Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften, Bereich EuI

Wissenschaftliche Projektarbeit

Virtuelle Inbetriebnahme eines CAD-basierten Simulink-Modells mit der OpenPLC-Soft-SPS

Eingereicht am: 17. Juli 2019

von: Jan Bartelt, B. Eng.
geboren am 12. April 1994
in Schwerin

Betreuer: Prof. Dr.-Ing. Thorsten Pawletta

Aufgabenstellung

Thema:

Untersuchung zur virtuellen Inbetriebnahme eines CAD-basierten Simulink-Modells mit der OpenPLC-Soft-SPS.

Schwerpunkte:

- Einarbeitung in die Toolkette
  - CAD-Umgebung
  - Simscape Multibody Link
  - Simulink PLC Coder
  - OpenPLC
- Entwicklung eines Demonstrationsbeispiels
  - Implementierung eines einfachen Prozessmodells
  - Implementierung einer einfachen Steuerung
  - virtuelle Inbetriebnahme
- Dokumentation der Modelle und Erstellen der schriftlichen Arbeit

Die Projektarbeit ist nach den Richtlinien einer wissenschaftlichen Arbeit zu verfassen („Mini-Bachelorarbeit“). Siehe als Beispiel das Dokument Bsp-einer-Projekt-Arbeit.pdf auf der WWW-Seite. Neben einem schriftlichen Exemplare ist die Arbeit (Originaldateien & PDF-Dateien) sowie die entwickelte Software auf einem dokumentierten digitalen Datenträger abzugeben.

Betreuer und Prüfer:	Prof. Dr. Thorsten Pawletta, FIW/MVU, FG CEA Hochschule Wismar
Abgabetermin:	17.06.2019

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit behandelt die Virtuelle Inbetriebnahme (VIBN) einer Maschine mit Matlab Simulink. Unter VIBN versteht man das Entwickeln, Testen und Validieren einer Maschinensteuerung, ohne die reale Maschine selbst. Stattdessen wird die Steuerung mithilfe einer virtuellen Maschine entwickelt.[3]

Maschinen werden gegenwärtig meist mithilfe einer 3D-CAD-Umgebung konstruiert. Dabei werden die einzelnen Bauteile erstellt und durch Informationen zu Material, Dichte und weiteren physikalischen Größen ergänzt. Die Bauteile werden dann zu Baugruppen zusammengefügt. Auf dieses Weise kann die gesamte Maschine in der CAD-Umgebung montiert, betrachtet und bewegt werden. Anschließend können vom CAD-Modell technische Zeichnungen abgeleitet werden, welche die Grundlage für die Fertigung und Montage der Maschine sind.

Mit dem CAD-Modell steht also ein virtuelles Abbild der Maschine zur Verfügung, dass geometrisch, physikalisch und kinematisch weitestgehend mit der realen Maschine übereinstimmt. Daher ist es zweckmäßig, dieses Modell als Grundlage für die Simulation der Maschine zu nutzen. Dadurch lässt sich der Zeitaufwand zum Erstellen eines Simulationsmodells wesentlich reduzieren.

Das Ziel dieser Arbeit ist es, dem Leser anhand eines Demonstrationsmodells die nötigen Kenntnisse zu vermitteln, um folgende Arbeitsschritte selbstständig durchführen zu können:

- erzeugen eines Simulink-Modells auf Basis einer 3D-CAD-Konstruktion
- modellieren der dazugehörigen Maschinensteuerung mit Matlab Stateflow
- Co-Simulation von Steuerung und Maschine in Simulink
- generieren des Steuerungscodes mit Simulink PLC-Coder
- erzeugen des Steuerungsprogramms für die OpenPLC Soft-SPS
- Virtuelle Inbetriebnahme der Maschine mit dem Steuerungsprogramm

2 Erarbeitung eines Demonstrationsbeispiels

2.1 Vorbetrachtungen

In dieser Arbeit wird versucht, die behandelten Methoden und verwendeten Tools möglichst anschaulich und nachvollziehbar zu erläutern. Daher wird an dieser Stelle ein Demonstrationsbeispiel erarbeitet und eingeführt. Dieses ermöglicht alle Arbeitsschritte praxisnah darzustellen und kann für andere Projekte entsprechend adaptiert werden.

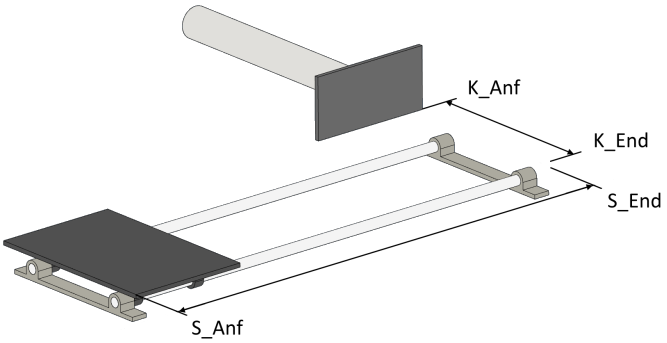
Bei der Entwicklung des Demonstrationsbeispiels wurde besonderes Augenmerk auf zwei Dinge gelegt: Die Umsetzung des Demonstrationsbeispiels soll die wesentlichen Aspekte abdecken, die zur virtuellen Inbetriebnahme mit Simulink vonnöten sind. Gleichzeitig soll es so einfach und übersichtlich wie möglich bleiben, um nicht mit unnötigen Details zu verwirren.

2.2 Paketeschieber

Aus den hiervor genannten Überlegungen entstand der „Paketeschieber“. Der Paketeschieber ist eine Maschine, die ein Objekt (Paket) mittels eines Schlittens befördert und es anschließend mit einem Schieber vom Schlitten schiebt. Bild 2.1 zeigt den Aufbau der Maschine als Ganzes, Bild 2.2 die einzelnen Bauteile als Explosionsansicht.

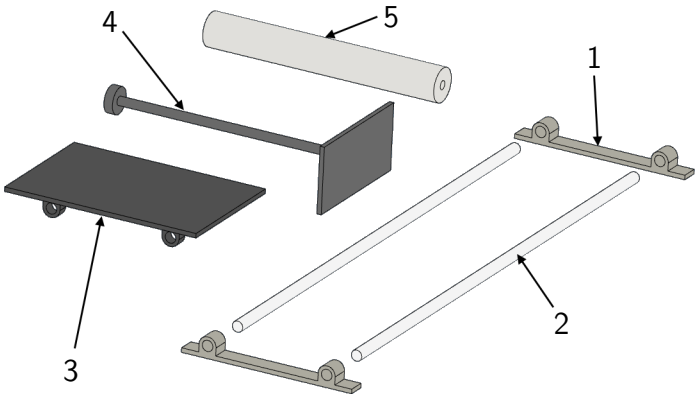
Aktoren

Der Schlitten des Paketeschiebers soll von einem Linearantrieb angetrieben werden. Dieser soll eine Antriebskraft von 100 N in beide Fahrrichtungen besitzen. Als Schnittstelle zur Steuerung besitzt er zwei digitale Eingänge: Der Eingang `EN` dient zum Ein- und Ausschalten des Antriebs. Der Eingang `DIR` steuert die Fahrtrichtung des Antriebs. Die daraus resultierenden Schaltmöglichkeiten sind in Tabelle 2.1 aufgeführt.



S_Anf	Schlitten Anfang
S_End	Schlitten Ende
K_Anf	Kolben Anfang
K_End	Kolben Ende

Bild 2.1: Aufbau des Paketeschiebers und Position der Endlagenschalter



1	Traverse
2	Linearschiene
3	Schlitten
4	Kolben/Schieber
5	Zylinder

Bild 2.2: Darstellung der Komponenten des Paketeschiebers als Explosionsansicht

Schlitten			Kolben	
EN	DIR	F in N	EN	F in N
0	0	0	0	-70
0	1	0	1	100
1	0	-100		
1	1	100		

Tabelle 2.1: Reaktion der Aktoren auf die möglichen Eingangssignale

Der Schieber wird über einen Pneumatikzylinder betätigt. Dieser soll eine Schubkraft von 100 N besitzen. Da es sich um einen Differentialzylinder handeln soll, besitzt er eine geringere Rückstellkraft von 70 N. Zur Steuerung des Kolbens dient ein einzelner digitaler Eingang EN. Liegt an diesem ein Low-Pegel an, wirkt die Rückstellkraft auf den Kolben, bei High-Pegel die Vorschubkraft.

Sensoren

Schlitten und Kolben besitzen jeweils zwei Endschalter, deren Position ebenfalls in Bild 2.1 zu sehen ist. Die Endschalter befinden sich ein Stück vor dem mechanischen Anschlag des Schlittens, um eine harte Kollision durch das rechtzeitige Ausschalten des jeweiligen Aktors zu verhindern. Der Schlitten soll darüber hinaus über einen Druckschalter verfügen, der betätigt wird, sofern ein Paket auf die Platte gestellt wird. Somit stehen fünf Sensoren zur Verfügung, die der Steuerung Informationen über die Position der Aktoren, bzw. dem Status der Maschine liefern.

Randbedingungen

Das virtuelle Modell des Paketeschiebers soll eine valide Grundlage für die Programmierung der Steuerung bilden. Dafür ist es nicht zwingend notwendig alle Eigenschaften der realen Maschine zu modellieren. Um den Aufwand zur Umsetzung des Modells zu begrenzen werden daher folgende Vereinfachungen vorgenommen:

- Reibung wird vernachlässigt
- das Paketgewicht wird vernachlässigt
- Antriebe besitzen keine Eigendynamik

2.3 Steuerung

Wie bereits im letzten Abschnitt angedeutet, soll die Maschine ein Paket, das auf den Schlitten gestellt wird, zunächst zum Schienenende befördern und es dann mit dem Kolben herunterschieben. Dieser Prozess soll mit einer SPS automatisiert werden. Hierfür bietet es sich an, die dafür notwendigen Schritte mit einem Zustandsautomaten zu formalisieren.

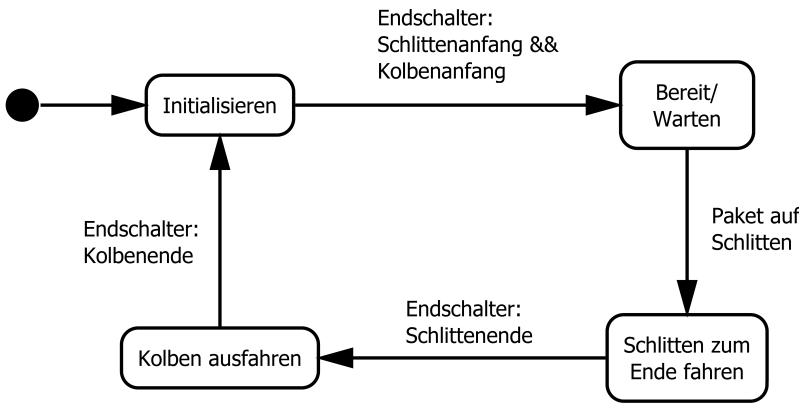


Bild 2.3: Zustandsautomat der Paketeschiebersteuerung

Im Bild 2.3 ist der Zustandsautomat zu sehen, der als Grundlage des späteren SPS-Programms dienen soll. Nach dem Start der Steuerung erfolgt zunächst eine Initialisierung. In diesem Zustand werden Kolben und Schlitten in ihre Anfangslage gebracht. Diese ist erreicht, wenn die Endlagenschalter `K_Anf` und `S_Anf` betätigt wurden. Die Erfüllung dieser Bedingung versetzt die Steuerung in den *Bereit*-Zustand. In diesem verharrt die Steuerung, bis ein Paket auf den Schlitten gestellt wird. Dies führt dazu, dass im Zustand *Schlitten zum Ende fahren* der Antrieb des Schlittens so angesteuert wird, dass er sich in Richtung Schienenende bewegt. Er wird solange verfahren, bis der Endschalter `S_End` betätigt wird. Anschließend wird der Kolben ausgefahren, bis auch er den Endlagenschalter `K_End` betätigt. Dadurch wird das Paket vom Schlitten geschoben. Daraufhin wird die Maschine über den *Initialisieren*-Zustand erneut in ihre Ausgangsposition gebracht. Die Steuerung wartet anschließend im *Bereit*-Zustand wieder auf das nächste Paket.

3 Vorgehensmodell und Toolkette

Der Arbeitsablauf von der Konstruktion des CAD-Modells, bis zur virtuellen Inbetriebnahme umfasst zahlreiche Zwischenschritte und Wechsel der genutzten Software. Da die Softwarewerkzeuge (Tools) hintereinander verwendet werden, spricht man auch von einer Toolkette. Im Folgenden wird daher ein Überblick über die erforderlichen Schritte und das Zusammenspiel der verschiedenen Programme, bzw. Tools und der erzeugten Dateien geliefert, bevor diese im Detail näher beleuchtet werden.

3.1 Verwendete Tools

In der vorliegenden Arbeit kam im Wesentlichen folgende Software zum Einsatz:

- Microsoft Windows 10 Home (x64)
- Autodesk® Inventor® Professional 2019
- MATLAB® R2017a
- OpenPLC v3

3.2 Erzeugen des Simulink-Modells

Bevor ein Simulink-Modell aus einer CAD-Baugruppe erzeugt werden kann, muss diese in einem bestimmten Format exportiert werden. Die Vorgehensweise zum Export des CAD-Modells hängt von der CAD-Umgebung ab, in der es erstellt wurde. Eine Übersicht über die verschiedenen Möglichkeiten wird in Bild 3.1 dargestellt. Bei der Nutzung von Autodesk Inventor und Creo Parametrics erfolgt zunächst der Export des Modells über das Simscape-Multibody-Link-Plugin.

Der Export erfolgt also aus der CAD-Umgebung heraus. Wurde das CAD-Modell jedoch mit der Online-CAD-Umgebung Onshape erstellt, erfolgt der Export über

Matlab. Dies geschieht mit der Funktion `smexportonshape()`, welche als Argument die URL des Onshape-Modells als String erhält.[1]

Unabhängig von der genutzten CAD-Umgebung werden beim Export sowohl STEP-Dateien aller genutzten Bauteile als auch eine XML-Datei erzeugt. Diese enthält Informationen darüber, wie die Bauteile des Modells miteinander in Beziehung stehen und welche physikalischen Eigenschaften sie haben (z. B. Material, Schwerpunkt, etc.).

