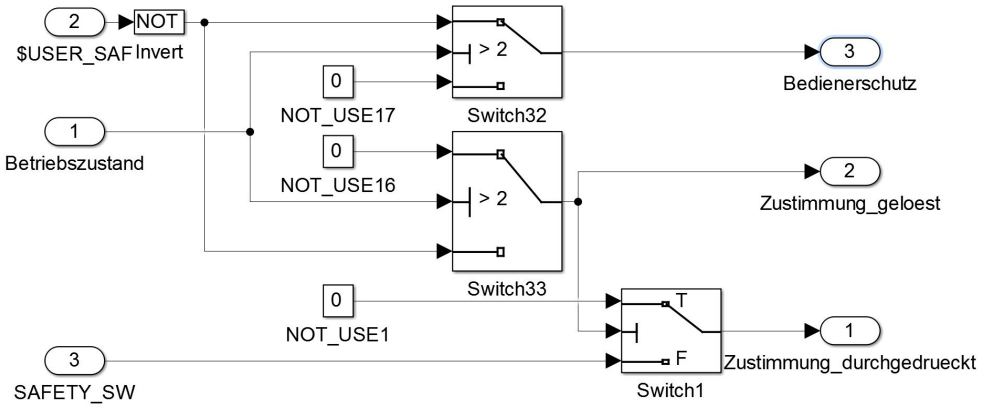


Abbildung 4.4: Zustimmschalterstellungen

Das Subsystem X11 Schnittstelle mit untergeordneten Subsystemen

Eine Erweiterung für das Einlesen von Signalen über die X11 Schnittstelle wurde bereits im Subsystem X11 Schnittstelle berücksichtigt, das im Anhang unter B.4 zu finden ist. Die Bausteine des DS1103 Slaves werden hierbei für das Einbinden der Signale verwendet und es lässt sich genauso wie bei den Ausgängen der X12 Schnittstelle auf eine Simulation umschalten. Hierbei ist zu beachten, dass die sicherheitsrelevanten Signale in der X11 Schnittstelle direkt an die Peripherien und Schalter angeschlossen sind. Dadurch werden beim Auslösen über die X11 Schnittstelle die Schalterstellungen ausgewertet und nicht wie bei der X12 Schnittstelle ein Signal empfangen, das von der Steuerung gesetzt wurde.

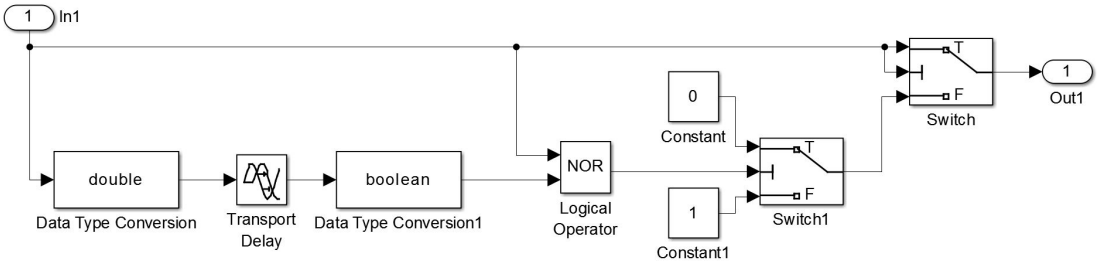


Abbildung 4.5: TOF Baustein

Das Programm Simulink enthält in seiner Standardbibliotek keinen eigenen TOF-Baustein, um den Abschaltimpuls des Testsignals zu berücksichtigen. Daher ist ein solcher mit der entsprechenden Funktionsweise in dem Subchart „TOF“ vereinfacht nachgebaut und in Abbildung 4.5 dargestellt. Eine Typenkonvertierung innerhalb des Bausteins ist nötig, um das Signal sowohl zeitlich zu verzögern und rückzukoppeln, als auch als boolesche Variable auszuwerten.

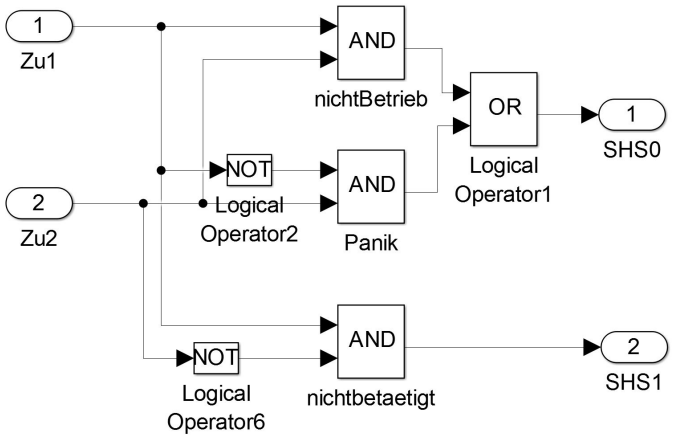


Abbildung 4.6: Subsystem LogExtZustimm[9, S.59]

Es ist auch möglich, bis zu zwei weitere Zustimmschalter an der X11 Schnittstelle anzuschließen. Für diesen Fall ist die Logik zum Einbinden der Schalter im Simulink-Modell berücksichtigt. Die Darstellung wird in Abbildung 4.6 abgebildet [9, S.59]. Wenn sie nicht angeschlossen sind, müssen die dazugehörigen Pins überbrückt sein. Also entsprechen sie in der Simulation im überbrückten Zustand einer 1.

Die Chartumgebung

Der Zustandsautomat ist in verschiedene Subcharts unterteilt, die gleichzeitig aktiv sind. Die Kommunikation der Subcharts untereinander erfolgt über lokale Variablen. Die Zahlen in den Subcharts geben die Reihenfolge der Ausführung an. Die Subcharts werden vom Automaten als parallele Zustände behandelt. Jedoch kann ein Prozessor nur eine Aufgabe in einem Prozessorzyklus aufrufen und bearbeiten, weshalb die Angabe einer Reihenfolge nötig ist. Der Zustand Safety braucht daher die „Execution Order 1“, da in diesem Subchart die lokalen Variablen initialisiert werden und damit erstmalig einen Wert zugeordnet bekommen. Bei einer anderen Reihenfolge würde ein Fehler vorliegen, weil die Variablen auf eine Bedingung geprüft würden, ohne vorher einen Wert zugewiesen bekommen zu haben.

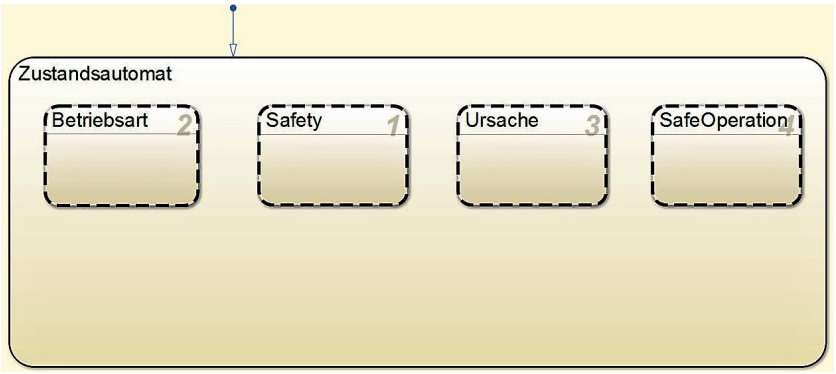


Abbildung 4.7: Chartumgebung

Das Subchart Betriebsart

Da die Wechsel in die Zustände bei einigen Auslösern von der Betriebsart abhängig sind, wird die Betriebsart in einem eigenen Subchart des Zustandsautomaten dargestellt, das auch im Anhang unter B.5 zu finden ist. Das Subchart ist mit dem Namen „Betriebsart“ gekennzeichnet. Bei den Betriebsarten ist ein Wechsel von jedem Zustand in jeden möglich, der sich nach der Variable „mode“ entsprechend anpasst. Deshalb ist für dieses Subchart kein fester Startzustand gewählt, sondern es erfolgt eine direkte Anpassung entsprechend

der Variable. Die Variable richtet sich nach dem Drehschalter der Simulation, mit dem die Betriebsart eingestellt werden kann. Es ist auch denkbar, die Betriebsart über die X12 Schnittstelle abzufragen. Dafür sind aber mindestens drei Ausgänge der Schnittstelle nötig, die bereits mit anderen Variablen belegt sind. Deshalb muss bei einem Wechsel der Betriebsart, die Änderung am Roboter und in der Visualisierung manuell erfolgen. Mit dem Drehschalter in der Visualisierung kann der Zustand nur im Automaten geändert werden, da eine solche Änderung in der Robotersteuerung mit einem Schlüssel freigegeben werden muss und keine Systemvariable für den Zweck als Eingang, sondern nur als Ausgang vorgesehen ist. Wird die Betriebsart geändert, während der Roboter ein Programm abfährt, wird ein Sicherheitshalt ausgelöst. Um einen Wechsel der Betriebsart während des Betriebs zu erkennen, ist die Variable „Betriebsartwechsel“ eingeführt worden.

Das Subchart Safety

Alle Sicherheitszustände werden im Subchart „Safety“ abgebildet. Einzusehen ist das Subchart in Anhang unter B.6. Sie entsprechen den Stopp-Reaktionen. Der enthaltene Automat verfügt über sieben Zustände. Angefangen mit dem allgemeinen Betriebszustand „IDLE“ und den jeweils drei Sicherheitsstopps und Sicherheitshalts, in die vom Auslöser abhängig gewechselt wird. Als Anfangszustand wird der Zustand „IDLE“ angenommen. Die Sicherheitszustände sind in einer sternförmigen Topologie um den Zustand aufgebaut. Das hat den Vorteil, dass die auslösenden Signale wieder rückgesetzt sein müssen, bevor in einen anderen Zustand gewechselt werden kann. Der Safety-Subchart ist das Herzstück des Zustandsautomaten. Hier werden die Signale aus allen Subcharts verwendet und zu einer Auswertung zusammengefügt.