Aus den exportierten Dateien kann nun ein Simulink-Modell erzeugt werden. Dazu wird zunächst Matlab geöffnet. Über das Command-Window kann nun mithilfe der Funktion `smimport('Paketeschieber')` das Simulink-Modell generiert werden. Dabei wird sowohl ein Simulink-Modell des Paketeschiebers generiert als auch ein Matlab-Skript, dass die Informationen aus der XML-Datei enthält.

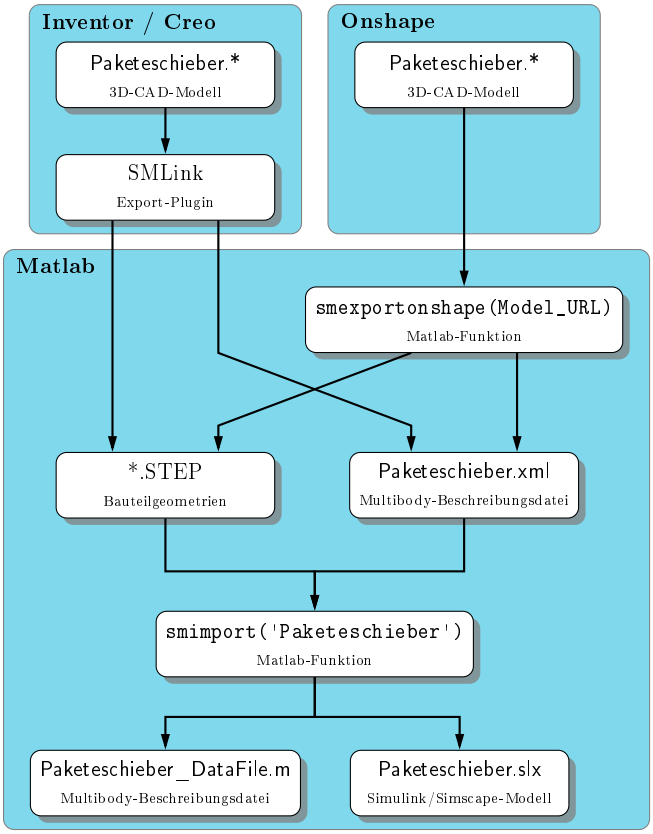


Bild 3.1: Vorgehensweise beim Erzeugen des Simulink-Modells aus einem 3D-CAD-Modell

[1] z. B. `smexportonshape('https://cad.onshape.com/documents/2d1413794c7b4842f62c1565-/w/5dea3475e52e83e8af8bc463/e/3c59d248f6ef6144b6da6aa6')`

3.3 Verfahrensablauf Virtuelle Inbetriebnahme

In diesem Abschnitt werden die einzelnen Teilschritte zur virtuellen Inbetriebnahme anhand des Vorgehensmodells in Bild 3.2 aufgezeigt.

Nachdem das Simulink-Modell generiert wurde, muss zunächst überprüft werden, ob dabei Fehler aufgetreten sind. Sollte dies der Fall sein, muss das CAD-Modell nachgebessert und erneut exportiert werden.[2] Wurden alle Fehler behoben, wird das Modell um Aktoren und Sensoren erweitert, denn diese können im CAD-Modell zwar mechanisch, aber nicht funktionell modelliert werden. Aus dem gleichen Grund müssen bewegliche Teile (Kolben und Schlitten) in ihrer Bewegung eingeschränkt werden. Dann wird – zur besseren Übersicht – aus dem erweiterten Modell ein Submodell erstellt. Dieses besitzt die jeweiligen Ein- und Ausgänge, über die die Steuerung mit dem Paketeschieber interagieren kann. Neben dem Paketeschieber-Submodell wird dann ein Stateflow-Chart erzeugt, in dem die Funktionalität der Steuerung, sowie ihre Schnittstellen modelliert werden. Paketeschieber- und Steuerungsmodell werden verbunden und können so simuliert werden. Über Signalquellen können verschiedenste Betriebssituationen studiert werden. Funktionieren Maschine und Steuerung wie gewünscht, wird mit dem Simulink-PLC-Coder aus dem Stateflow-Chart Strukturierter Text (ST)[3] generiert.

Hier erfolgt der Wechsel in den PLCOpenEditor, welcher Bestandteil der OpenPLC-Installation ist. Zunächst wird ein neues Projekt angelegt, in das anschließend der generierte ST-Code eingebettet wird. Dann müssen noch einige SPS-Parameter konfiguriert werden (Ein- & Ausgänge, Zykluszeit, usw.). Abschließend kann der fertige Steuerungscode generiert werden.

Nun wird in das Webinterface der OpenPLC-Runtime gewechselt. Dort wird zunächst die emulierte Hardware für die UDP-Kommunikation mit Simulink konfiguriert. Als Nächstes kann das Programm hochgeladen werden.

Zurück in Simulink wird das Stateflow-Chart durch UDP-Schnittstellen ersetzt, welche die Kommunikation mit OpenPLC ermöglichen. Sobald alle Schnittstellen konfiguriert wurden, erfolgt die virtuelle Inbetriebnahme durch das Starten des OpenPLC-Programms und des Simulink-Modells.

[2] siehe dazu Abschnitte 4.3 und B

[3] höhere Programmiersprache zur SPS-Programmierung, genormt nach IEC 61131-3 [5], siehe auch Abschnitt 5.1

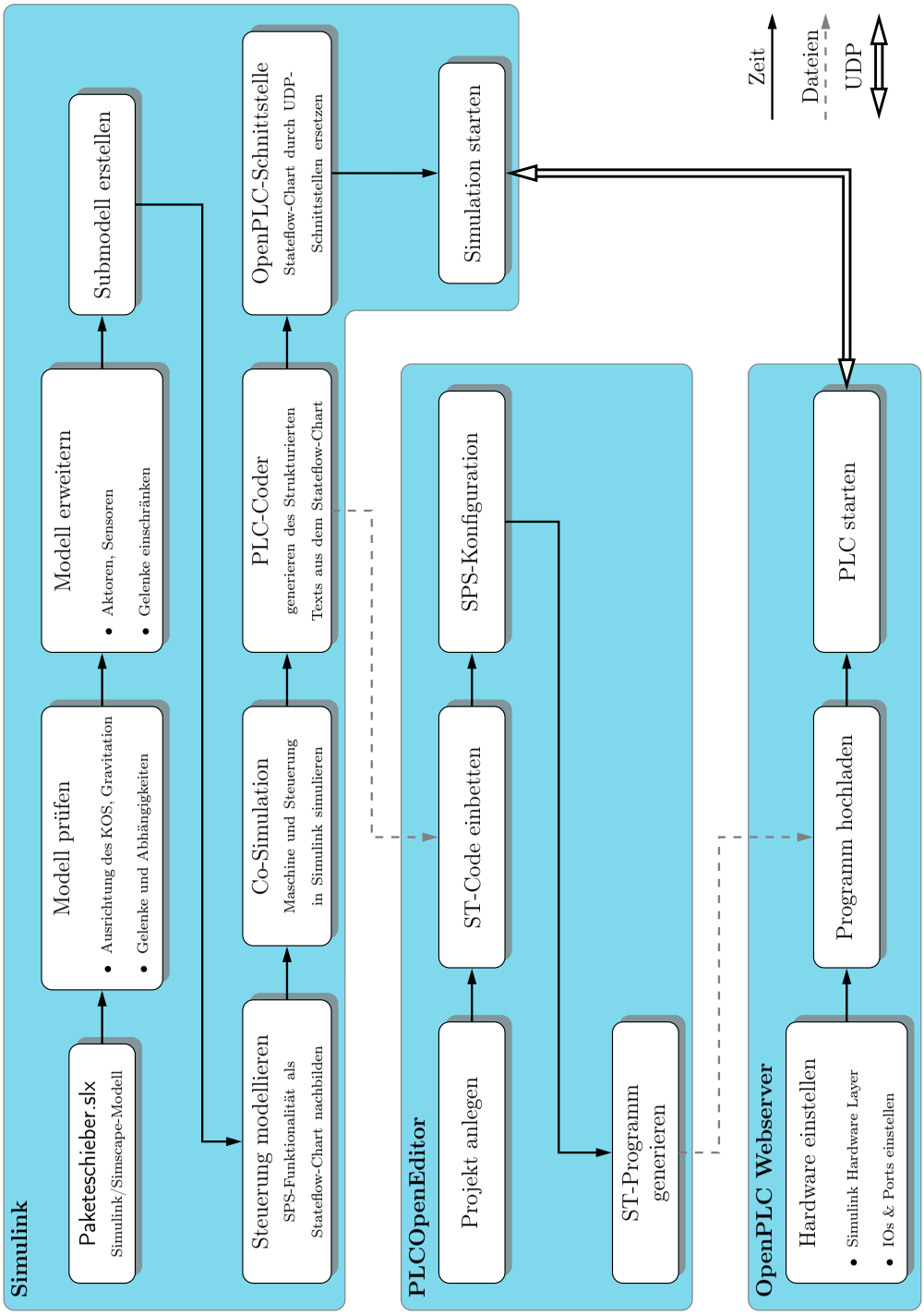


Bild 3.2: Vorgehensmodell zur virtuellen Inbetriebnahme eines Simulinkmodells mit der OpenPLC-Soft-SPS

4 Vom CAD- zum Simulink-Modell

4.1 Besonderheiten bei der CAD-Konstruktion

Da das CAD-Modell die Basis des späteren Simulink-Modells bilden soll, gilt es bei der Konstruktion ein paar Besonderheiten zu beachten. Da bereits Erfahrungen im Umgang mit Autodesk Inventor vorlagen, wurde sich im Rahmen dieses Projektes dazu entschlossen, diese Softwareumgebung für die Erstellung des CAD-Modells zu nutzen. [1]

Als Erstes werden die Einzelteile des CAD-Modells in gewohnter Art und Weise konstruiert. Jedem Teil sollte anschließend ein Material zugewiesen werden, damit seine Masse korrekt exportiert werden kann. Es ist sinnvoll starr verbundene Einzelteile in Unterbaugruppen zusammenzufassen. Im Falle des Paketeschiebers kann zum Beispiel aus den Traversen und Linearführungen die Unterbaugruppe *Schiene* gebildet werden.

Daraufhin wird die Paketeschieber-Baugruppe erstellt, die später exportiert werden soll. In dieser Baugruppe wird der Paketeschieber aus den Unterbaugruppen und restlichen Einzelteilen zusammengesetzt. Besondere Sorgfalt ist bei der Erzeugung beweglicher Verbindungen vonnöten. Nicht alle in Inventor zur Verfügung stehenden Verbindungstypen lassen sich durch das smlink-Plugin exportieren (z. B. Tangential). Es hat sich bewährt, die Freiheitsgrade beweglicher Bauteile zunächst mit Abhängigkeiten so zu beschränken, dass sie sich nur noch in der konstruktiv vorgesehenen Weise bewegen lassen. Anschließend wird eine Verbindung hinzugefügt, mit deren Hilfe gegebenenfalls vorhandene Endanschläge eingestellt werden können. Eine illustrierte Beschreibung des Vorgehens am Beispiel der Verbindung Schlitten/Schiene ist im Anhang zu finden (B).

Ist das Modell vollständig zusammengesetzt, können Schlitten und Kolben mit der Maus bewegt werden. Für die spätere Arbeit mit dem Simulink-Modell ist es nö-

[1] Als frei zugängliche Alternative wurde auch Onshape getestet. Mit der Konstellation Matlab 2017b und der aktuellen Onshape-Version konnten jedoch keine lauffähigen Simulationsmodelle erzeugt werden.

tig, dass Schlitten und Kolben vor dem Export in eine der Endlagen gebracht werden.

Zu guter Letzt wird die Baugruppe exportiert. Dazu kommt das smlink-Plugin zum Einsatz. Dieses findet sich bei Inventor unter dem Menüpunkt *Zusatzmodule/Simscape Multibody Link/Export Simscape Multibody*. Nach der Wahl des Exportverzeichnisses öffnet sich ein Matlab Command Window. Dieses kann geschlossen werden, sobald die Meldung über den Abschluss des Exports in Form eines weiteren Fensters erscheint.

4.2 Erzeugen des Simulink-Modells

Unabhängig von der gewählten CAD-Umgebung liegen nun, nach dem Export des Modells, die `Paketeschieber.xml`-Datei und die dazugehörigen STEP-Dateien vor. Um daraus das Simulink-Modell zu erzeugen wird im Matlab Command die Funktion `smimport('Paketeschieber')` aufgerufen. Diese erzeugt aus der XML-Datei ein Matlab-Skript. Zudem öffnet sich das generierte Simulink-Modell. Dieses ist in Bild 4.1 zu sehen.

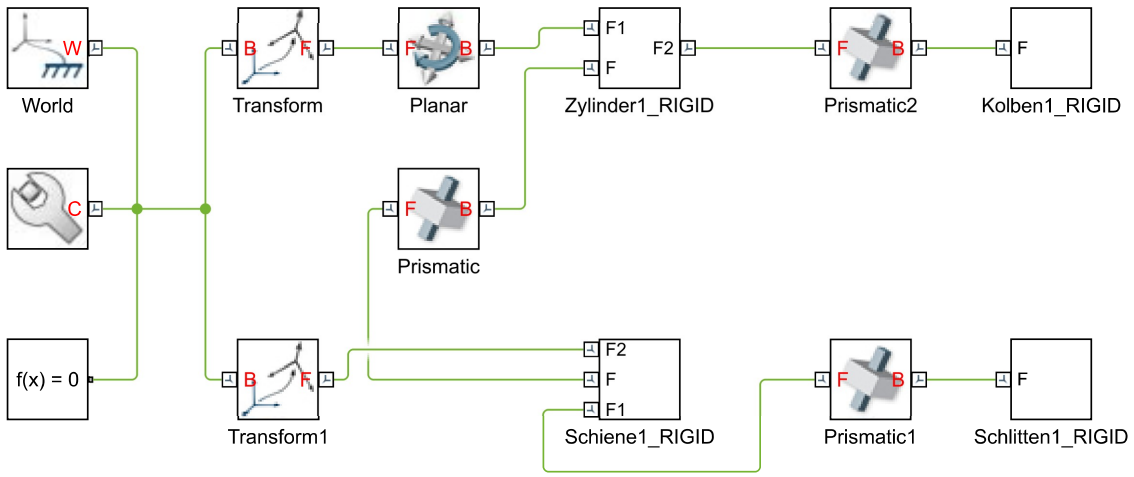


Bild 4.1: Generiertes Simulink-Modell

Das Modell besteht aus den Blöcken der Bauteile und den Gelenken die sie Verbinden. Dazu kommen weitere Blöcke, die die Koordinatentransformation zwischen den Bauteilen vornehmen und Konfigurationsblöcke, wie zum Beispiel der `Solver Configuration`-Block.

Wenn im CAD-Modell Verbindungen der Art $A \longleftrightarrow B \longleftrightarrow C$ erzeugt wurden, kann es passieren, dass diese durch den Export zu einer Verbindung $A \longleftrightarrow C$

reduziert werden. Ebenso kann es passieren, dass „Phantomblöcke“ erzeugt werden, wie zum Beispiel `Transform`-Blöcke mit nicht verbundenen Ports. Dadurch erscheint das Simulink-Modell zunächst etwas unübersichtlich und kontraintuitiv. Daher ist es meist sinnvoll die Anordnung der Blöcke händisch anzupassen.

Nachdem das Simulink-Modell erzeugt wurde, kann es nun erstmals gestartet werden. Dabei öffnet sich der *Mechanic Explorer*, in dem das 3D-Modell des Paketeschiebers zu sehen ist. Am unteren Bildrand befindet sich ein Balken, der die aktuelle Simulationszeit anzeigt. Unten links ist ein Koordinatensystem zu sehen. Mit dessen Hilfe sollte zunächst überprüft werden, ob die Gravitationsrichtung des Modells mit der gewollten Richtung übereinstimmt. Die Gravitationsrichtung und -stärke des Modells lässt sich über den `Mechanism Configuration`-Block einstellen.

4.3 Trial and Error

Ist die Gravitationsrichtung richig eingestellt, sollte der Paketeschieber während der Simulation zunächst nur verharren. Die Arbeit mit dem smlink-Plugin hat jedoch gezeigt, dass nicht immer alle Verbindungen und Gelenke so aus Inventor exportiert werden, wie es vorgesehen war. Es kommt mitunter vor, dass eigentlich verknüpfte Bauteile während der Simulation auseinanderfallen.

Da weder ersichtlich ist, in welcher Weise das smlink-Plugin verschiedene Verbindungen exportiert, noch wie die `smimport()`-Funktion das Simulink-Modell generiert, konnte dem Problem bisher nur durch ausprobieren verschiedener Abhängigkeits- und Verbindungskombinationen begegnet werden.

Das bedeutet, dass beim Auseinanderfallen des Modells die fragliche Verbindung in Inventor ausfindig gemacht und diese durch eine funktionell gleiche Verbindungskombination ersetzt wurde. Anschließend erfolgt erneut der Export des Modells und der Import in Matlab. Der so entstehende iterative Prozess ist im Bild 4.2 schematisch dargestellt.

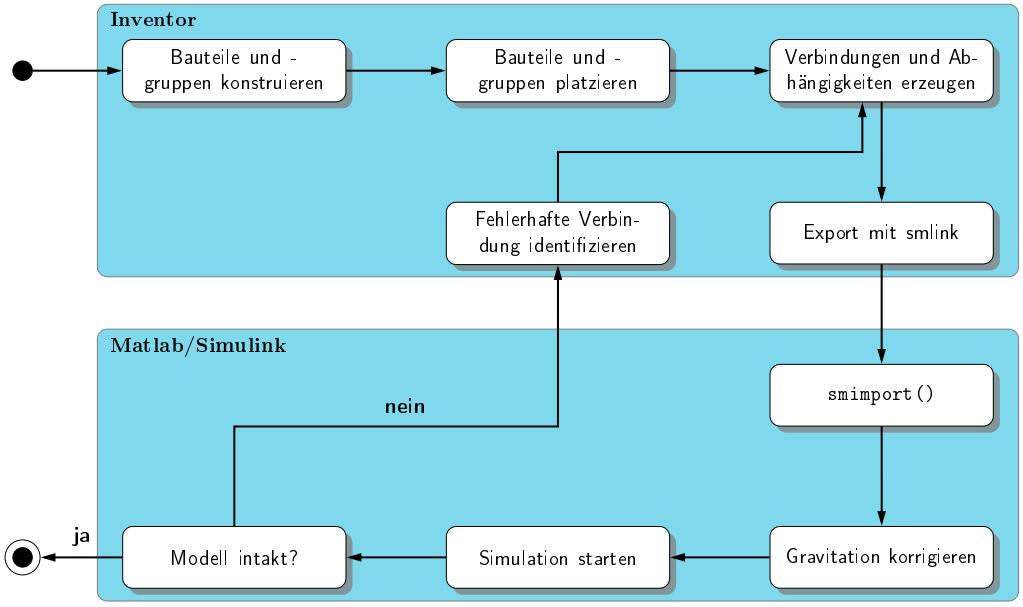


Bild 4.2: Schema der iterativen Vorgehensweise zum Erzeugen eines intakten Simulink-Modells

4.4 Erweitern des Simulink-Modells

Nachdem der Paketeschieber nun mechanisch intakt als Simulink-Modell vorliegt, soll er um die in Abschnitt 2.2 beschriebenen Aktoren und Sensoren erweitert werden. Am Beispiels des Schlittens wird im Folgenden erläutert, wie Gelenke um Antriebe und Sensoren ergänzt werden können.

4.4.1 Aktoren

Für den Schlitten wird ein Linearantrieb modelliert. Dazu wird im Blockschaltbild zunächst die bewegliche Verbindung am Schlitten identifiziert. Im Bild 4.1 ist es der Block `Prismatic1`. Durch einen Doppelklick auf den Block öffnet sich ein Dialogfenster. In diesem wird unter `Z Prismatic Primitive (Pz)/Actuation/Force` der Punkt `Provided by Input` ausgewählt. Dadurch entsteht am Block ein Eingang, an dem die Stärke der in z-Richtung wirkenden Kraft auf den Schlitten eingestellt werden kann. Der Schlitten besitzt nun einen Linearantrieb mit variabler Antriebskraft.

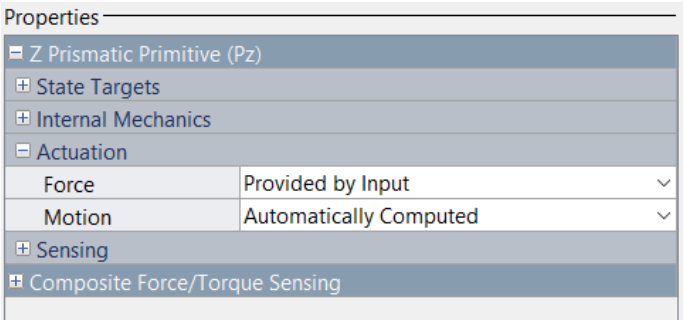


Bild 4.3: Dialogfenster des Gelenks `Prismatic1`. Unter *Actuation/Force* ist die Vorgabe einer Kraft über einen Blockeingang ausgewählt.

4.4.2 Sensoren

Analog zu den Aktoren wird dem Gelenkblock `Prismatic1` über das Dialogfenster zunächst ein Port hinzugefügt. Unter dem Punkt `Z Prismatic Primitive (Pz)/Sensing` wird dazu der Punkt `Position` angewählt. Nun kann die aktuelle z-Koordinate des Schlittens über den neuen Port gelesen werden.

Die so erzeugten Ein- und Ausgangsports sind keine „normalen“ Simulink-Ports. Sie arbeiten nur mit sogenannten *Physical Signals*. Daher ist es notwendig entsprechende Konverterblöcke zu nutzen, um sie mit den Standardblöcken zu verbinden. Die Konverterblöcke lassen sich im Simulink Library Browser unter *Simscape/Utilities* finden. Der um Aktor, Sensor und Konverterblöcke erweiterte Schlitten ist in Bild 4.4 zu sehen.

4.4.3 Endanschläge

Nachdem der Paketeschieber um Aktoren erweitert wurde, können Simulationen mit verschiedensten Antriebskräften durchgeführt werden. Dabei fällt auf, dass die Simscape-Gelenkblöcke (z. B. `Prismatic`) zwar die Freiheitsgrade der verbundenen

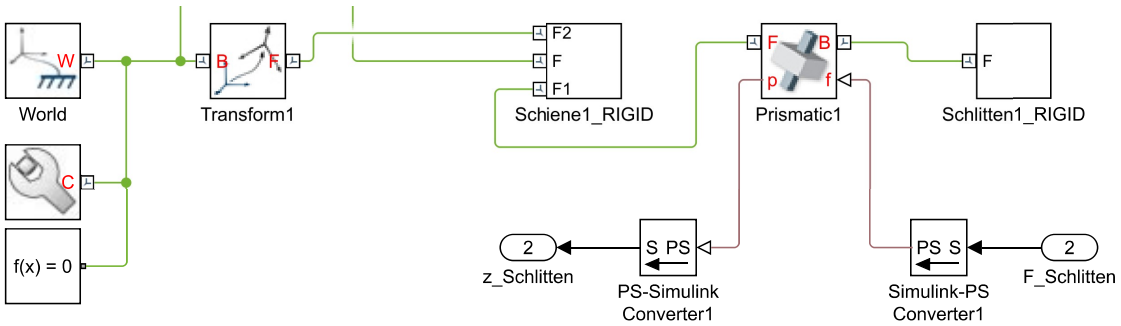


Bild 4.4: Um Antrieb und Positionssensor erweiterter Schlitten, sowie Konverterblöcke für PS-Signale

Komponenten einschränken, eine Bewegung entlang dieser ist aber unbegrenzt möglich. Daher passiert es, dass der Schlitten über die Schienenenden hinausfährt, wenn er nicht vorher abgebremst wird.

Diesem Problem mit positionsabhängigen Gegenkräften zu begegnen ist sehr aufwendig und unübersichtlich. Daher wurde eine elegantere Lösung gesucht. Diese fand sich in der Simscape Multibody Multiphysics Library (SMML)[9], die über MathWorks File Exchange bezogen werden kann.[2]

In dieser Bibliothek stehen sogenannte Hard Stop Friction (HSF)-Blöcke zur Verfügung, mit denen sehr komfortabel Endanschläge für Rotationen und Translationen modelliert werden können. Wichtig für die Nutzung der Blöcke ist, dass sich die betroffenen Komponenten in einer Endlage befinden, da nur die Distanz bis zum nächsten Endanschlag eingestellt werden kann, nicht aber die absolute Position. Darüber hinaus lassen sich auch andere Eigenschaften der Anschläge parametrieren, wie zum Beispiel Dämpfung und Reibung.

Die Verwendung der HSF-Blöcke ist denkbar einfach: Wie in Bild 4.5 zu sehen, wird der HSF-Block lediglich mit dem Ein- und Ausgang des betreffenden Gelenkblocks verbunden. Sollte sich bei der Simulation herausstellen, dass die Bewegung in der falschen Richtung begrenzt wurde, so werden die Anschlüsse des Blocks vertauscht.

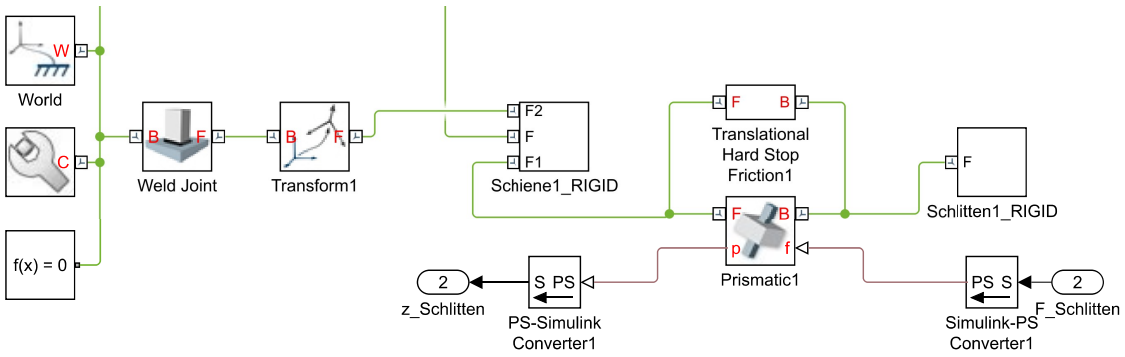


Bild 4.5: Verwendung des HSF-Blocks und Fehlerbehebung durch Weld-Block

Bei der Verwendung des HSF-Blocks kann es zu einer Fehlermeldung kommen. Diese besagt, dass die Verwendung des Block keine Auswirkung hätte, da er einseitig mit dem Weltblock verbunden ist. Dieser Fehler lässt sich dadurch korrigieren, dass zwischen Weltblock und dem Gelenkblock ein `Weld`-Block eingefügt wird, wie ebenfalls in Bild 4.5 zu sehen ist. Dieser erzeugt eine Schweißverbindung zwischen Welt und Gelenk, verändert das System grundsätzlich aber nicht.

[2] zur Installation siehe Anhang A.2

Nachdem Schlitten und Kolben, wie hiervor beschrieben, mit Antrieben und Sensoren ausgestattet sind, wird zur besseren Übersicht ein Submodell erzeugt. Dieses ist in Bild 4.6 dargestellt.

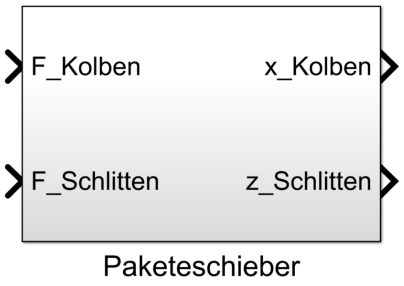


Bild 4.6: Submodell des Paketeschiebers, mit Eingangsports zur Kraftvorgabe und Ausgangsports zur Positionsbestimmung von Schlitten und Kolben

4.4.4 digitale Ein- und Ausgänge

Bei der Erarbeitung des Demonstrationsbeispiel in Kapitel 2 wurden Endlagenschalter an der Schiene des Schlittens vorgesehen. Diese praxisnahe Möglichkeit zur Positionsbestimmung soll auch in der Simulation berücksichtigt werden.