Das Subchart Ursache

In einem weiteren Subchart „Ursache“ ist die Erkennung der Fehlerursache geregelt. Im Anhang ist das Subchart unter B.7 zu finden. Hier werden alle „Input“-Variablen der Chartumgebung berücksichtigt, um die Ursache für eine Stopp-Reaktion zu identifizieren. Die topologische Anordnung erfolgt ebenfalls wie in dem Subchart „Safety“ sternförmig. Zusätzlich zu den Fehlerursachen ist noch ein weiterer Zustand eingefügt, in dem zwar eine Stopp-Reaktion vorliegt, aber die Ursache nicht erkannt wurde. Für die Überprüfung des Zustands wird ein direktes „Feedback“-Signal von der Steuerung verwendet, das angibt, ob der Roboter gestoppt hat.

Das Subchart SafeOperation

In dem Subchart „SafeOperation“ wird der Zustandsautomat um die Auswertung der programmierbaren Arbeitsraum- und Geschwindigkeitsüberwachung erweitert. Dabei wird berücksichtigt, ob eine Überwachung aktiviert ist und eine Überschreitung der Arbeitsraumgrenze oder der maximalen Geschwindigkeit auftritt oder sie deaktiviert ist und eine Überschreitung der Grenzen keinen Effekt hat. In beiden Fällen wird ausgewertet, ob die eingestellte Raumgrenze oder die eingestellte maximale Geschwindigkeit bereits verlassen wurde und die Situation in einem Zustand gespeichert. Wird die Überwachung bei einer überschrittenen Grenze eingeschaltet, wird ein anderer Sicherheitshalt ausgelöst, als wenn sie bereits beim Überschreiten der Grenze aktiv war. Die Zuordnung des auszulösenden Sicherheitshalts ermittelt und mit der Variable „SafeOp“ an das Subchart „Safety“ übertragen. Der Aufbau des Subcharts „SafeOperation“ ist im Anhang unter B.8 abgebildet.

Variablenübersicht

In den Tabellen 4.3, 4.4 und 4.5 sind alle in der Chartumgebung definierten Variablen aufgelistet und ihre Funktionen kurz angedeutet. Die Variablen sind in die Typen „Local“, „Input“ und „Output“ eingeteilt. Die „Input“-Variablen werden in den Zustandsautomaten eingelesen und können abgefragt, aber nicht innerhalb des Automaten geändert werden. Die lokalen Variablen des Typs „Local“ sind für die Kommunikation innerhalb der Chartumgebung zuständig. Dabei können sie sowohl beschrieben, als auch gelesen werden. Dafür sind die „Output“-Variablen nur beschreibbar und können nicht innerhalb der Chartumgebung abgefragt werden. Sie dienen dazu, Informationen aus dem Zustandsautomaten in das Simulink-Modell herauszuführen. Sie werden für die Visualisierung der Zustände und Ursachen verwendet.

Tabelle 4.3: Chart-Variablenliste alphabetisch sortiert

Variable	Typ	Funktion
antriebe_aus	Input	Prüft, ob die Antriebe eingeschaltet sind
Aus_Raum	Local	Gibt an, ob der Manipulator sich außerhalb des programmierten Arbeitsraums befindet
Bedienerschutz	Input	Wertet das von den Lichtschranken gesetzte Bedienerschutzsignal aus
betrieb_an	Input	Prüft, ob der Roboter in Betrieb ist. Wird für ein Stopp durch den Betriebsartwechsel benötigt
betriebsartwechsel	Local	Variable, um einen Betriebsartwechsel zu erkennen. Kann als Ursache für einen Stopp erkannt werden
erweiterter_bedienerschutz	local	Die Variable erweitert das Bedienerschutzsignal um die Sicherheitsperipherien
Fehlerursache	Output	Die Variable gibt die Fehlerursache für die Visualisierung als Dezimalzahl aus
Forderung_Quittierung	Output	Erlaubt in der Visualisierung eine nötige Quittierung anzuzeigen
Geschw_nOK	Local	Ermöglicht beim Einschalten der Sicherheitsüberwachung das Auswerten, ob die maximal eingestellte Geschwindigkeit bereits überschritten ist
Geschw_override	Input	Simuliert ein Überschreiten der maximal eingestellten Geschwindigkeit oder in T1 der maximal erlaubten Geschwindigkeit
interner_fehler1	Input	Simuliert einen internen Fehler der Kategorie 1
interner_fehler2	Input	Simuliert einen internen Fehler der Kategorie 2
keine_Fahrfreigabe	Input	Stoppt den Roboter, wenn die Fahrfreigabe aufgehoben wird
Kontaktmatte	Input	Simuliert eine angeschlossene Kontaktmatte
mode	Input	Hier wird dem Automaten bekannt gemacht, welche Betriebsart vorliegt

Tabelle 4.4: Chart-Variablenliste alphabetisch sortiert - Fortsetzung

Variable	Typ	Funktion
nothalt	Input	Prüft, ob am SmartPAD oder ein externer Nothalt ausgelöst wurde
Nothalt_Intern	Input	Prüft, ob am SmartPAD ein Nothalt ausgelöst wurde. Wird für die Unterscheidung des externen Notausschalters und des Schalters am SmartPAD benötigt
Peri_Signal	Input	Das Peri-Signal wird gesetzt, wenn alle nötigen Bedingungen erfüllt sind. Wird hier nur als optische Anzeige ausgewertet und nicht im Automaten abgefragt
Quittierung_angefordert	Local	Wird gesetzt, wenn der Bedienerschutz auslöst. Wird durch Quittierung_BS zurückgesetzt
Quittierung_BS	Input	Hiermit bestätigt der Bediener, dass er und kein anderer sich mehr im Gefahrenbereich aufhält und der Betrieb wieder aufgenommen werden kann
Quittierung_Meldung	Input	Quittiert den erkannten Zustand und erlaubt die Wiederaufnahme des Betriebszustandes bei behobener Ursache
Raum_verletzt	Input	Wird gesetzt, wenn der Manipulator den programmierten Arbeitsraum verlässt
Safeop	Local	Wenn die Überwachung aktiviert ist und die programmierten Räume verlassen werden oder die Geschwindigkeit überschritten wird, dann löst die Variable einen Sicherheitshalt aus
SafetyEye	Input	Simuliert ein Halt durch das Safety Eye
SHS1	Input	Direktes Auslösen des Sicherheitshalt1
SHS2	Input	Direktes Auslösen des Sicherheitshalt2
Sicherer_Betriebshalt	Input	Direktes Auslösen des sicheren Betriebshalts mit STOP0
sicherheitsfehler	Input	Prüft, ob ein Sicherheitsfehler einen Stopp ausgelöst hat
spannung_aus	Input	Prüft, ob ein Stopp durch Einfallen der Spannungsversorgung ausgelöst wurde

Tabelle 4.5: Chart-Variablenliste alphabetisch sortiert - weitere Fortsetzung

Variable	Typ	Funktion
starttaste_geloest	Input	Simuliert ein Lösen der Starttaste auf dem SmartPAD im T1- oder T2-Betrieb
stop	Output	Gibt den Stoppzustand für die Visualisierung als Dezimalzahl aus
Stopp_gemessen	Input	Wenn ein Stopp in der Steuerung ausgelöst wird, wird das Signal gesetzt. Der Automat prüft, ob der Stopp erkannt wurde
stopptaste_druecken	Input	Simuliert ein Drücken der Stopptaste auf dem SmartPAD
Stoppzustand_erkannt	Input	Wird zur Überprüfung benötigt, ob ein Stoppzustand erkannt wurde
Ueberwachung_aktiv	Input	Wenn die Überwachung aktiviert ist, werden programmierte Räume und Geschwindigkeit überwacht
view_mode	Output	Gibt die Betriebsart für die Visualisierung aus
zustimmung_durchgedrueckt	Input	Erkennt ein Durchdrücken der Zustimmschalter im T1- oder T2-Betrieb
zustimmung_geloest	Input	Erkennt ein Lösen der Zustimmschalter im T1- oder T2-Betrieb

4.3 ControlDesk Visualisierung

Die Visualisierung erfolgt mit der Software ControlDesk. Die gesetzten Signale werden in grüner Farbe und rückgesetzte Signale in blauer Farbe angezeigt. Um alle nötigen Funktionen unterzubringen, wurden verschiedene Layouts erstellt. Das eine der beiden Layouts ist für die Simulation gedacht und wird in Abbildung 4.8 angezeigt. Hier können alle Ein- und Ausgangsvariablen des Automaten geändert und die Reaktionen des Automaten getestet werden. Das andere Layout stellt die GUI der realen Schnittstellen da und ist in Abbildung 4.9 gezeigt. Von hier aus kann der Bediener die Zustände des Industrieroboters einsehen, den Grund für einen ausgelösten Sicherheitshalt oder Sicherheitsstopp erkennen, die Schalterstellungen einsehen und mit simulierten Schaltern diverse Variablen manipulieren.