Die Endlagenschalter sind digitale Signalquellen. Sie sollen einen High-Pegel erzeugen, wenn der Schlitten einen definierten Abstand zu ihnen unterschreitet. Andernfalls erzeugen sie einen Low-Pegel. Dieses Verhalten lässt sich über die Sensing-Funktionen des `Prismatic`-Blocks nicht realisieren.

Der aktuelle Pegel eines Schalters lässt sich als Zustand auffassen, der von der Position des Schlittens abhängig ist. Aus diesem Grund bietet es sich an, das oben beschriebene Verhalten mit Stateflow zu modellieren. Der so entstandene Zustandsautomat der Endlagenschalter ist in Bild 4.7 abgebildet.

Über die Eingänge `z_Schlitten` und `x_Kolben` erhält das Chart die Positionen der Aktoren. Aus diesen wird dann der Pegel des entsprechenden Endlagenschalters bestimmt. Über die vier Ausgänge werden die Pegel der Endlagenschalter mit dem Datentyp `boolean` zurückgegeben. Die maximalen Verfahrwege von Schlitten und Kolben sind als `s_Schlitten` und `s_Kolben` in dem Matlab-Skript `Paketeschieber_Parameter.m` definiert. Die Endlagenschalter schalten bereits kurz vor dem mechanischen Schienenende, um eine harte Kollision zwischen Traverse und Schlitten zu verhindern. Die Variable `delta_Sens` gibt diese Entfernung, zwischen Schaltpunkt und Schienenende an. Auch sie ist in dem Matlab-Skript hinterlegt.

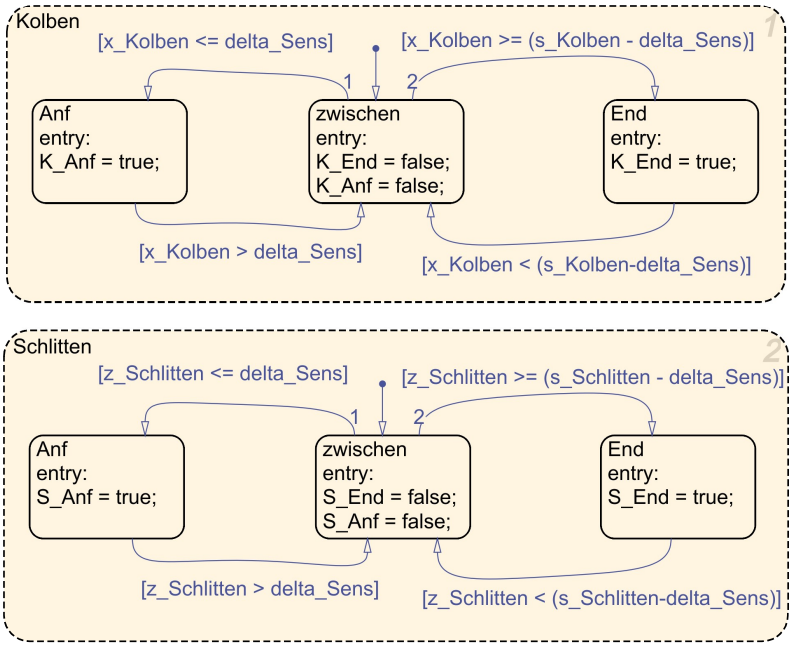


Bild 4.7: Stateflow-Chart der Endlagenschalter

Auf diese Weise lassen sich die analog vorliegenden Positionen der Aktoren in digitale Signale überführen. Ebenso werden auch die analogen Eingänge der Aktoren mit Zustandsautomaten in digitale Eingänge überführt. Daraus entsteht der Paketeschieber, mit den in Kapitel 2 vorgesehenen digitalen Ein- und Ausgängen. Bild 4.8 zeigt den digitalisierten Paketeschieber.

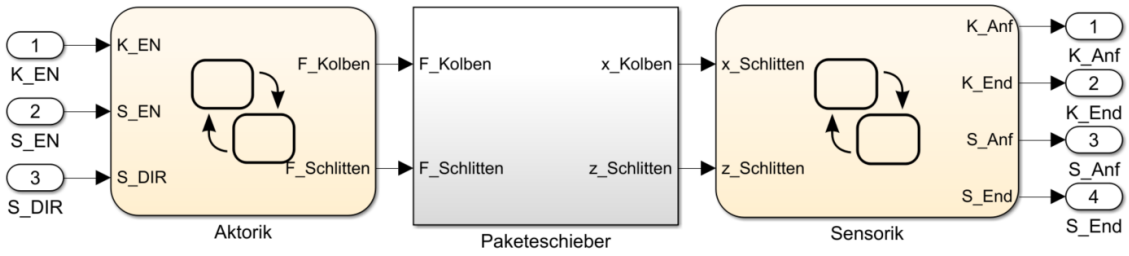


Bild 4.8: Paketeschieber mit digitalen Ein- und Ausgängen

4.5 Testen des Modells

Bevor das erstellte Modell des Paketeschiebers zum Entwickeln der Steuerung genutzt werden kann, muss überprüft werden, ob es wie vorgesehen funktioniert. Dazu wird das Verhalten des Modells bei unterschiedlichen Ansteuerungen begutachtet. Dabei lässt sich leicht feststellen, ob es sich Aktoren und Sensoren kausal und plausibel verhalten.

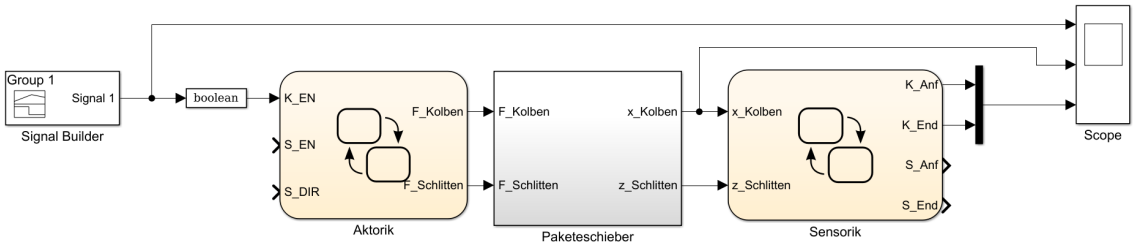


Bild 4.9: Testbench zum Prüfen des Kolbenverhaltens

Zu diesem Zweck wird um das Paketeschiebermodell eine Testbench (virtueller Prüfstand) aufgebaut. Diese besteht im wesentlichen aus Signalquellen für die Eingänge der Aktoren und Scope-Blöcken zur Anzeige der Ausgangssignale. Bild 4.9 zeigt den Aufbau der Testbench, der Übersichtlichkeit halber ohne Testmöglichkeit für den Schlitten.

Zunächst soll die korrekte Funktion des Kolbens nachgewiesen werden. Da er nur zwei mögliche Eingangssignale erhalten kann (low und high), genügt ein Rechtecksignal um diese abzudecken. Das Signal, die Pegel der Sensoren und die Position des Kolbens sind in Bild 4.10 zu sehen. Bei 0,5 s wird der Pegel des EN-Eingangs auf high gesetzt. Man kann erkennen, dass der Kolben ab diesem Moment beschleunigt. In der zuvor parametrierten Entfernung zur jeweiligen Endlage, ändert der Endlagenschalter Kolben_Anf seinen Pegel von high auf low. Da keine Reibung modelliert wurde, beschleunigt der Kolben immer weiter, bis er am Schienenende angelangt ist. Hier kann der Pegelwechsel des Kolben_End beobachtet werden. Beim ausschalten des Kolbens durch einen Low-Pegel an EN beschleunigt der Kolben in die andere Richtung und es kann erneut sowohl die ordnungsgemäße Funktion der Endlagenschalter, als auch das Stoppen des Kolbens in der Endlage selbst geprüft werden.

Auf diese Weise wird auch der Schlitten getestet. Da dieser zwei Eingänge zur Steuerung besitzt, müssen die daraus resultierenden Kombinationsmöglichkeiten beim Testen berücksichtigt werden. Im Bild 4.11 sind die gewählten Signalverläufe, die Reaktion des Schlittens und der Endlagenschalter zu sehen. Bei ausgeschaltetem Antrieb ($EN = 0$) bewegt sich der Schlitten nicht. Bei eingeschaltetem Antrieb beschleunigt der Schlitten. Die Richtung der Beschleunigung wird durch den Pegel des DIR-Eingangs vorgegeben. Auch die Endlagenschalter ändern ihren Pegel in Abhängigkeit der Position des Schlittens. Somit entspricht auch das Verhalten des Schlitten den Vorgaben.

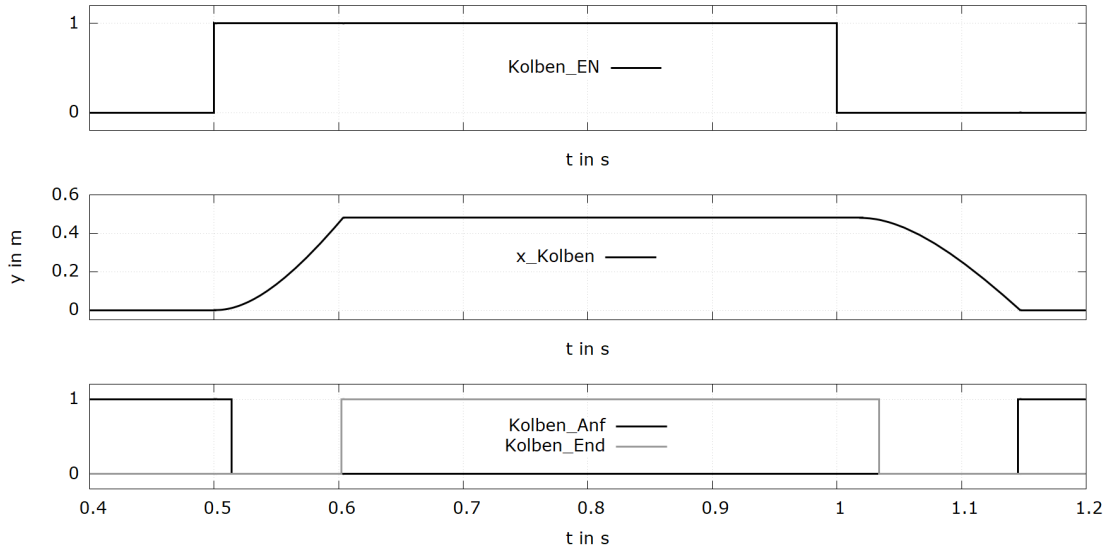


Bild 4.10: Verhalten des Kolbens und der Sensoren bei einem rechteckförmigen Eingangssignal

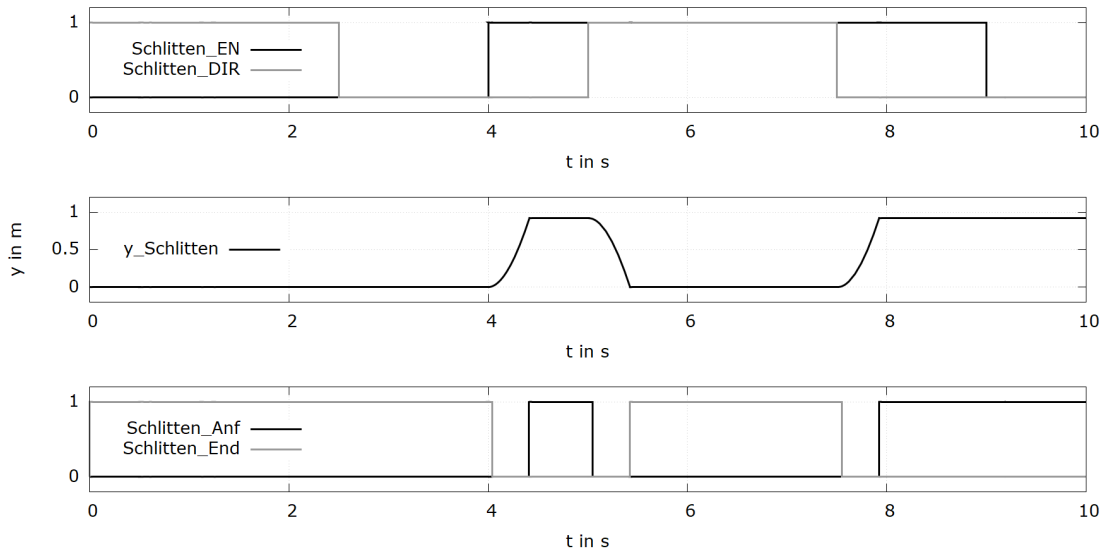


Bild 4.11: Verhalten des Schlittens und der Sensoren bei verschiedenen Eingangssignalen

5 Erzeugen des Steuerungsprogramms

5.1 Grundlagen

Als Steuerung für den Paketeschieber soll die freie Soft-SPS *OpenPLC* (engl.: offene SPS) genutzt werden. Sie stellt im Wesentlichen die SPS-Funktionalitäten der Norm IEC 61131-3 zur Verfügung. Diese enthält Richtlinien zur SPS-Programmierung und definiert Programmiersprachen. Nach der Norm stehen dem Programmierer folgende Programmiersprachen zur Verfügung:

- Kontaktplan (KOP)
- Funktionsbausteinsprache (FBS)
- Ablaufsprache (AS)
- Anweisungsliste (AWL)
- ST

Alle fünf Sprachen werden von OpenPLC unterstützt. Die ersten zwei Sprachen dieser Aufzählung sind graphische Sprachen, die letzten beiden textbasiert. Von der Ablaufsprache existiert sowohl eine graphische als auch eine textbasierte Variante.

Der Simulink PLC-Coder kann lediglich Code in zwei dieser genormten Sprachen generieren: ST und KOP. Da KOP als SPS-Programmiersprache mehr und mehr an Bedeutung verliert, wird sich im Rahmen dieser Arbeit auf die Verwendung von ST beschränkt. Bei ST handelt es sich um eine Hochsprache, mit der auch komplexe Programme übersichtlich formuliert werden können.[5]

Da der Steuerungscode mithilfe des PLC-Coders aus dem Stateflow-Chart generiert wird, sind keine tiefgreifenden Programmierkenntnisse in ST notwendig. Aus diesem Grund wird an dieser Stelle auf eine ausführliche Einführung in die Programmiersprache verzichtet.