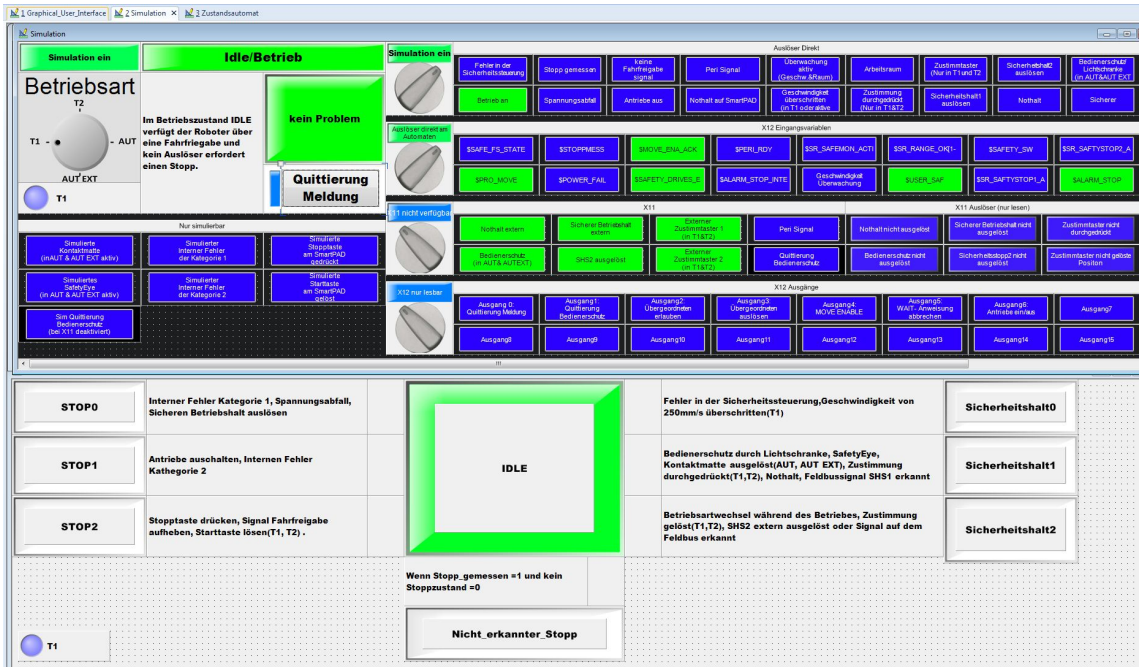


Abbildung 4.8: Visualisierung der Simulation

In der linken oberen Ecke wird in beiden Layouts angezeigt, ob das simulierte oder das GUI der Schnittstellen aktiv ist. Zum Wechsel zwischen der Simulation und der realen Schnittstelle muss der entsprechende Drehschalter bedient werden. Beim Starten des Programms ist standardmäßig die Simulation aktiv eingestellt. Die Funktionen lassen sich immer durch Drehschalter hinzufügen oder entfernen. Über einen solchen Drehschalter wird auch die X11 Schnittstelle und die Option zum Senden über die X12 Schnittstelle dazugeschaltet. Wobei in der Startkonfiguration beide Optionen ausgeschaltet sind. Des weiteren lässt sich über einen solchen Drehschalter in der Simulation auswählen, ob die Auslöser direkt am Automaten getestet werden können oder die Eingänge der Schnittstelle mit den Logiken und invertierten

Signalen simuliert wird. Die An- und Abwahlen von Signalen erfolgen direkt über die Schalter der Signalnamen. Dabei gilt, dass in der Simulation alle Signale durch das An- und Abwählen mit der linken Maustaste geändert werden und in den Schnittstellen GUI nur ausgelesen werden können und daher nicht änderbar sind. Ausnahmen von der Regel sind die dauerhaft simulierten Signale und die Eingänge der X12 Schnittstelle. Sie können in beiden Layouts geändert werden. Ebenso befinden sich bei der X11 Schnittstelle in beiden Layouts LEDs, die den Wert der auslösenden Signale anzeigen. Sie können in beiden Layouts nicht direkt geändert werden, sondern werden über die links von ihrer Position liegenden Eingänge geändert.

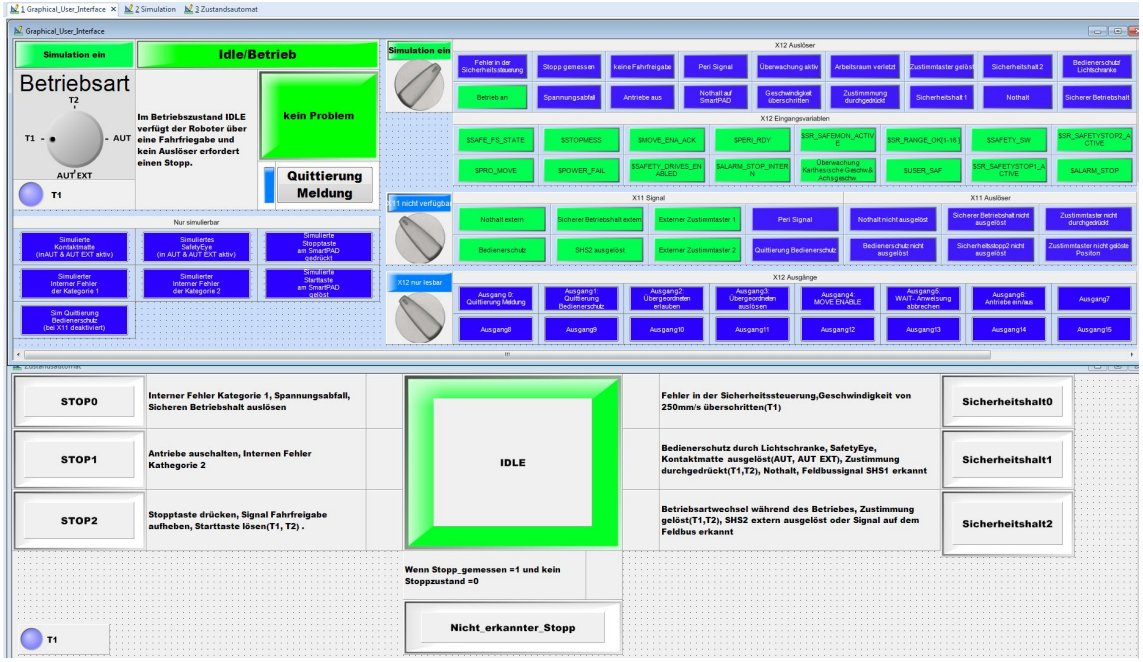


Abbildung 4.9: Graphical User Interface

Unter beiden Layouts, die zur Unterscheidung mit unterschiedlichen Hintergrundfarben gekennzeichnet sind, ist eine Visualisierung des Zustandsautomaten dargestellt. Die Anordnung unter den beiden Layouts wurde gewählt, damit der Zustandsautomat während der Bedienung der Layouts verfolgt werden kann. Falls gewünscht, können die Positionen auch getauscht werden, so dass sich der Automat in der oberen und die Layouts für die Interaktion, in der unteren Bildschirmhälfte angeordnet werden. Der aktive Zustand wird über einen grünen Rahmen markiert und zwischen den Zuständen die Handlungsoptionen des Bedieners angezeigt. Wenn eine Stopp-Reaktion ausgelöst wurde, wird ebenfalls das Symbol der auslösenden Ursache eingeblendet. Dadurch wird das Beheben der Ursache erleichtert. Bei einem Zustandswechsel muss eine Bestätigung mit „Quittierung Meldung“ erfolgen. Wenn

eine solche Bestätigung oder die Quittierung des Bedienerschutzes erforderlich ist, wird der zu betätigende Schalter mit einem roten Rahmen hinterlegt.

4.4 Testen

In diesem Abschnitt werden die realisierten Verbindungen zwischen der Platine, dem Zustandsautomaten und der Visualisierung geprüft und einige Funktionen getestet.

Der „External“- Mode in Simulink wird von dSPACE seit dem Release 6.3 nicht mehr unterstützt [1, S.59]. Daher müssen alle Tests mit der Visualisierung in ControlDesk durchführbar sein und können nicht direkt in Simulink getestet werden. Sollte nur die Funktionsweise des Zustandsautomaten in Simulink getestet werden, ist unbedingt darauf zu achten, dass in der Konstante „Simulation“ eine 1 eingetragen ist. Mit der Einstellung ist die Simulation aktiviert und der Automat ist nicht mehr von den Eingängen der Hardware abhängig. Die Simulationslaufzeit sollte hierfür als unendlich eingestellt werden und gegebenenfalls müssen die dSPACE Bausteine auskommentiert werden.

Für einen Test der Übersetzungsplatine wird die „female“-Variante des 50-poligen SUB-D-Steckers der Platine mit der CP30 Schnittstelle des dSPACE Echtzeitboards verbunden. Die Übersetzungsplatine benötigt einen Generator für die Einspeisung von 24V und zum Anschließen von GND. Damit die Signale aus der Schnittstelle des Echtzeitboards eingestellt werden können, muss das Simulink Programm auf das Echtzeitboard geladen sein und das ControlDesk-Programm mit deaktivierter Simulation ausgeführt werden. Durch Betätigen der X12 Eingänge in der Visualisierung kann wahlweise ein „Low“- oder „High“-Signal ausgegeben werden. Mit einem Multimeter wird die Spannung an dem anderen Ende der Platine für die „dSPACE-X12“-Übersetzung kontrolliert. Die korrekte Funktion ist nachgewiesen, wenn die Werte wie in Tabelle 4.6 vorliegen.

Tabelle 4.6: Verbindungstest Platine - Für die Ausgänge des Echtzeitboards

X12 Eingänge in ControlDesk	Multimeter-Anzeige X12 Seite
0 (BOOL) blau	0,16 Volt (ca.0 Volt)
1 (BOOL) grün	23,25 Volt (ca.24 Volt)

Um auch die „X12-dSPACE“-Übersetzung zu testen, wird der gleiche Aufbau verwendet, außer dass noch ein freies Kabel an den GND Anschluss des Generators angeschlossen wird. Die einzelnen Pins der Platine werden für den Test mit dem Massekabel des Generators kurz verbunden und die Reaktion an den LEDs für die X12 Ausgänge in der Visualisierung beobachtet. Die Reaktionen sehen bei korrekter Funktionsweise wie in Tabelle 4.7 aus.

Tabelle 4.7: Verbindungstest Platine - Für die Eingänge des Echtzeitboards

X12 Ausgänge in ControlDesk	Kabel des Generators
0 (BOOL) blau	0 Volt (GND)
1 (BOOL) grün	24 Volt

Zum Testen der Roboterprogramme kann mit dem Programm Office Lite V8.3.17 in Verbindung mit KUKA Sim Pro 2.2.2 ein virtueller Roboter mit einer virtuellen Steuerung und SmartPAD in Betrieb genommen werden. Die Programme für Bewegungen, die im Roboter Interpreter ausgeführt werden, sind in dieser Entwicklungsumgebung gut zu simulieren und zu testen. Die Änderungen in WorkVisual an einer Projektdatei lassen sich zwar auf die virtuelle Steuerung schieben, jedoch sind EA-Verschaltungen im Beckhoff Buskoppler und auch die Programmierungen im Submit-Interpreter für die X12-Schnittstelle ohne Auswirkungen. Eine solche Projektdatei muss auf eine reale Steuerung geschoben werden. Falls das Projekt fehlerhaft ist, lässt sich das ursprünglich auf der Steuerung liegende fixierte Projekt wieder in der Projektverwaltung herstellen. Zum Testen der im Projekt konfigurierten X12-Ausgänge muss der Submit-Interpreter angewählt und gestartet sein. Der gestartete Submit-Interpreter lässt sich an dem grün hinterlegten „S“ auf dem SmartPAD, wie in Abbildung 4.10 zu sehen, erkennen. Eine anliegende Spannung von etwa 26 Volt kann mit einem Multimeter gegen Masse gemessen werden, wenn der Ausgang auf „TRUE“ gesetzt wurde. Alle konfigurierten Ausgänge sollten nach Möglichkeit einmal mit einem auf „TRUE“ gesetzten Signal getestet werden, da bei auf „FALSE“ rückgesetzten Ausgängen, genauso wie im nicht konfigurierten Zustand, keine Spannung abfällt.

Abbildung 4.10: Kontrollanzeige auf dem SmartPAD

Sollte auch bei einem gesetzten „TRUE“ Signal keine Spannung anliegen, sollte die E/A-Verschaltung in WorkVisual nochmal überprüft werden. Ist sie korrekt verschaltet, muss die Ursache an der Zuweisung der Variablen zu einem digitalen Ausgang liegen. Die Werte der digitalen Ausgänge können auf dem SmartPAD ausgegeben werden. Dafür muss die Anzeige im Hauptmenü, wie in Abbbildung 4.10 verdeutlicht, geöffnet werden. In der Anzeige entspricht eine grüne LED einem auf „TRUE“ gesetzten Signal.

4.5 Inbetriebnahme

Für die Inbetriebnahme des Zustandsautomaten muss das Simulinkmodel „Zustandsautomat“ auf das Echtzeitboard geladen sein. Das ist in Simulink mit dem Befehl „Bulid“ möglich. Dazu muss das Echtzeitboard angeschlossen und eingeschaltet sein. Ebenso darf der Zugriff nicht von einem anderen Programm blockiert werden. Die Steuerung, die durch den Zustandsautomaten emuliert werden soll, muss die Projektdatei „AGILUS_Moritz_SafetyMod_X12.wvs“ mit den Konfigurationen für die X12 Schnittstelle geladen haben. Für eine andere Steuerung, als die zur Konfiguration verwendete, muss eine andere Projektdatei erzeugt oder bearbeitet werden, da die eingestellten Adressen in der Steuerung nur für die im Projekt verwendete Steuerung gültig sind. Hierzu müssen die Änderungen für die SPS.SUB, die „machine.dat“ und der EA-Konfiguration aus den vorigen Kapiteln erneut vorgenommen werden. Sollte die Projektdatei nicht auf die Steuerung geladen sein, kann sie mit dem Programm WorkVisual geöffnet und mit dem Befehl „Installieren“ auf die Steuerung übertragen werden. Dazu muss die Steuerung eingeschaltet sein und sich im gleichen Netzwerk befinden. Eine Übertragung der Datei muss an dem SmartPAD vom Bediener quittiert werden. Dazu muss im Vorwege eine Benutzergruppe mit genügend Rechten ausgewählt sein. Die Änderung der Benutzergruppe ist im Hauptmenü unter „Konfiguration\Benutzergruppe“ einstellbar. Mögliche Benutzergruppen für die Installation sind „Experte“ oder „Systeminbetriebnehmer“. Anschließend muss auf dem SmartPAD der Submit-Interpreter angewählt und gestartet werden, um die Programmierung in der SPS.SUB zu aktivieren. Sind alle Projekte auf der Hardware installiert und ist die Übersetzungsplatine mit der CP30 Schnittstelle auf dem Echtzeitboard, der Spannungsversorgung und der X12 Schnittstelle an der Robotersteuerung verbunden, kann in ControlDesk die Visualisierung „Zustandsautomat_Visualisierung_V01“ geöffnet und mit „Go online“ gestartet werden. Hierzu muss sich der Lizenz-Dongle im Computer befinden.

5 Fazit

5.1 Schlussbetrachtung

In der Bachelorarbeit wurde ein Zustandsautomat mit Simulink Sateflow realisiert, der die Sicherheitszustände der KR C4 compact Robotersteuerung aufzeigt. Die Sicherheitszustände sind Stopp-Reaktionen, die als Folgen von auslösenden Signalen eingenommen werden.

Die Signale, die der Zustandsautomat dafür benötigt, wurden durch die Entwicklung einer elektrischen Verbindung und programmiertechnische Einstellungen aus der Steuerung bereitgestellt. Dazu wurden die Spezifikationen der Schnittstellen ausgewertet und für die Verbindung der CP30 mit der X12 Schnittstelle eine Übersetzungsplatine mit Optokopplern entwickelt, die die Spannungspegel der Schnittstellen anpasst. Damit die Signale auf der X12 Schnittstelle verfügbar sind, wurden die nötigen Systemvariablen, die nicht bereits in der „machine.dat“-Datei digitalen Ausgängen zugewiesen sind, in der „SPS.SUB“ ausgewertet und ebenfalls digitalen Ausgängen zugeordnet. Die Pinbelegung konnte daraufhin in WorkVisual unter der E/A-Verschaltung verbunden werden.

Für die Inbetriebnahme des Zustandsautomaten wurden zwei Oberflächenlayouts in dem Programm ControlDesk erzeugt. Zum einen kann der Zustandsautomat in Verbindung mit der Hardware, aber zum anderen auch als Simulation implementiert werden. Nicht als Signale auswertbare Auslöser mussten in beiden Oberflächenlayouts als simulierte Schalter dargestellt werden. Dabei wird der Zustandsautomat auf beiden Oberflächenlayouts visualisiert und die Handlungsoptionen für die Bedienung angezeigt.

5.2 Weiteres Vorgehen und Ausblick

Als weiteres Vorgehen empfiehlt es sich, die „machine.dat“ Datei anzupassen, um mit der X12 Schnittstelle nicht nur Daten auslesen zu können, sondern auch Signale in die Schnittstelle einzuspeisen. Durch die Anpassung wird die Benutzung der Schalter in der Visualisierung für die X12 Eingänge und die Benutzung der entsprechenden Simulink Blöcke ermöglicht. Eine mögliche Anpassung der Datei könnte wie folgt aussehen.

```
;Neu zugeordnete Signale:
SIGNAL $CONF_MESS $IN[100]
;externe Quittierung von Fehlermeldungen
SIGNAL $IMM_STOP $IN[101]
;(extern Nothalt)
SIGNAL $SAFEGATE_OP $IN[102]
;(Nothalt extern STOP1)
SIGNAL $MOVE_ENABLE $IN[103]
; Fahrfreigabe
SIGNAL $LOOP_CONT $IN[104]
;unterbricht WAIT Anweisung
SIGNAL $DRIVES_ENABLE $IN[105]
;Antriebe einschalten
;SIGNAL $MOVE_ENA_ACK $OUT[150]
;nur wenn nicht $SPOC_MOTION_ENABLE verwendet wird
```

Ebenso können weitere Variablen für den Zugriff auf die Steuerung in Erfahrung gebracht werden und den noch nicht belegten Schaltern in der Visualisierung zugeordnet werden. Wenn Variablen benutzt werden sollen, die in der „machine.dat“ einem FALSE zugeordnet sind, muss diese Zeile vorher mit einem Semikolon auskommentiert werden, da das FALSE hier keinem Wert entspricht, sondern die Signale stilllegt [16, S.133]. Genauso können die gewählten Ausgänge durch andere Variablen ersetzt werden, falls andere Informationen aus der Steuerung bezogen werden sollen.

Es empfiehlt sich auf jeden Fall eine weitere Übersetzungsplatine mit der gleichen Schaltung, die bereits für die „X12-dSPACE“-Übersetzung verwendet wird, zu erstellen, um die X11 Schnittstelle auslesen zu können. Dazu genügt es, einen 37-poligen Sub-D male Stecker an der Platine mit der CP31 Schnittstelle des Echtzeitboards zu verbinden. Die andere Seite der Platine muss dann mit den entsprechenden Signalen im Schaltkasten der X11 Schnittstelle verkabelt werden.

Außerdem besteht die Möglichkeit, eine SPS an die Steuerung anzuschließen. Dadurch ergeben sich für die Steuerung neue Möglichkeiten und die Anzahl der Funktionen mit der Steuerung wird erweitert. Weitere Funktionen sollten auch mit dem Auslesen der Ethernet Sicherheitsschnittstelle X66 auswertbar sein. Jedoch ist die nicht gleichzeitig mit der X11 Schnittstelle verwendbar und wird im aktuellen Laboraufbau für die Verbindung mit Sim Pro 2.2.2 verwendet. Über die X66 Schnittstelle werden auch Sicherheitsdaten übertragen. Allerdings enthalten die Daten mehr Informationen. Dadurch können beispielsweise alle Achsen bei der Datenabfrage einzeln ausgewertet werden. Es wäre möglich auszugeben, welche der Achsen die maximal eingestellte Geschwindigkeit überschritten hat. Genauso könnte

der programmierte Arbeitsraum in 14 Melderäume unterteilt werden und in der Visualisierung würde angezeigt werden können, welcher der Melderäume vom Roboter verlassen wurde.