5.2 Modellierung der Steuerungslogik in Matlab Stateflow

Da der Paketeschieber nun als virtuelle Maschine zur Verfügung steht, kann an ihm die Steuerung entworfen werden. Dazu wird der bereits im Abschnitt 2.3 vorgestellte Zustandsautomat in Matlab Stateflow modelliert und anschließend am Paketeschieber getestet.

5.2.1 Umsetzung des Automaten

Das Stateflow-Chart (Bild 5.1) der Steuerung spiegelt im Wesentlichen das in Abschnitt 2.3 eingeführte Steuerungskonzept wider. Davon abweichend wurden die Zustände zum Handhaben des Paketes in einen Übergeordneten Zustand zusammengefasst. Bei der Implementierung der Zustände wurde darauf geachtet, dass der Schlittenantrieb beim Erreichen der Endanschläge abgeschaltet wird. Dies ist beim Kolben nicht notwendig, da ein druckloser Betrieb des Pneumatikzylinders nicht vorgesehen ist.

Da zur Steuerung des Paketeschiebers digitale Ein- und Ausgänge genutzt werden, wurden die Ports des Steuerungs-Charts auf den Datentyp *boolean* eingestellt.

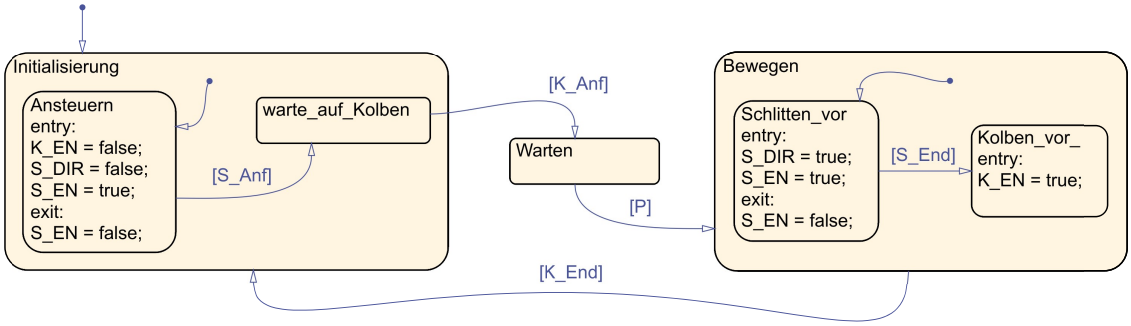


Bild 5.1: Stateflow-Chart als Basis des späteren Steuerungsprogramms

5.2.2 Testen des Steuerungsdesigns

An dieser Stelle können zum ersten Mal Maschine und Steuerung gemeinsam simuliert und getestet werden. Dazu wird der in Bild 5.3 gezeigte Aufbau verwendet.

Durch den Start der Simulation wird im Mechanics Explorer die gesteuerte Maschine angezeigt. Durch die Animation des Systems können grobe Fehler schnell gefunden

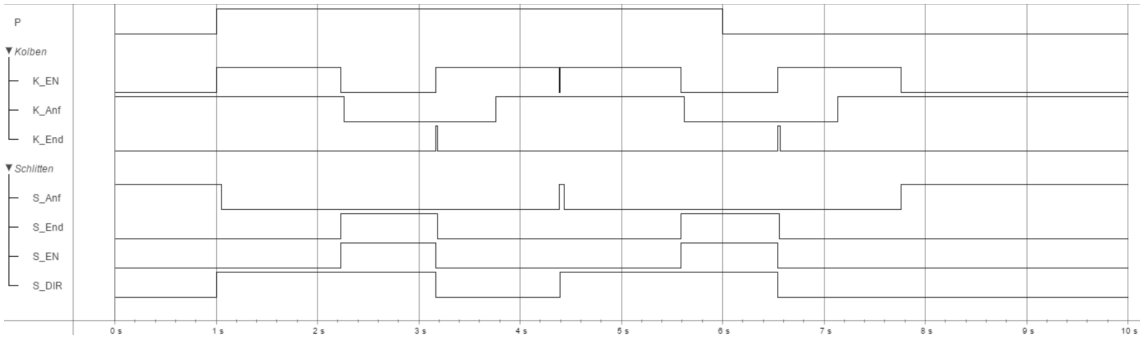


Bild 5.2: Pegel der Ein- und Ausgänge von Maschine und Steuerung

werden. Zusätzlich dazu können in Simulink, mithilfe des Logic Analyzers, die Signalpegel aller Ein- und Ausgänge über der Zeit untersucht werden. Ein Auschnitt aus dem Logic Analyzer ist in Bild 5.2 zu sehen.

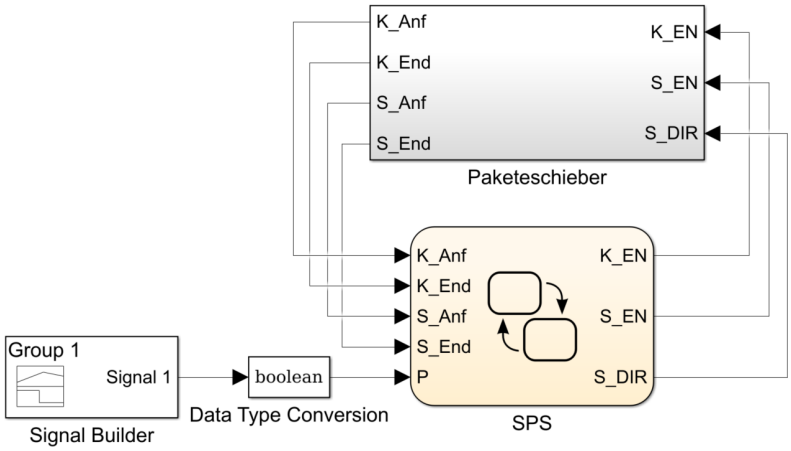


Bild 5.3: Aufbau zum Test des Steuerungsdesigns am Paketeschieber

5.3 Generieren des Strukturierten Texts mit Simulink PLC-Coder

Um das in Stateflow modellierte Chart als Grundlage des OpenPLC-Programms nutzen zu können, muss es in einer geeigneten Form exportiert werden. Zu diesem Zweck wird der Simulink PLC-Coder verwendet. Dazu wird zunächst mit einem Rechtsklick das Kontextmenü des `SPS`-Charts geöffnet. Unter *PLC Code/Options* wird das Einstellungsfenster geöffnet. Die Option *Target IDE* gibt die Zielentwicklungsumgebung des Exports an. Unter den Auswahlmöglichkeiten steht auch *PLCOpen XML* zur Verfügung. Es hat sich jedoch gezeigt, dass die durch den PLC-Coder generierte Projektdatei nicht mit dem PLCOpenEditor der OpenPLC-Umgebung kompatibel ist. Aus diesem Grund wird der Punkt *Generic* gewählt. Die übrigen Felder können auf ihren Standardeinstellungen belassen werden.

Als nächstes kann mit dem Kontextmenüpunkt *PLC Code/Check Subsystem Compatibility* geprüft werden, ob aus dem Stateflowchart ST-Code generiert werden kann. Nach der Überprüfung wird im Unterordner `../plcsrc` eine Textdatei angelegt, die Probleme und Hinweise enthält. Da viele Probleme beim Generieren des Codes automatisch gelöst werden, lohnt es sich vor dem Abarbeiten der Fehlerliste zunächst zu versuchen, den ST Code über den Menüpunkt *PLC Code/Generate Code for Subsystem* zu generieren. Auch dieser wird in dem Unterordner abgelegt.

5.4 Integration in OpenPLC

5.4.1 Erzeugen eines Projekts im PLCOpenEditor

Der generierte ST-Code kann nach dem Export mit einem Texteditor betrachtet werden. Dabei erkennt man, dass der Simulink PLC-Coder aus dem Stateflowchart einen *Function Block* generiert hat. Dieser Function Block stellt kein lauffähiges Steuerungsprogramm dar, sondern eine Programmorganisationseinheit (POE) nach der IEC 61131-3. Ein solcher Function Block, oder auch Funktionsbaustein, ähnelt einer Matlab-Funktion. Ein entscheidender Unterschied ist, dass ein Funktionsbaustein über ein „Gedächtnis“ verfügt. Das bedeutet, seine Rückgabewerte hängen nicht nur von den aktuellen Eingangswerten, sondern auch von den vorherigen (bzw. von den internen Variablen) ab.[5]

Auffällig sind die Eingangsvariablen `ssMethodType` und `b_P`. Durch einen Blick auf den ST-Code wird jedoch schnell deutlich, dass über die Variable `ssMethodType` eine Initialisierung, bzw. ein Reset des SPS-Blocks vorgenommen wird. Der Port `b_P` ist lediglich der `P`-Port des Stateflowcharts. Dieser wurde bei der Codegenerierung umbenannt, da der Name `P` bereits reserviert ist. Der Hinweis dazu findet sich in der Logdatei, die bei der Kompatibilitätsprüfung durch den PLC-Coder angelegt wurde.

Um den erzeugten Funktionsbaustein nutzen zu können, ist es notwendig ihn in ein Programm einzubinden. Dies geschieht mit dem PLCOpenEditor, der der OpenPLC-Installation beiliegt. In diesem wird zunächst ein Projekt angelegt und die erforderlichen Felder in der erscheinenden Maske ausgefüllt (Name, Organisation, etc.). Am linken Rand des Fensters ist nun der Projektexplorer zu sehen. Durch einen Klick auf das große Pluszeichen wird ein Funktionsblock hinzugefügt. Als Name wird der

Name des generierten Blocks verwendet (in diesem Fall `SPS`). Als Bausteintyp bleibt Funktionsblock ausgewählt. Die Sprache ist ST. Nach Bestätigung der Eingaben öffnet sich ein neuer Reiter. In der oberen Hälfte können Variablen hinzugefügt und bearbeitet werden. In der unteren Hälfte können die Anweisungen programmiert werden.

Da der `SPS`-Funktionsblock schon als ST-Code zur Verfügung steht, kann dieser weitgehend wiederverwendet werden. Dazu wird er mit einem Texteditor geöffnet (z. B. Notepad) und der gesamte Inhalt der Datei in das Textfeld im PLCOpenEditor kopiert. Da der PLCOpenEditor aus den Variablen im oberen Menü und den Anweisungen im Textfeld einen neuen Funktionsblock generiert, müssen einige Anpassungen vorgenommen werden. Zunächst müssen die Variablen im Textfeld über das Variablenmenü neu angelegt werden. Dafür reicht es lediglich die Felder Name, Klasse und Typ aus dem Textfeld zu übernehmen (siehe Bild 5.4). Die restlichen Felder bleiben leer.

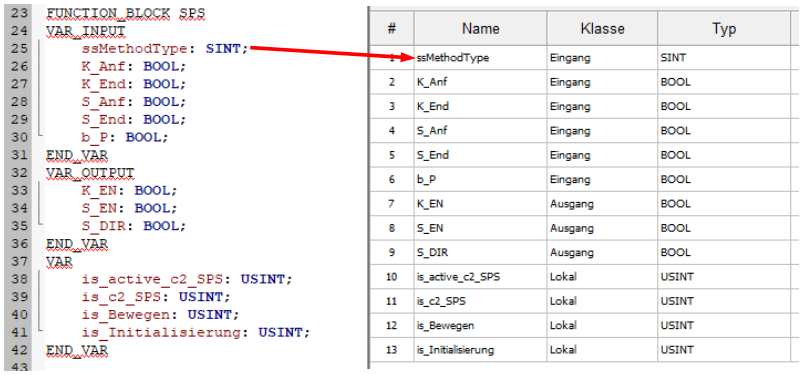


Bild 5.4: Übertragen der Variablen in das entsprechende Menü

Wenn alle Schlängellinien unter den Variablennamen im Textfeld verschwunden sind, wurden die Namen ohne Tippfehler übertragen. Im Anschluss daran werden der gesamte Textblock, der im Bild 5.4 zu sehen ist und das `END_FUNCTION_BLOCK` am Ende des Codes gelöscht. Damit ist SPS-Funktionsblock fertig konfiguriert.

Nun wird dem Projekt über das große Plus ein Programm hinzugefügt. Das Programm bildet die Schnittstelle zwischen dem Funktionsblock und den Ein- und Ausgängen der SPS. Wie alle hier erstellten Komponenten, muss der gewählte Name (hier `Prog0`) großgeschrieben werden. Aus Anschauungsgründen wird dieses Mal FUP als Programmiersprache ausgewählt. Durch einen Rechtsklick in das leere Fenster wird über *Hinzufügen/Funktionsblock/Benutzerbausteine* der erzeugte SPS-Baustein ausgewählt und hinzugefügt. Nun können über das Variablenmenü erneut Variablen für jeden Port erstellt werden. In diesem Fall sind die erzeugten Variablen alle vom Typ `Local` und bekommen die Adresse der jeweiligen SPS-Schnittstelle

zugewiesen. Dabei bekommen digitale Eingänge Adressen der Form %IX0.x, wobei das x für die Nummer des Ports, beginnend bei null steht. Digitale Ausgänge werden in der Form %QX0.x adressiert.

Über einen Rechtsklick neben den SPS-Funktionsblock und den Menüpunkt *Hinzufügen/Variable* können die zuvor erstellten Variablen dem Blockschaltbild hinzugefügt werden. Durch Wählen der entsprechenden Klasse lässt sich die Richtung des Variablenports festlegen. Ein Stolperstein stellt der `Reset`-, bzw. `ssMethodType`-Port dar. Durch den PLC-Coder hat dieser den Datentyp `SINT` zugewiesen bekommen. Die digitalen Ports der OpenPLC-Soft-SPS unterstützen jedoch nur den Datentyp `BOOL`. Da die Variable `ssMethodType` jedoch lediglich den Wertebereich von null und eins benötigt, kann eine Typenkonvertierung vorgenommen werden. Zu diesem Zweck wird aus der Bibliothek am rechten Fensterrand, aus dem Unterpunkt Typumwandlung, die Funktion `BOOL_TO_SINT` in das Blockschaltbild gezogen. Verbindet man nun alle Ports sollte sich der in Bild 5.5 Aufbau ergeben.