Bei der Inbetriebnahme des Roboters kann eine SAK-Fahrt, ein Bremstest oder eine Justage-Referenzierung vom System gefordert werden. Die Fahrten sind keine direkten Sicherheitszustände, aber wenn mehr Daten aus dem Roboter ausgelesen werden können, kann der Zustandsautomat und die Visualisierung um die Emulation der Fahrten erweitert werden. Da die KR C4 compact Steuerung keinen Anschluss für einen Referenzierungsschalter besitzt, könnte ein solcher über die X12 Schnittstelle und der Visualisierung hinzugefügt werden.

Eine Auswertung der internen Fehler könnte durch das Auswerten von „ERROR_T“-Datentypen im Submit-Interpreter ermöglicht werden. Durch welchen Fehler, welche Stopp-Reaktion hervorrufen wird, könnte eventuell empirisch ermittelt werden.

Bei dem Zustandsautomaten wird jeder Zustandswechsel mit der selben Ausgabevariablen ausgegeben und muss in der Visualisierung bestätigt werden. Das ist in der Visualisierung auch nötig, um die Zustände in der selben Anzeige darzustellen. Dagegen kann in der aktuellen Visualisierung hierdurch nur eine Meldung zur Zeit ausgegeben werden. Damit mehrere Meldungen angezeigt werden können und ein Zustandswechsel auch ohne die Meldung zu quittieren stattfinden kann, müsste jeder Zustand eine eigene Ausgabevariable zugewiesen bekommen. Die darf dann nicht direkt ausgewertet werden, sondern müsste in Matlab zwischengespeichert werden. Die gespeicherten Werte könnten dann für die Meldungen benutzt und einzeln bestätigt werden.

Literaturverzeichnis

[1] dSPACE GmbH: *Release 6.3 - New Feature and Migration*. http://www.dspace.de/ftp/patches/NFMG/RELEASE63/NewFeaturesAndMigration.pdf. Version: 2008

[2] dSPACE GmbH: *DS1103 PPC Controller Board - Hardware Installation and Configuration*. Release 2015-A. Rathenaustraße 26, D-33102 Paderborn, 2015

[3] dSPACE GmbH: *dSPACE Website - Compatibility Tables*. https://www.dspace.com/de/gmb/home/support/supvers/supverscompm.cfm. Version: 2016

[4] dSPACE GmbH: *dSPACE Website - Connector und LED Panels*. https://www.dspace.com/de/gmb/home/products/hw/singbord/conledpanels.cfm. Version: 2016

[5] dSPACE GmbH: *dSPACE Website - VEOS Plattform für PC-basierte Simulation von Modellen*. https://www.dspace.com/de/gmb/home/products/sw/simulation_software/offline_simulator.cfm. Version: 2016

[6] KUKA Roboter GmbH: *Arbeitsbereichbegrenzung A1 - Für Produktfamilie KR AGILUS - Montage- und Betriebsanleitung*. Option KR AGILUS Working Range Limitation A1 V1 de. Zugspitzstraße 140, D-86165 Augsburg, 2012

[7] KUKA Roboter GmbH: *Einsatz und Programmierung von Industrierobotern*. Edu Pack Einsatz und Programmierung von Industrierobotern V4 de. Zugspitzstraße 140, D-86165 Augsburg, 2013

[8] KUKA Roboter GmbH: *KR AGILUS sixx - Mit W- und C-Variante Betriebsanleitung*. BA KR AGILUS sixx V11. Zugspitzstraße 140, D-86165 Augsburg, 2014

[9] KUKA Roboter GmbH: *KR C4 compact - Betriebsanleitung*. BA KR C4 compact V8. Zugspitzstraße 140, D-86165 Augsburg, 2014

[10] KUKA Roboter GmbH: *KR C4 compact - Montageanleitung*. MA KR C4 compact V8. Zugspitzstraße 140, D-86165 Augsburg, 2014

[11] KUKA Roboter GmbH: *KUKA System Software 8.3 - Bedien- und Programmieranleitung für Endanwender*. KSS 8.3 END V4. Zugspitzstraße 140, D-86165 Augsburg, 2014

[12] KUKA ROBOTER GMBH: *KUKA System Software 8.3 - Bedien- und Programmieranleitung für Systemintegratoren*. KSS 8.3 SI V4. Zugspitzstraße 140, D-86165 Augsburg, 2014

[13] KUKA ROBOTER GMBH: *KUKA.SafeOperation 3.2 - Für KUKA System Software 8.3*. KST SafeOperation 3.2 V4. Zugspitzstraße 140, D-86165 Augsburg, 2014

[14] KUKA ROBOTER GMBH: *Optionale Schnittstellen - Für KR C4 compact Montage- und Betriebsanleitung*. MA KR C4 compact Interfaces V6. Zugspitzstraße 140, D-86165 Augsburg, 2014

[15] KUKA ROBOTER GMBH: *Sicherheit Industrierobotern mit KR C4*. Safety KR C4 V12. Zugspitzstraße 140, D-86165 Augsburg, 2015

[16] KUKA ROBOTER GMBH: *Systemvariablen - Für KUKA System Software 8.1, 8.2, 8.3 und 8.4*. KSS 8.1, 8.2, 8.3, 8.4 System Variables V1. Zugspitzstraße 140, D-86165 Augsburg, 2015

[17] KUKA ROBOTER GMBH: *WorkVisual 3.1 - Für KUKA System Software 8.3 und 8.2*. KST WorkVisual 3.1 V4. Zugspitzstraße 140, D-86165 Augsburg, 2015

[18] KUKA ROBOTER GMBH: *KUKA System Software 8.3 - Bedien- und Programmieranleitung für Systemintegratoren*. KSS 8.3 SI V5. Zugspitzstraße 140, D-86165 Augsburg, 2016

[19] KUKA ROBOTER GMBH: *Optionale Schnittstellen - Für KR C4 compact Montage- und Betriebsanleitung*. MA KR C4 compact Interfaces V10. Zugspitzstraße 140, D-86165 Augsburg, 2016

[20] O.VERF.: *Logikfunktionen und SPS*. https://www.xplore-dna.net/mod/page/view.php?id=168. Version: 2016

[21] PILZ GMBH & CO.KG: *Sichere Sensorik PSEN*. 04 2016

[22] ROBOTATION ACADEMY GMBH: 5. Roboter Kongress. In: *Podiumsdiskussion: Gehört der Zusammenarbeit zwischen Roboter und Werkern die Zukunft?* Hannover Messe, 2016, S. 1

[23] SHARP ELECTRIONIC COMPONENTS: *Datasheet PC817 Series*. http://pdf1.alldatasheet.com/datasheet-pdf/view/43371/SHARP/PC817.html. Version: 2010

Teil I

Anhang

A Änderung in PLC der SPS.SUB

```
;FOLD USER PLC
;Make your modifications here
$OUT[100]=$SAFE_FS_STATE
;$OUT[1022]=$PRO_MOVE
;schreibgeschützt weil in Machine.dat
;$OUT[1010]=$STOPMESS
;schreibgeschützt weil in Machine.dat
$OUT[101]=$POWER_FAIL
;$OUT[150]=$MOVE_ENA_ACK ;als Alternative
;muss in machine.dat aus FALSE entfernt werden
;$OUT[1015]=$SPOC_MOTION_ENABLE
;schreibgeschützt weil in Machine.dat
$OUT[102]=$SAFETY_DRIVES_ENABLED
$OUT[1012]=$PERI_RDY
;$OUT [1002] = $ALARM_STOP_INTERN
;schreibgeschützt weil in Machine.dat
$OUT[103]=$SR_SAFEMON_ACTIVE

IF $SR_AXISSPEED_OK OR $SR_CARTSPEED_OK THEN
$OUT[104]=FALSE
ELSE
$OUT[104]=TRUE
ENDIF

IF $SR_RANGE_OK THEN
$OUT[105]=FALSE
ELSE
$OUT[105]=TRUE
ENDIF

$OUT[1011]=$USER_SAF
```

```
IF $SAFETY_SW == #RELEASED THEN
$OUT[106]=FALSE
ELSE
$OUT[106]=TRUE
ENDIF

$OUT [107] = $SR_SAFETYSTOP1_ACTIVE
$OUT [108] = $SR_SAFETYSTOP2_ACTIVE
;$OUT [1013] = $ALARM_STOP
;schreibgeschützt weil in Maschinen.dat

;weitere sinnvolle Variablen:
;$OUT [1000] = $IN_HOME
;Kommentar entfernen zum Benutzen
;$OUT [109] = $SR_SAFEOPSTOP_ACTIVE
;Kommentar entfernen zum Benutzen
;Eingänge müssen in machine.dat geändert werden
;ENDFOLD (USER PLC)
```

B Simulink-Modell

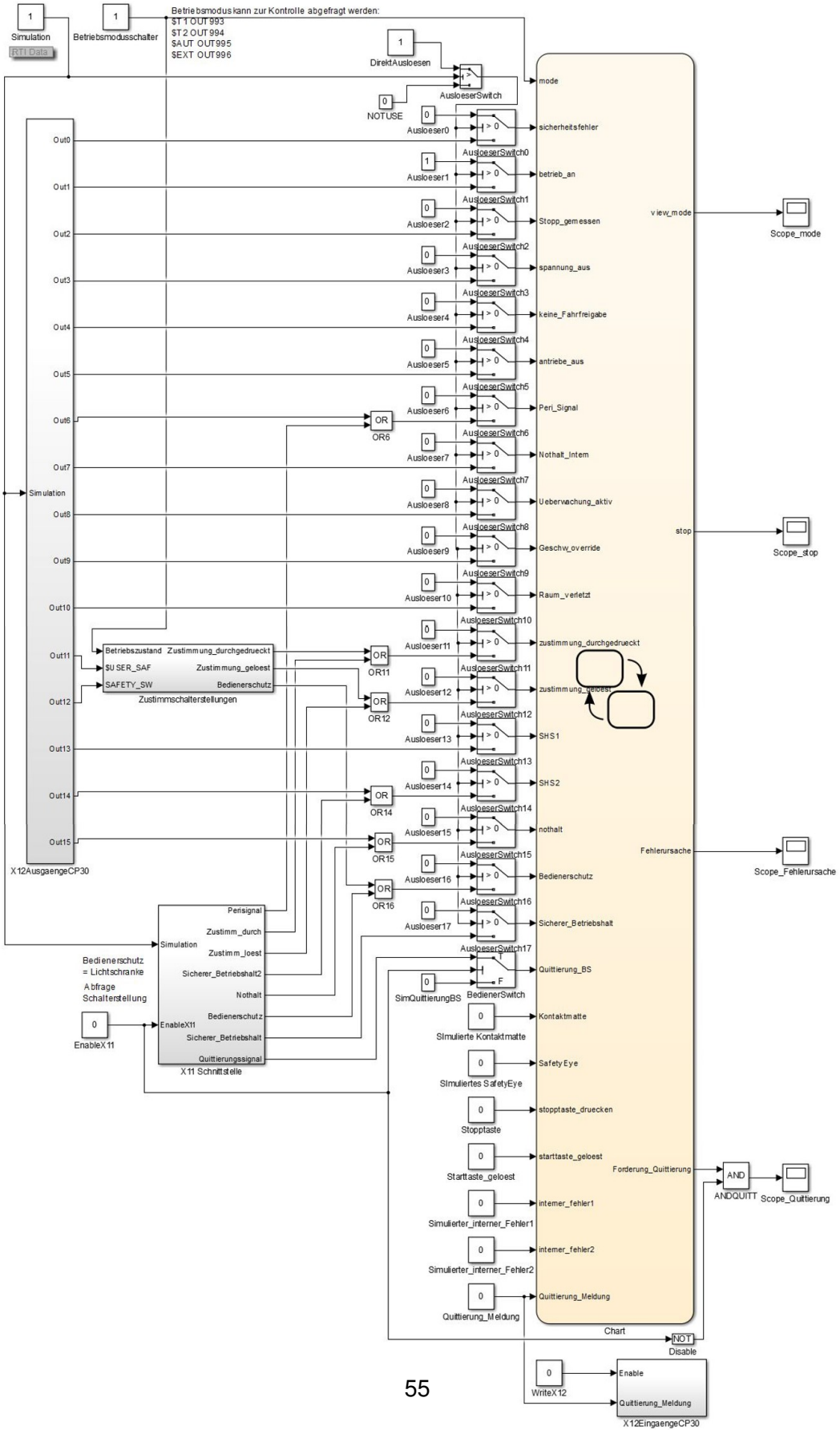


Abbildung B.1: Übersicht Simulink-Modell

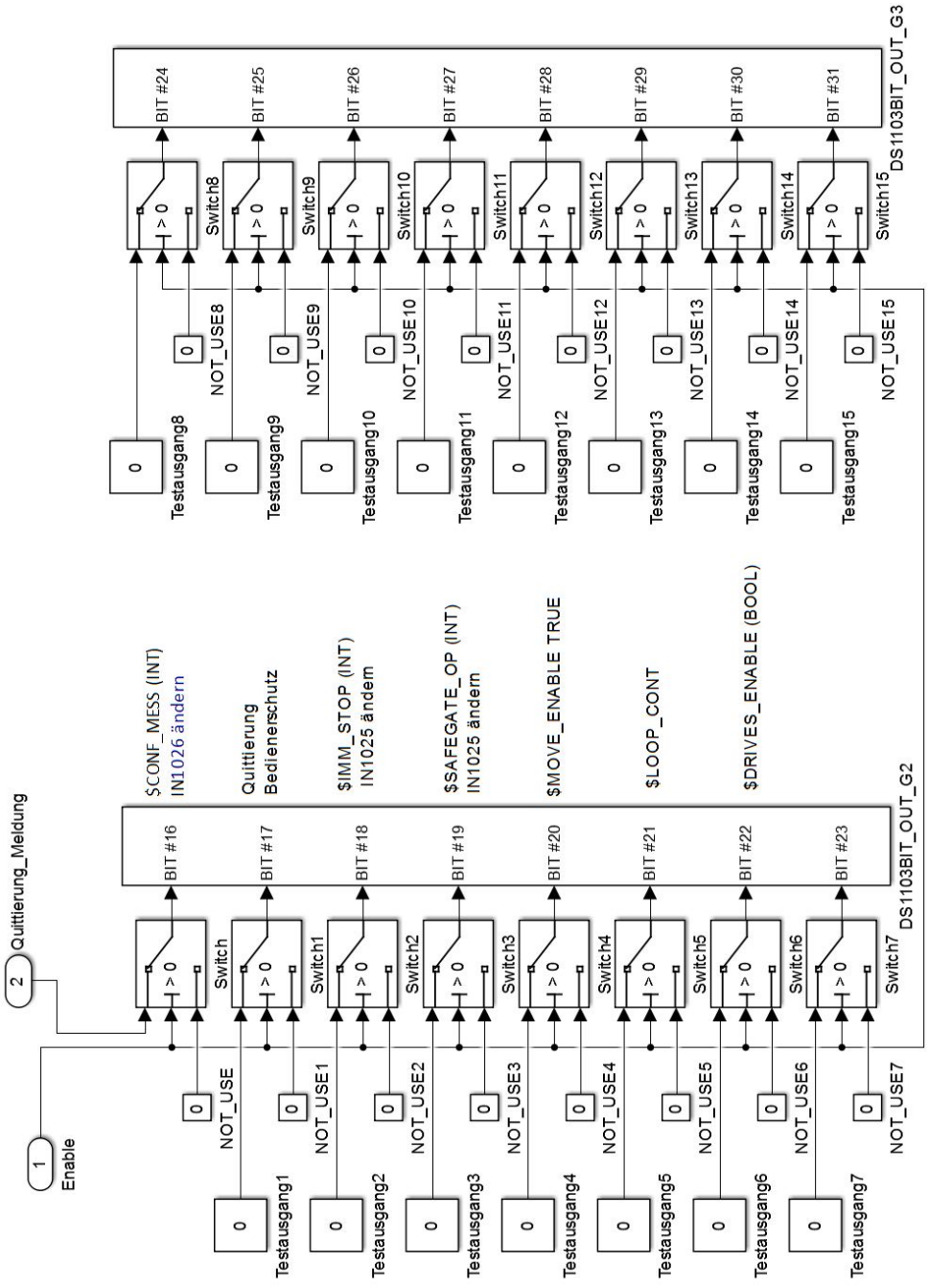


Abbildung B.2: Subsystem X12EingaengeCP30

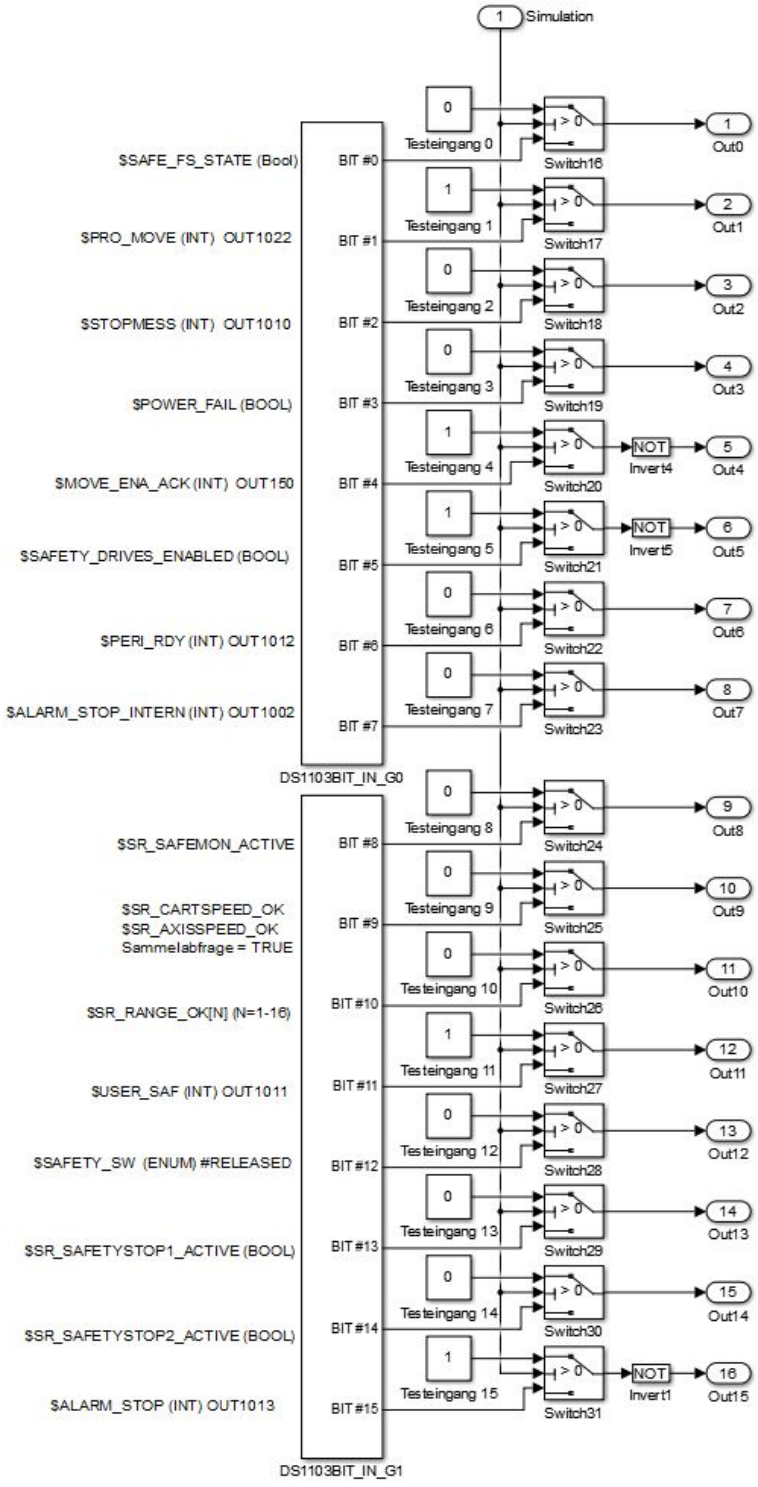


Abbildung B.3: Subsystem X12AusgaengeCP30