#	Name	Klasse	Typ	IEC-Adresse
1	SPS0	Lokal	SPS	
2	K_Anf	Lokal	BOOL	%IX0.0
3	K_End	Lokal	BOOL	%IX0.1
4	S_Anf	Lokal	BOOL	%IX0.2
5	S_End	Lokal	BOOL	%IX0.3
6	P	Lokal	BOOL	%IX0.4
7	Reset	Lokal	BOOL	%IX0.5
8	K_EN	Lokal	BOOL	%QX0.0
9	S_EN	Lokal	BOOL	%QX0.1
10	S_DIR	Lokal	BOOL	%QX0.2

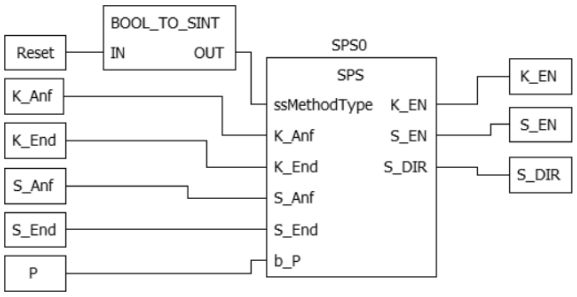


Bild 5.5: SPS-Programm als FUP, sowie die dazugehörige Portkonfiguration

Zu guter Letzt muss die SPS noch konfiguriert werden. Dazu wird über das große Plus die Konfiguration `Config0` erzeugt. Dieser Konfiguration wird dann die Ressource `Res0` hinzugefügt. Innerhalb der Ressource wird ein Task `Task0` erzeugt. Diesem Task kann eine Zykluszeit zugewiesen werden, mit der er später aufgerufen wird. Als letztes wird der Ressource eine Instanz `Inst0` hinzugefügt. Diese erhält den Typ `Prog0` und den zuvor erstellten `Task0`. Nun kann über das Menü *Datei* das Programm generiert werden.

5.4.2 Konfigurieren von OpenPLC

Wurde OpenPLC, wie im Anhang A.3 beschrieben, installiert und eingerichtet, kann zunächst die OpenPLC-Runtime gestartet werden. Diese ermöglicht den Zugriff auf die OpenPLC-Weboberfläche. Mit einem Browser kann sie über die Adresse `http:`

`//localhost:8080` aufgerufen werden. Nachdem sich mit den Standarddaten angemeldet wurde (username & password: openplc), sollte zunächst ein eigenes Benutzerkonto über den Menüpunkt *Users* angelegt werden.

Wurde sich mit dem neuen Konto angemeldet, wird das *Hardware*-Menü geöffnet. Dort wird als Hardware-Layer *Simulink* ausgewählt. Daraufhin wird in der *Hardware Layer Code Box* die modifizierte Header-Datei `custom_layer.h`[1] angezeigt. Sie beinhaltet die wesentlichen Einstellungen, die die spätere Kommunikation mit Simulink betreffen. Zuerst sollte eingestellt werden, wie viele digitale und analoge Ports genutzt werden sollen. Da im Falle des Paketeschiebers keine analogen Ports genutzt werden, können die Werte `AINPUTS` und `AOUTPUTS` auf null gesetzt werden. Die Anzahl der digitalen Ports entspricht der Anzahl der genutzten Adressen, d.h. `DINPUTS 6` und `DOUTPUTS 3`.

Die nächsten Einstellungen legen die Parameter der UDP-Kommunikation zwischen Simulink und OpenPLC fest. Sollen OpenPLC und Simulink gemeinsam auf einem PC ausgeführt werden muss zunächst die Zeile `#define SIM_NET` durch `//` auskommentiert werden (default). Ist das nicht der Fall, müssen unter `SERVER_IP` die IP-Adresse des genutzten Netzwerkadapters und unter `Simulink_IP` die IP-Adresse des Rechners eingestellt werden, auf dem der Paketeschieber simuliert werden soll.

5.4.3 Kompilieren des Programms

Das generierte Programm liegt nun immer noch als Strukturierter Text vor. Dieser wird nun über die OpenPLC-Weboberfläche in ein lauffähiges Programm kompiliert. Über den Menüpunkt *Programs* wird das Programm zunächst ausgewählt und die Auswahl per *Upload Program* bestätigt. Anschließend wird das Programm benannt und hochgeladen.

Es muss berücksichtigt werden, dass OpenPLC von der hochgeladenen Datei eine Kopie erstellt. Das hat zur Folge, dass Änderungen am Programm nur durch Löschen und erneutes Hochladen des Programms übernommen werden können. Die *Reload program*-Schaltfläche hat in diesem Zusammenhang keine Funktion.

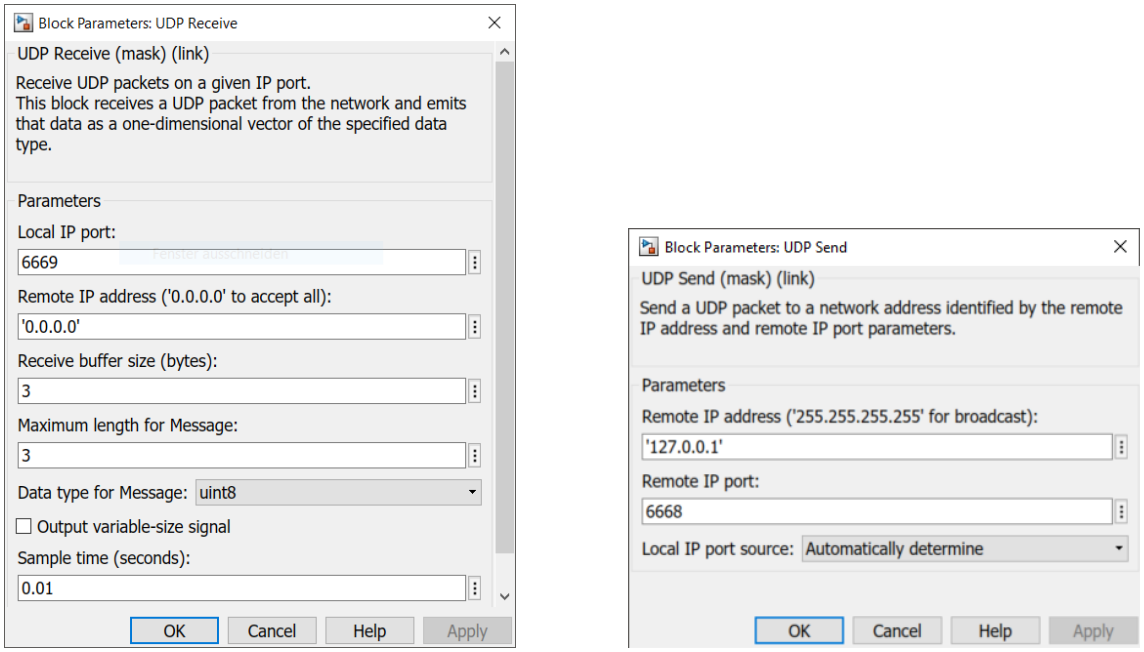
[1] zur Installation siehe Anhang A.3; zur Funktion der Simulink/OpenPLC-Schnittstelle siehe Anhang C

6 Virtuelle Inbetriebnahme

6.1 Vorbereitung des Simulink-Modells

Wie im letzten Kapitel beschrieben, werden zur Kommunikation zwischen dem Paketeschieber-Simulink-Modell und dem OpenPLC-Steuerungsprogramm Datagramme über UDP-Verbindungen verschickt. Das Simulink-Modell muss daher um diese Schnittstellen erweitert werden. Sie ersetzen das Stateflowchart, das dem Steuerungsprogramm zu Grunde liegt.

Die UDP-Verbindungen werden über die Blöcke `UDP Send` und `UDP Receive` aus der *DSP System Toolbox* realisiert. Damit diese ordnungsgemäß funktionieren, müssen sie entsprechend konfiguriert werden. Die Konfigurationen sind im Bild 6.1 abgebildet.



(a) `UDP Receive` **(b)** `UDP Send`

Bild 6.1: Konfiguration der UDP-Verbindungsblöcke

UDP receive Der *Local IP port* entspricht dem `SIMULINK_PORT`, der im Simulink-Hardware-Layer des OpenPLC-Webinterfaces konfiguriert wurde. Die *Remote IP adress* kann in der Regel auf dem Standardwert '0.0.0.0' belassen werden. Soll eine virtuelle Inbetriebnahme über ein Netzwerk vorgenommen werden, kann es sinnvoll sein, die IP-Adresse des PCs anzugeben, auf dem das Steuerungsprogramm ausgeführt wird. Die Größen des Empfangsbuffers sollte der Länge des zu empfangenden Datagramms entsprechen. Deren Länge in Bytes errechnet sich zu

$$n = 2 * \texttt{AOUTPUTS} + \texttt{DOUTPUTS}$$

bzw. im Falle des Paketeschiebers

$$n = 2 * 0 + 3 = 3$$

Der Datentyp der Nachricht ist `uint8`. Dadurch sind die Einstellungen für Empfangspuffer und maximale Nachrichtenlänge identisch. Durch die begrenzte Nachrichtenlänge und Größe des Empfangspuffers wird dieser von neu eintreffenden Daten überschrieben, wodurch der Simulation nur die aktuellsten Daten zur Verfügung gestellt werden. Die *Sample time* des Block sollte unter der gewählten Zyklus des Steuerungsprogramms liegen.

UDP send Der `UDP send`-Block besitzt lediglich drei Parameter. Die Änderung der IP-Adresse ist, wie beim `UDP receive`-Block, nur vonnöten, wenn das Steuerungsprogramm auf einem PC innerhalb des Netzwerk ausgeführt wird. Der *Remote IP port* entspricht auch in diesem Fall der entsprechenden Konfiguration des Simulink-Hardware-Layers (`SERVER_PORT`). Der letzte Parameter, *Local IP port source*, wird auf den Wert *Autmatically determine* eingestellt.

Byte (Un-)Packing Das empfangene Datagramm ist eine Aneinanderreihung der einzelnen Ausgangswerte der Steuerung. Um diese wieder in einzelne Werte zu zerlegen, wird der Block `Byte Unpacking` aus der *Simulink Real Time*-Toolbox genutzt. Er ermöglicht es, die einzelnen Bytes der Nachricht wieder analogen `uint16` und digitalen `uint8` Signalen zuzuordnen oder auch Arrays aus ihnen zu bilden. Am Beispiel des Paketeschiebers ergibt sich als erster Parameter eine Liste der gewünschten drei digitalen Ausgänge: `{'uint8', 'uint8', 'uint8'}`. Der zweite Parameter gibt die Dimension der jeweiligen Ports an. In diesem Fall sind alle drei Ports eindimensional: `{[1], [1], [1]}`

Auf die gleiche Weise wird auf der `UDP Send`-Seite mit dem entsprechenden `Byte Packing`-Block verfahren. Hier werden sechs `uint8`-Ports benötigt. Die ersten vier Ports werden von den Endschaltern des Paketeschiebers belegt. Der fünfte Port ist dem Paketesensor zugeordnet. Der letzte Port wird benötigt, um den vom PLC-Coder erzeugten Reset-Zustand zu steuern.

Da die digitalen Ein- und Ausgänge als Bytes vom Typ `uint8` verschickt und empfangen werden, ist eine Typkonvertierung zu den boolschen Signalen des Paketeschieber-Modells notwendig. Diese kann mit einem `Data Type Conversion`-Block durchgeführt werden.

Wurde das Paketeschieber-Modell wie beschrieben erweitert, ergibt sich der in Bild 6.2 gezeigte Aufbau.

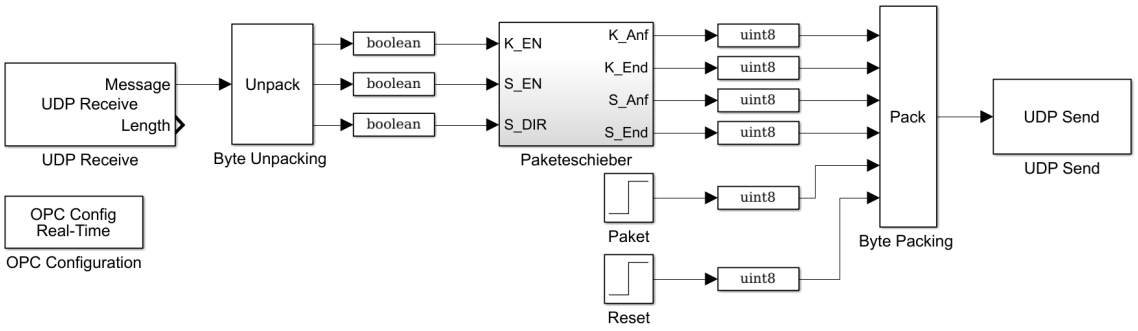


Bild 6.2: Um UDP-Schnittstellen erweitertes Simulink-Modell des Paketeschiebers

6.1.1 Entschleunigung der Simulation

Wird die Simulation des Modells gestartet fällt auf, dass die Zeit zur Berechnung der Simulation kürzer ist, als die Simulationszeit selbst. Diese Beschleuningung ist bei der Virtuellen Inbetriebnahme ein Störfaktor, da die Zykluszeit und die Geschwindigkeit der Übertragung gegenüber einer „wirklichen“ Inbetriebnahme nicht wesentlich oder nicht in gleichem Maße beschleunigt sind. Aus diesem Grund ist es sinnvoll, die Simulation auszubremsen, sodass sie in normaler Geschwindigkeit abläuft. Dies gelingt mit dem `OPC config Real-Time`-Block aus der *OPC*-Toolbox. Er benötigt keine Konfiguration und muss dementsprechend nur im Simulink-Modell platziert werden.

6.2 Virtuelle Inbetriebnahme

Nachdem sowohl OpenPLC als auch das Simulink-Modell konfiguriert wurden, kann die virtuelle Inbetriebnahme erfolgen. Dazu wird lediglich die *Start PLC*-Schaltfläche im OpenPLC-Webinterface gedrückt und die Simulation in Simulink gestartet. Sofern die Simulationszeit im Modell auf *inf* gestellt wurde, spielt die Reihenfolge des Startens keine Rolle.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde zunächst ein Demonstrationsbeispiel erarbeitet. Dieses umfasst die erdachte Maschine *Paketeschieber*, die ein Paket mittels eines Pneumatikzylinders und eines Linearantriebs befördert. Dazu wurde ein Steuerungskonzept für den Paketeschieber entwickelt.

Auf Basis des Demonstrationsbeispiels erfolgte eine 3D-CAD-Konstruktion des Paketeschiebers in Autodesk Inventor. Aus dieser wurde anschließend, über das Simscape Multibody Link Plugin, ein Simulink-Modell generiert, welches um Aktoren und Sensoren erweitert wurde.

Nachdem der Paketeschieber als virtuelle Maschine zur Verfügung stand, konnte das Steuerungskonzept in Stateflow modelliert werden. Daraufhin folgte die erste Co-Simulation von Maschine und Steuerung in Simulink. Mithilfe des Simulink PLC Coders wurde aus dem Stateflow-Chart ST-Code erzeugt, der die Grundlage für das OpenPLC-Steuerungsprogramm darstellt. Dieses wurde mit dem PLCOpenEditor erstellt und exportiert.

Nachdem das Simulink-Modell durch UDP-Schnittstellen erweitert und OpenPLC kofiguriert wurde, konnte die virtuelle Inbetriebnahme des Paketeschiebers erfolgen.

In Ermangelung einer Schnittstelle zwischen OpenPLC und Simulink, wurde der Simulink-Hardware-Layer derart modifiziert, dass er eine direkte Kommunikation der beiden Anwendungen über UDP-Verbindungen gestattet.

7.1 Ausblick

Die Arbeit mit der vorgestellten Toolkette hat einige Stärken, aber auch Fallstricke und Schwächen derselben aufgezeigt. Die Stärken der beschriebenen Lösung liegen zum einen in der Einfachheit, mit der ein funktionsfähiges Simulationsmodell aus einer CAD-Konstruktion erzeugt werden kann. Zum anderen können mithilfe von

Stateflow sehr schnell Steuerungsdesigns modelliert und an der virtuellen Maschine getestet werden.

Der Export von CAD-Modellen ist jedoch mitunter Fehlerbehaftet oder kann – im Fall von Onshape – gar nicht erfolgen. Es bleibt zu überprüfen, ob diese Probleme in aktuelleren Matlab- und somit auch SMLink-Versionen weiterhin auftreten.

Mit dem Simulink PLC-Coder exportierte PLCOpenEditor-Projekte waren mit der aktuellen OpenPLC-Version nicht kompatibel. Es bleibt zu untersuchen, ob dieses Problem mit einer aktuelleren Matlab-Version weiterhin auftritt. Zudem wäre es denkbar, das fertige Steuerungsprogramm automatisch aus dem generischen ST-Code zu generieren, statt diesen händisch in ein PLCOpenEditor-Projekt einzupflegen.

Die Kommunikation zwischen Simulink und OpenPLC konnte mithilfe des modifizierten Hardware-Layers hergestellt werden. Diese Lösung biete jedoch noch Optimierungsbedarf. So könnten Optionen ergänzt werden, die die Kommunikationen mit mehreren Simulink-Modellen zulassen und die Portbelegung vereinfachen. Darüber hinaus steht inzwischen das OpenPLC-Simulink-Interface des OpenPLC-Autors Thiago Alves zur Verfügung, welches zukünftig erprobt werden kann.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

AS	Ablaufsprache
AWL	Anweisungsliste
CAD	Computer Aided Design
FBS	Funktionsbausteinsprache
HSF	Hard Stop Friction
KOP	Kontaktplan
POE	Programmorganisationseinheit
SMML	Simscape Multibody Multiphysics Library
SPS	speicherprogrammierbare Steuerung
ST	Strukturierter Text
STEP	Standard for the exchange of product model data
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VIBN	Virtuelle Inbetriebnahme
XML	Extensible Markup Language

Literaturverzeichnis

[1] Alves, Thiago: *OpenPLC on Windows*. https://www.openplcproject.com/getting-started-windows, Abruf: 2019-05-20

[2] Bracht, Uwe ; Geckler, Dieter ; Wenzel, Sigrid: *Digitale Fabrik*. 2. Aufl. Berlin, Heidelberg : Springer Vieweg, 2018. – ISBN 978–3–662–55783–9

[3] In: Eigner, Martin: *Einleitung – Modellbasierte Virtuelle Produktentwicklung*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2014. – ISBN 978–3–662–43816–9, 1–13

[4] Glöckler, Michael: *Simulation mechatronischer Systeme*. Springer Fachmedien Wiesbaden, 2018. – ISBN 978–3–658–20703–8

[5] John, Karl H. ; Tiegelkamp, Michael: *SPS-Programmierung mit IEC 61131-3*. 4. Aufl. Springer-Verlag Berlin Heidelberg, 2009. – ISBN 978–3–642–00269–4

[6] Lerche, Jens: *Virtuelle Inbetriebnahme mit Simulink: Streckenmodellierung, Reglerentwurf und Systemsimulation*. MathWorks, https://de.mathworks.com/videos/virtual-commissioning-with-simulink-122941.html, Abruf: 2019-05-20

[7] MathWorks (Hrsg.): *Export an Autodesk Inventor Robot Assembly Model*. MathWorks, https://de.mathworks.com/help/physmod/smlink/ug/installing-and-linking-simmechanics-link-software.html, Abruf: 2019-05-20

[8] MathWorks (Hrsg.): *Install the Simscape Multibody Link Plug-In*. MathWorks, https://de.mathworks.com/help/physmod/smlink/ug/export-robot-assembly-from-autodesk-inventor-software.html, Abruf: 2019-05-20

[9] Miller, Steve: *Simscape Multibody Multiphysics Library*. MATLAB Central File Exchange, 2019. https://de.mathworks.com/matlabcentral/fileexchange/37636-simscape-multibody-multiphysics-library, Abruf: 2019-05-20

A Einrichten der Toolkette

In diesem Kapitel wird auf die Einrichtung der Toolkette in einer Windows-Umgebung eingegangen. Die Installationsdateien der genutzten Software liegen dieser Arbeit auf einem Datenträger bei. Auf diesem befinden sich, in dem Ordner *Toolchain Setup*, Unterordner, die entsprechend der Installationsreihenfolge durchnummeriert sind.

Im Folgenden werden die entsprechenden Bezugsquellen der Software angegeben. Es wird jedoch empfohlen, die beiliegenden Dateien zu nutzen, um Kompatibilitätsprobleme mit eventuell neueren Versionen zu vermeiden.

A.1 Inventor & Simscape Multibody Link

Der Export des 3D-CAD-Modells aus Autodesk Inventor erfolgt mithilfe des *Simscape Multibody Link*-Plugins. Dieses kann über die MathWorks-Webseite heruntergeladen werden.[1] Dabei ist darauf zu achten, die richtige Variante des Plugins, für die verwendete Matlab-Version zu verwenden. Zu jeder Variante gibt es eine ZIP-Datei und ein m-Skript. Diese müssen sich zur Installation in einem gemeinsamen Ordner befinden. Anschließend wird Matlab mit Administratorrechten gestartet. Über das Command Window wird das Skript mit dem Dateinamen der ZIP-Datei aufgerufen (z. B. `install_addon 'smlink.r2017a.win64.zip'`). Die Installation erfolgt daraufhin automatisch.

Um das Plugin aus Inventor heraus nutzen zu können, ist es noch nötig, es zu aktivieren. Dazu wird im Command Window `smlink_linkinv` eingegeben. Nun sollte das SMLink-Plugin im Menü Zusatzmodule in Inventor zur Verfügung stehen. Es kann jedoch passieren, dass Inventor das unbekannte Plugin zunächst blockiert. In diesem Fall muss die Ausführung des Plugins in dem aufgehenden Meldefenster noch erlaubt werden.

A.2 SMML-Bibliothek

Um die in Abschnitt 4.4.3 vorgestellten HSF-Blöcke nutzen zu können, muss die SMML zunächst heruntergeladen und entpackt werden.[2] Der Bibliothek liegt ein

[1] `https://de.mathworks.com/campaigns/offers/download_smlink.html` (letzter Aufruf: 03.06.2019)

[2] `https://de.mathworks.com/matlabcentral/fileexchange/37636-simscape-multibody-multiphysics-library` (letzter Aufruf: 03.06.2019)

Matlab-Skript namens `startup_sm_ssci.m` bei. Durch Ausführen des Skripts öffnet sich eine Web-Oberfläche, die oben die zwei enthaltenen Bibliotheken auflistet, die eigentliche *Simscape Multibody Multiphysics Library* und die *Simscape Forces Library*. Wird die SMML angewählt, öffnet sich ein Simulink-Modell, in dem die Blöcke der Bibliothek zur Verfügung stehen.

Es hat sich als praktisch erwiesen, den Ordner *library* in das Matlab-Arbeitsverzeichnis zu kopieren und das Skript `Systemparameter.m` um die entsprechenden Zeilen aus dem Startup-Skript zu erweitern. Diese sind als kopiert gekennzeichnet.

A.3 OpenPLC

OpenPLC wird zunächst gemäß des Leitfadens *OpenPLC on Windows*[3] installiert. Danach stehen zum einen die OpenPLC-Runtime, samt Webinterface und zum anderen der PLCOpenEditor zur Verfügung. Der Leitfaden erklärt in Kurzform auch die Nutzung der beiden Programme.

Nach der Installation von OpenPLC müssen noch die modifizierten Hardware-Layer-Dateien in die entsprechenden Verzeichnisse eingefügt werden. Es bietet sich an vor dem Überschreiben Sicherungskopien der originalen Dateien anzulegen. Der Standardinstallationspfad für OpenPLC ist *C:/Users/NUTZERNAME/OpenPLC/*. Die modifizierte `custom_layer.h`-Datei wird dort in dem Unterordner */Runtime/home/OPENPLCNUTZER/OpenPLC_v3/webserver/core/* eingefügt. In dem Unterordner *hardware_layers* wird dann die modifizierte `simulink.cpp`-Datei eingefügt. Beide Dateien sind auf dem beiliegenden Datenträger im Ordner */Toolchain Setup/4. OpenPLC Hardware-Layer/* zu finden.

[3] `https://www.openplcproject.com/getting-started-windows` (letzter Aufruf: 03.06.2019)

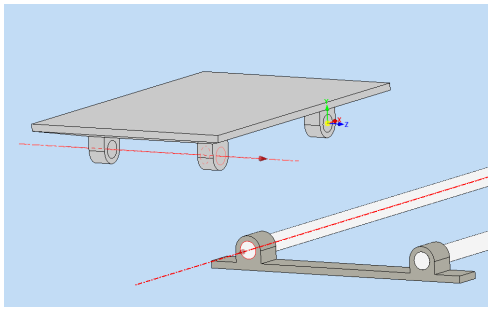
B Abhängigkeiten & Verbindungen der CAD-Komponenten

In Abschnitt 4.1 wurde bereits eine erprobte Methode zum Erstellen von Verbindungen und Abhängigkeiten vorgestellt. Im Folgenden wird diese Methode nochmals am Beispiel der Verbindung Schlitten/Schiene illustriert dargestellt.

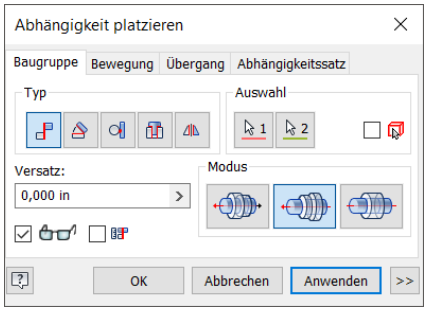
In Bild B.1 ist zu sehen, wie die entsprechenden Abhängigkeiten erstellt werden. Dazu wird zunächst das Werkzeug *Abhängig machen* über die Menüleiste ausgewählt (*Zusammenfügen/Beziehungen/Abhängig machen*). Nachdem sich das Dialogfenster geöffnet hat, ist standardmäßig der Abhängigkeitstyp *passend* angewählt. Mit diesem wird zunächst die zylindrische Fläche einer Linearführung ausgewählt. Anschließend die zylindrische einer Bohrung im Schlitten, die später auf der Linearführung gleiten soll (siehe Bild B.1a). Danach muss gegebenfalls die Orientierung der beiden Komponenten zueinander angepasst werden. Die Zuweisung der zweiten Schlittenseite erfolgt anschließend analog zur Ersten.

Der Schlitten lässt sich nun bereits auf der Schiene bewegen, gleitet jedoch über die Schienenenden hinweg. Durch hinzufügen einer Verbindung des Typs *Zylindrisch* können nun die Grenzwerte der Bewegung eingestellt werden. Dazu wird das Werkzeug *Verbindung* aufgerufen. Auch mit diesem Werkzeug werden die zylindrischen Flächen jeweils einer Linearführung und Schlittenbohrung ausgewählt. Dabei ist auf die Orientierung der Kreisfläche am Cursor zu achten. Diese ist in Bild B.2a als grauer Kreis mit grünem Mittelpunkt zu sehen. Die Normale der Kreisfläche sollte bei der Auswahl in die spätere Bewegungsrichtung ausgerichtet sein.

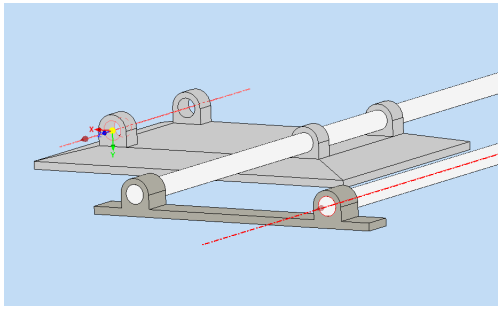
Im Reiter *Grenzwerte* können nun die Grenzen der Bewegung des Schlittens auf der Schiene eingestellt werden. Dies ist hilfreich um den Schlitten vor dem Export des Modells genau auf seine Endlage zu positionieren. Da es viele unterschiedliche Kombinationen von Beziehungen und damit einhergehenden Bauteilorientierungen gibt, müssen die entsprechenden Bewegungsgrenzen empirisch bestimmt werden. Die Begrenzung des Drehwinkels ist in diesem Fall nicht zwingend erforderlich, da der rotatorische Freiheitsgrad durch die zweite Führung unterbunden wird.