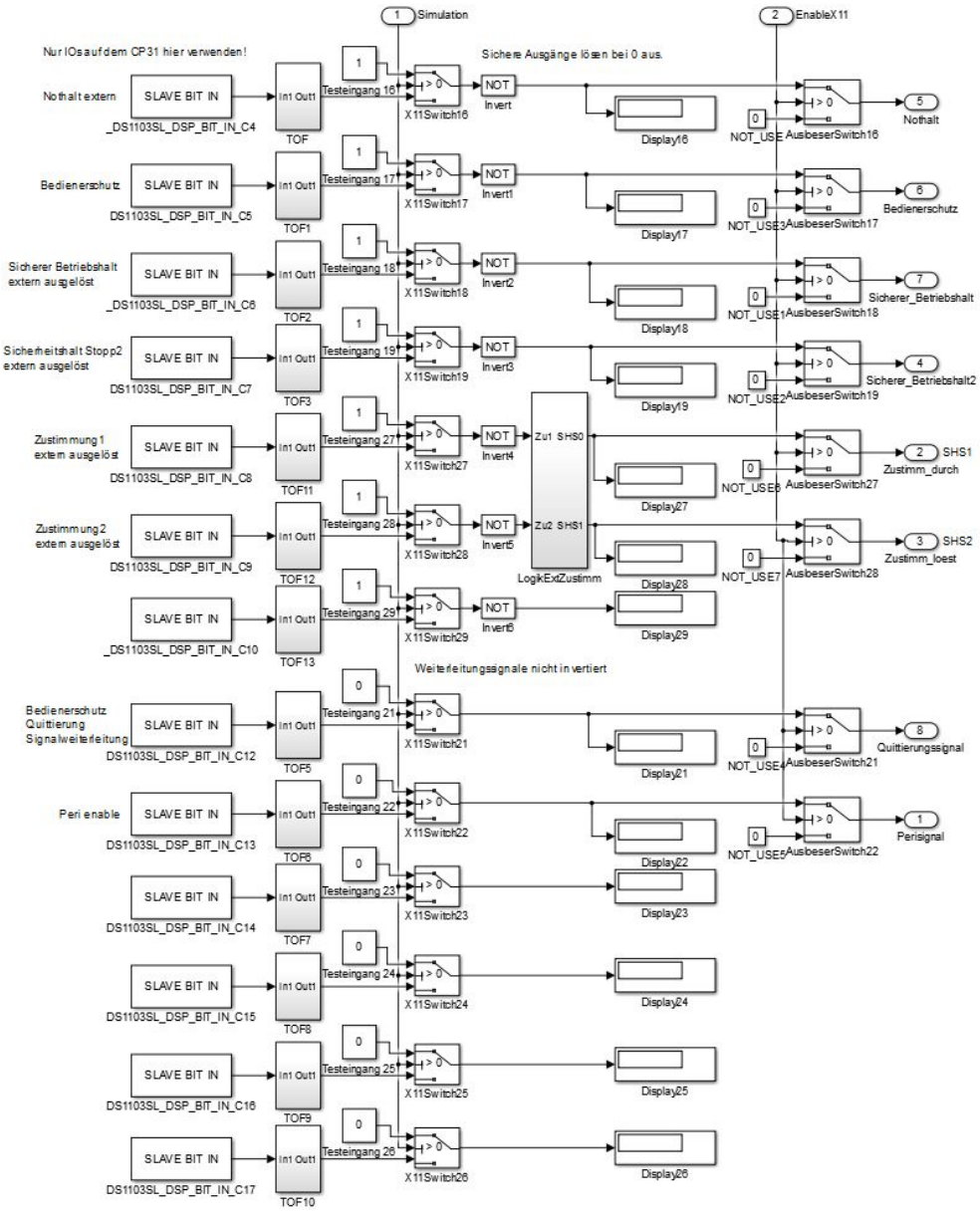


Abbildung B.4: Subsystem X11Schnittstelle

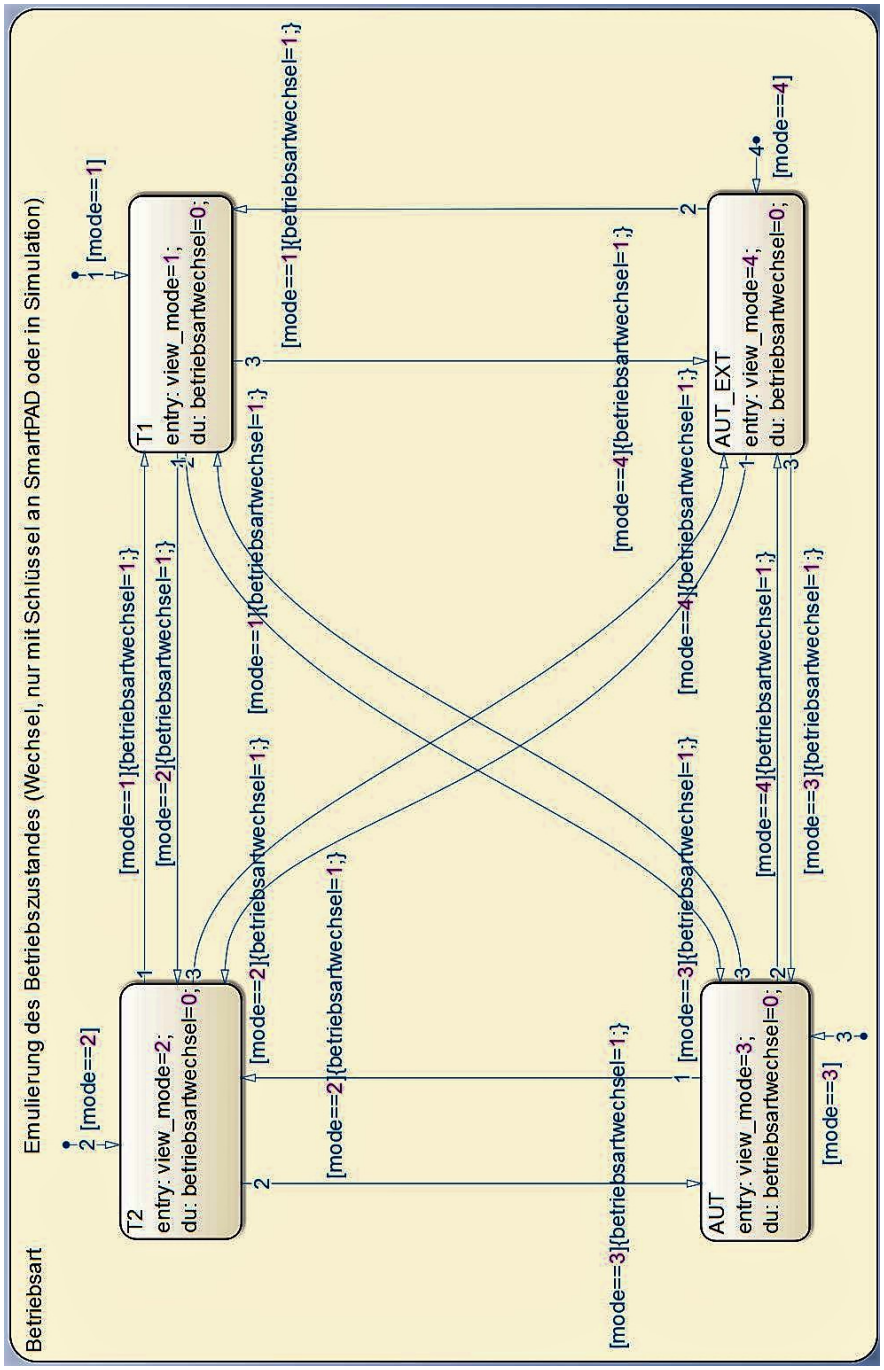


Abbildung B.5: Subchart Betriebsart

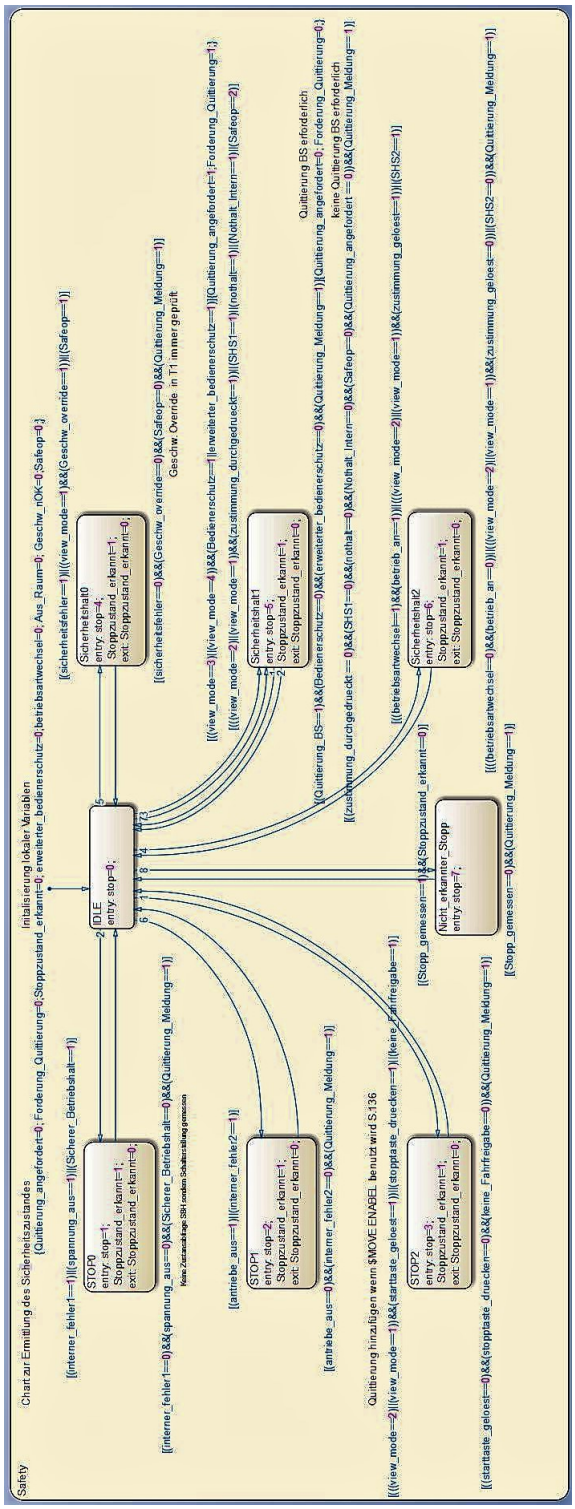


Abbildung B.6: Subchart Safety

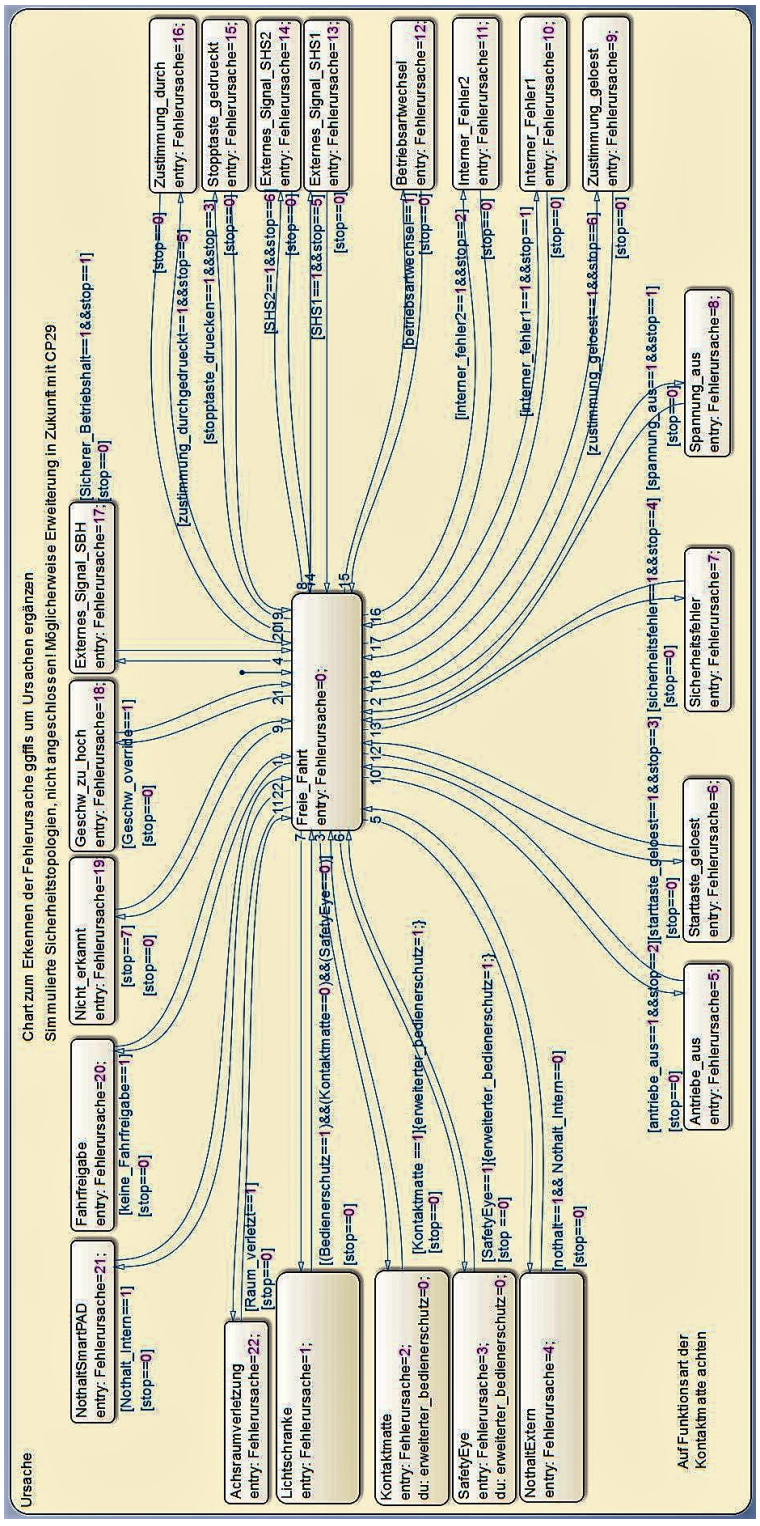