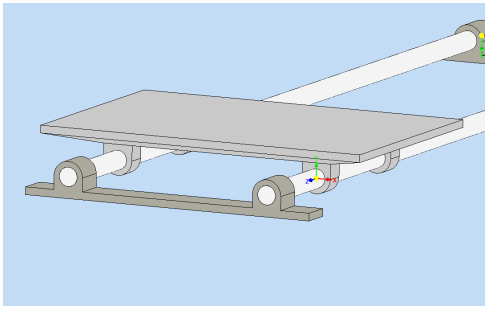
(a) Auswahl der Flächen



(b) Wählen der Ausrichtung

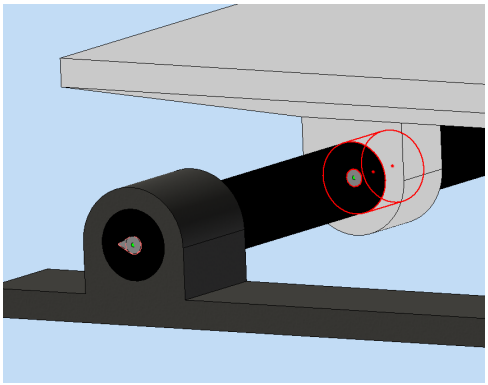


(c) Auswählen der zweiten Seite

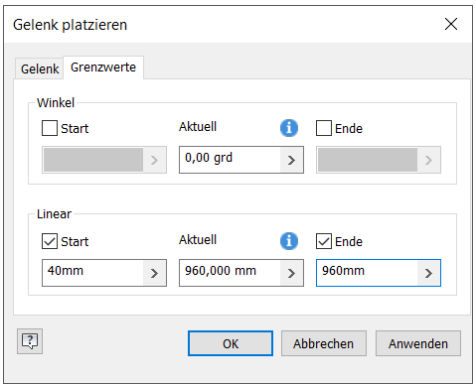


(d) Ergebnis der erzeugten Beziehungen

Bild B.1: Erstellen der nötigen Abhängigkeiten zwischen Schiene und Schlitten



(a) Auswahl der Flächen wie in B.1



(b) Einstellen der Bewegungsgrenzen

Bild B.2: Erstellen der Verbindung zwischen Schiene und Schlitten mit Bewegungsgrenzen

C Die Simulink/OpenPLC-Schnittstelle

Als Soft-SPS unter Windows besitzt OpenPLC keine physischen Schnittstellen, um eine Verbindung zwischen dem Steuerungsprogramm und anderen Systemen herzustellen. Zu diesem Zweck werden in der Weboberfläche, im Menü *Hardware*, zahlreiche sogenannte *OpenPLC Hardware Layer* bereit gestellt. Sie ermöglichen es, eine Verbindung zu einer physischen Plattform herstellen.

Unter den möglichen Hardware Layern findet sich auch der Simulink-Layer. Dieser ermöglicht jedoch **keine** unmittelbare Verbindung zwischen Simulink und OpenPLC. Stattdessen wird das Programm *OpenPLC-Simulink-Interface* benötigt, welches die Synchronisierung der Ein- und Ausgangsdaten zwischen Simulink und OpenPLC vornimmt. Dieses Programm stand zum Zeitpunkt der Ausarbeitung noch nicht zur Verfügung.[1] Aus diesem Grund wurde der Simulink-Layer den Anforderungen entsprechend modifiziert.

C.1 Aufbau des Simulink-Layers

Die entsprechende Layer-Datei `simulink.cpp` ist im OpenPLC Ordner `OpenPLC/-Runtime/home/USERNAME/OpenPLC_v3/webserver/core/hardware_layers` zu finden. Das Studium des Quellcodes offenbart die in Bild C.1 dargestellte Funktionalität. Der Kern der Datei ist die Funktion `*exchangeData(void *arg)`. Sie enthält eine Endlosschleife, in der ständig die Daten mit dem OpenPLC-Simulink-Interface über eine UDP-Verbindung ausgetauscht werden. Dabei werden die empfangenen Daten direkt in die (virtuellen) Register der SPS gespeichert und die zu sendenden Daten entsprechend von dort gelesen. Die `*exchangeData`-Funktion wird in einem eigenen Thread aufgerufen, um unabhängig vom SPS-Programm operieren zu können.

[1] Im Februar 2019 unterlag das *OpenPLC-Simulink-Interface* des Autors Thiago Alves einem Patent. Gegenwärtig ist es wieder unter `https://github.com/thiagoralves/OpenPLC_Simulink-Interface` zugänglich.

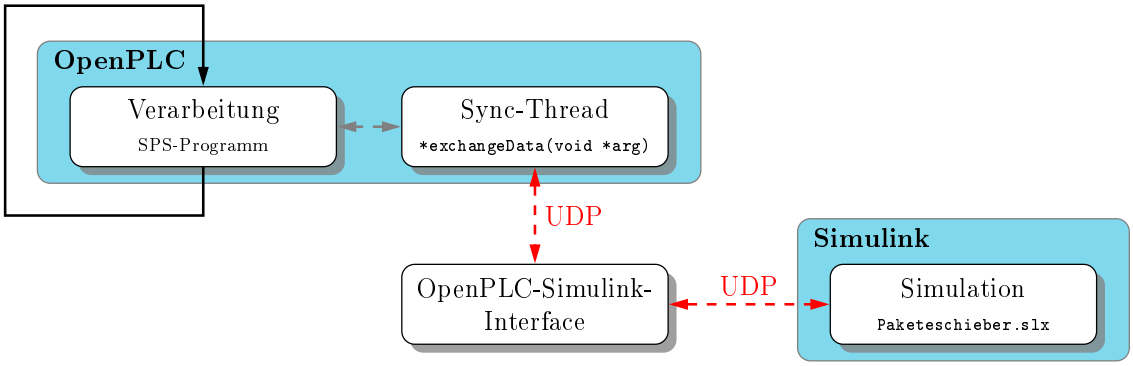


Bild C.1: Prinzip des originalen Simulink-Hardware-Layers

C.2 Modifikation des Simulink-Layers

Da das OpenPLC-Simulink-Interface nicht genutzt werden konnte, wurde auf Basis der gegebenen Funktionalität ein eigenes Konzept zur Synchronisierung der Daten erdacht. Dieses sieht vor, die Daten direkt, ohne das OpenPLC-Simulink-Interface, zwischen Simulink und OpenPLC zu übertragen. Dazu sollte der `*exchange`-Thread aus der originalen Datei genutzt werden. Darüber hinaus wurde darauf Wert gelegt, dass das ursprüngliche EVA-Prinzip der SPS wieder klarer ersichtlich wird. Dieses besagt, dass Daten von der SPS der Reihe nach **E**ingelesen, **V**erarbeitet und **A**usgegeben werden. Dazu werden die im originalen Simulink-Hardware-Layer leeren Funktionen `updateBufferIn()` und `updateBufferOut()` verwendet. Das daraus resultierende Konzept ist in Bild C.2 zu sehen.

Damit auf die Nutzung des OpenPLC-Simulink-Interface verzichtet werden kann, muss der Synchronisations-Thread nicht nur den bloßen Austausch von Datenpaketen abwickeln. Stattdessen muss dem Nutzer die Möglichkeit gegeben werden, die Kommunikationsparameter und die Anzahl der genutzten SPS-Ports einzustellen. Diese Einstellungen wurden durch Präprozessordirektiven in der Header-Datei `custom_layer.h` realisiert (siehe Listing C.1). Diese wird beim Einstellen des Hardware-Layers in der OpenPLC-Weboberfläche angezeigt. Dadurch können die Einstellungen benutzerfreundlich vorgenommen werden.

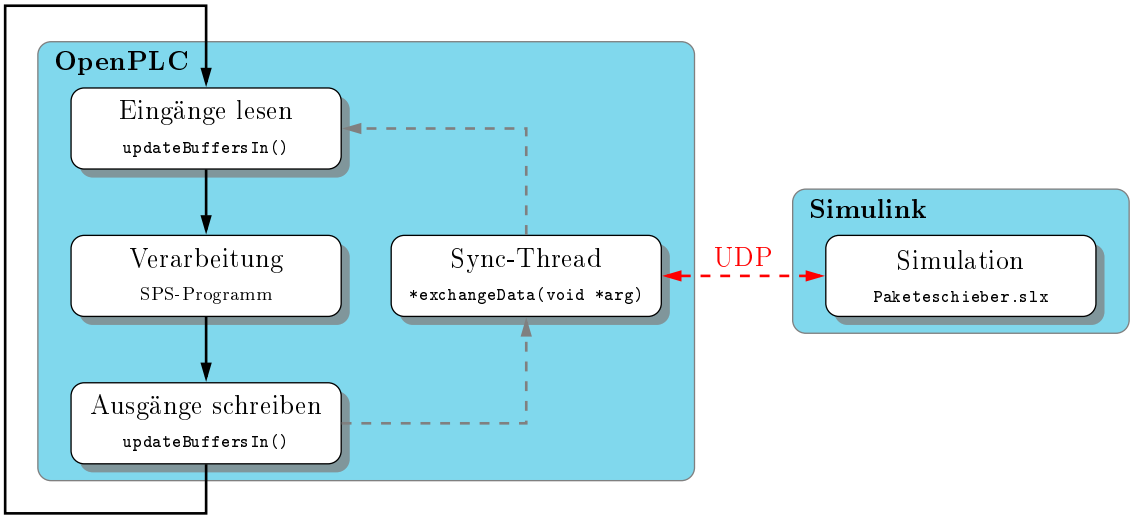


Bild C.2: Prinzip des modifizierten Simulink-Hardware-Layers

```
15 #define SERVER_PORT     6668 // set in Simulink UDP Send block
16 #define SIMULINK_PORT   6669 // set in Simulink UDP Receive block
17
18 /*#define SIM_NET /* if defined, OpenPLC and Simulink can run on different
       machines; then the IPs of the machines have to be defined below */
19 #ifdef SIM_NET
20 #define SERVER_IP       "192.168.1.6"
21 #define SIMULINK_IP     "192.168.1.9"
22
23 #else
24 #define SERVER_IP       "127.0.0.1" // Loopback adress for local communication!
25 #define SIMULINK_IP     "127.0.0.1" // Do not change!
26 #endif
27
28 /* number of IOs used by SIMULINK (maximum can be edited in simulink.cpp) */
29 #define AINPUTS         0       // max. 8       | corresponds to %IW0 to %IW15
30 #define AOUTPUTS        0       // max. 8       | corresponds to %QW0 to %QW15
31 #define DINPUTS         6       // max. 16      | corresponds to %I0.0 to %I0.15
32 #define DOUTPUTS        3       // max. 16      | corresponds to %Q0.0 to %I0.15
```

Listing C.1: Auschnitt aus der Headerdatei `custom_layer.h`, konfiguriert zur lokalen Kommunikation mit dem Paketeschieber-Simulink-Modell

Zuerst wird die Portkonfiguration vorgenommen. Die Portnummern können zunächst beliebig gewählt werden, sollten jedoch deutlich größer als 1024 sein, um Konflikte mit anderen Programmen und Diensten zu vermeiden. Der Serverport ist der Port, an dem OpenPLC die Daten von Simulink empfängt. Der Simulinkport ist dementsprechend der Port, an dem Simulink Daten von OpenPLC empfängt.[2] Als nächstes werden die IP-Adressen konfiguriert. Dazu muss zwischen zwei Fällen unterschieden werden: Werden OpenPLC und das Simulink-Modell auf dem selben PC ausgeführt, so muss das `#define SIM_NET` auskommentiert werden (default). In diesem Fall kann

[2] siehe dazu auch die Kommentare in der Headerdatei

mit dem Einstellen der Ein- und Ausgänge fortgefahren werden. Sollen OpenPLC und Simulink jedoch auf getrennten PCs im selben Netzwerk ausgeführt werden, so ist es nötig die IP-Adressen der verwendeten Netzwerkadapter entsprechend der vorgegebenen Standardwerte einzutragen.

Im Anschluss daran wird die Anzahl der genutzten SPS-Ports eingestellt. Da ungenutzte Ports entweder zu fehlerhaften Übertragungen oder zu unübersichtlichen Simulink-Modellen führen, sollten immer nur die Anzahl der tatsächlich verwendeten Ports eingestellt werden.

Im Falle der Paketeschieber-Steuerung ergeben sich sechs digitale Eingänge, durch die vier Endlagenschalter, den Paketeschalter und die Reset-Funktion. Dazu kommen drei digitale Ausgänge zur Ansteuerung der Aktoren.

C.2.1 Format des Datagramms

Die Anzahl der eingestellten Ein- und Ausgänge bestimmt den Aufbau des Datagramms, dass vom `*exchange`-Thread verschickt und zum Empfang erwartet wird. Dies muss bei der späteren Erweiterung des Simulink-Modells berücksichtigt werden. Der Aufbau eines Datagramms richtet sich nach folgenden Regeln:

- analoge Daten sind vom Typ `uint16`
- digitale Daten sind vom Typ `uint8`
- ein Datagramm beginnt mit den analogen Daten

In Bild C.3 sind beispielhaft Datagramme für je zwei analoge und drei digitale Ein- und Ausgänge zu sehen. Bei anderen Ein- und Ausgangskonfiguration werden die Datagramme entsprechend erweitert oder verkürzt. So fehlen zum Beispiel die analogen Daten beim Paketeschieber gänzlich.

Eingangsdatagramm

anIn0		anIn1		digIn0	digIn1	digIn2
Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
uint16		uint16		uint8	uint8	uint8

Ausgangsdatagramm

anOut0		anOut1		digOut0	digOut1	digOut2
Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
uint16		uint16		uint8	uint8	uint8

Bild C.3: Datagramme bei jeweils zwei analogen und drei digitalen Ein- und Ausgängen

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Fassung entspricht der auf dem Medium gespeicherten Fassung.

Ort, Datum, Unterschrift