Abbildung B.7: Subchart Ursache

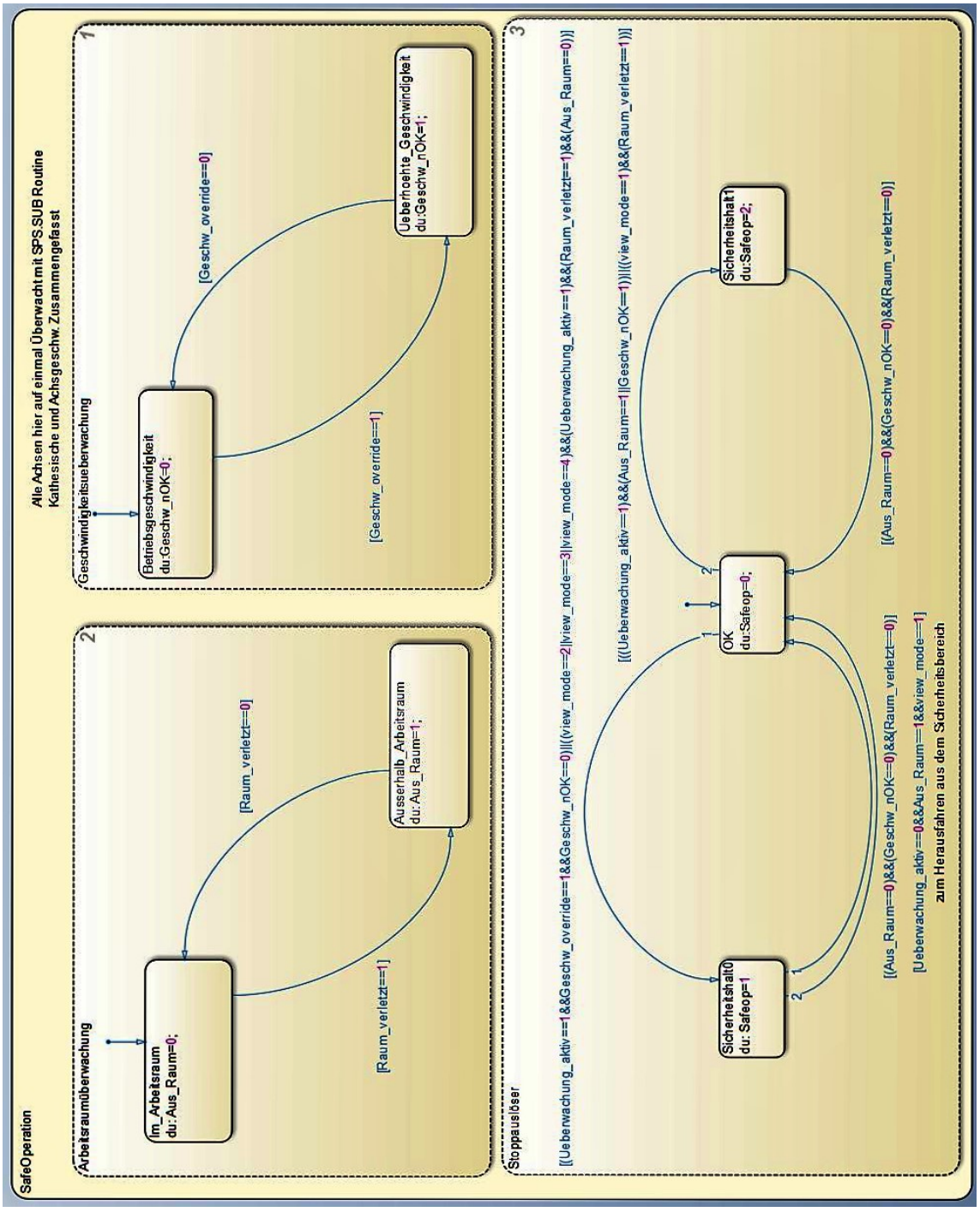


Abbildung B.8: Subchart SafeOperation

Versicherung über die Selbstständigkeit

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

Hamburg, 15. Juli 2016	
Ort, Datum	Unterschrift