Diplomarbeit

Automatisierte Parameteridentifikation einer permanenterregten Synchronmaschine

Werner Bergmayr

Institut für elektrische Antriebstechnik und Maschinen

Technische Universit Graz

Vorstand: Univ.-Prof. Dr.-Ing. Annette Mütze

Betreuer/Begutachter: Ass.Prof. Dipl.-Ing. Dr.techn. Roland Seebacher

Kurzfassung

Die vorliegende Arbeit behandelt die Parameteridentifikation an einer permanenterregten Synchronmaschine. Es werden Ansätze zur bewegungslosen Identifikation der wichtigsten Paramterer einer PMSM geliefert. Diese Ansätze werden auf einem Mikrocontroller realisiert und getestet. Zur Entwicklung der Funktionen wird eine Softwareentwicklungskette vorgestellt mit der man aus MATLAB/Simulink direkt Code entwerfen und kompilieren kann.

Dazu wird gezeigt wie man in MATLAB eine Entwicklungsumgebung aufsetzt und so konfiguriert mit der man problemlos eine Motorregelung betreiben kann. Zusätzlich wird erklärt wie man Schnittstellen wie den CAN-Bus anspricht und konfiguriert.

Es wird ein Grundwellenmodell einer PMSM entwickelt und die Parameteridentifikationsmethoden werden in Simulink modellbasiert entwickelt. Es werden dabei bereits in der Simulation Begrenzungen durch das Datenformat und die Abtastzeit eingebracht. Die auf diese Weise entwickelten Funktionen können direkt auf einem DSP zur Anwendung gebracht werden.

Die Implementierung wird mit einem spezifischen DSP aufgezeigt, eine Schnittstelle zur Messung konfiguriert und erklärt, und die Ergebnisse dokumentiert.

Es wird ausführlich über die richtige Auswertung eines Inkrementalencoders diskutiert und Fragen der Genauigkeit angesprochen.

Als Ergebnis wird eine Simulink Software-Bibliothek erstellt die ein ungeübter Anwender mit dieser Dokumentation einfach und richtig anwenden kann und die die Grundlage für weitere Entwicklungen in Simulink bilden soll.

Abstract

The work in hand deals with the topic of parameter identification of a permanent magnet synchronous motor. It presents an approach for motionless identification of the most important parameter of a PMSM. These approaches are being realized on a specific microcontroller and tested. A software-toolchain for the development of the functions is presented. With this toolchain it is possible to directly develop code from MATLAB/Simulink and compile it onto a microcontroller.

Therefor it is shown how the development environment is set up and configured in a way that it is possible to operate a motor control smoothly. Additionally it is shown how to address the CAN-bus interface and how it is configured.

The fundamental wave model of a PMSM is derived and used to develop the parameter-identification methods in Simulink modelbased. In this task it will be shown how to deal with limitations given by the data format of the microcontroller and the sample time of systems.

The implementation onto a specific DSP is shown and an interface will be set up for measurement, the measurements will be documented at the end.

A detailed discussion about the use of incremental encoders will be done and issues of accuracy are addressed.

The final result is a Simulink-library which should support an inexperienced user with setting up and configuring an environment as foundation for future developments. The library will allow an easy identification of a PMSM without previous knowledge about identification methods.

Eidestattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz,am ...05.03.2013..... .....

(Unterschrift)

Danksagung

Ich möchte mich bei meinen Diplombetreuern Herrn Dr. Roland Seebacher und Herrn Dr. Roland Greul sehr herzlich für ihre Unterstützung bedanken.

Sie haben mich bei dieser Arbeit immer wieder mit neuen Ideen unterstützt und mir geholfen mich auf das Wesentliche zu konzentrieren.

Vor allem bei Roland Greul möchte ich mich dafür bedanken, dass er immer positiv eingestellt war und mir oft geholfen hat, meine Motivation wieder zu finden.

Auch meinen Kollegen in der AVL List GmbH möchte ich für ihre Hilfe und die Gemeinschaft danken. Selbiges gilt für alle Freunde und Kollegen mit denen ich im TU Graz e-Power Racing Team arbeiten durfte.

Zu guter Letzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die während der Jahre des Studiums immer für mich da waren.

Vorwort

Die vorliegende Arbeit ist 2012/13 während meiner Tätigkeit in der AVL List GmbH entstanden.

Die Idee dazu entstand Ende 2011 in Diskussionen über die Anwendung von Leistungselektronik in der Fahrzeugantriebsbstechnik. Zu dieser Zeit war meine Aufgabe eine PSM als Antriebsmotor für den Elektrorennwagen des TU Graz e-Power Racing Team zu regeln. Durch den Kontakt mit permanenterregten Synchronmaschinen als Antriebsmotoren für Fahrzeuge und die Problemstellungen die sich dabei bei der Inbetriebnahme dieser Motoren ergeben, wurde der Gedanke gefasst eine Identifikationsmethode zu entwerfen, die relativ unabhängig von einer genau bestimmten Leistungselektronik einen unbestimmten Motor soweit identifizieren können soll, dass die grundlegenden Eigenschaften bestimmt sind und es damit möglich ist z.B. eine Stromregelung/Drehzahlregelung zu entwerfen.

Während meiner Tätigkeit in der AVL List GmbH ergab sich eine ähnliche Fragestellung bei der Anwendung von Leistungselektronik als Prüfstandstechnik. Dabei sollte es möglich gemacht werden, unbekannte Motoren nach einer kurzen Messung ohne dazugehörende Leistungselektronik als Prüfling ansteuern und regeln zu können. Im Antriebsstrang-Prüfstandsbereich wurde als Anforderung die Aufgabe definiert den Motor mit geringer mechanischer Bewegung identifizieren zu können und das trotz einer unbekannten Ausgangsposition bei Verwendung eines Inkrementalencoders.

In diesem Gedanken wurde Anfang 2012 entschieden, dazu eine Softwareentwicklungskette zu definieren und mit der Entwicklung zu beginnen. Um die Vorteile der modellbasierten Entwicklung nutzen zu können, kommt für die Entwicklung der Funktionen MATLAB/Simulink zum Einsatz. Die in Simulink erhältlichen Blocksets für verschiedenste Prozessoren ermöglichen Funktionen für eine spezifische DSP-Serie einheitlich zu entwickeln.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Permanenterregte Synchronmaschinen werden in den letzten Jahren immer öfter genannt, wenn es um Themen der hocheffizienten Antriebstechnik geht. Sie weisen allgemein die höchste Leistungsdichte unter den bekannten Standardmaschinen auf. Bei Synchronmaschinen wird die Position der Feldvektors direkt von der Rotorposition vorgegeben. Das Wissen über die Position des Feldvektors ermöglicht den Start der Maschine mit Maximalmoment.

Ein Start einer permanenterregten Synchronmaschine ist also im Allgemeinen nur bei vorhandenem Absolutwinkelgeber möglich. In den meisten industriellen Anwendungen finden sich aber optische Inkrementalencoder, da diese preislich weit günstiger sind als Resolver.

Um ohne Absolutwertgeber den Motor mit vollem Drehmoment starten zu können, muss die Rotorposition durch andere Methoden festgestellt werden. Man kann den Rotor zum Beispiel durch bestromen mit einem Gleichstrom in eine gewisse Position ausrichten oder den Fluss durch den Einsatz von Hallsensoren messen.

Ersteres ist jedoch mit dem Nachteil verbunden, dass man den Rotor dabei bewegen muss und dass diese Bewegung nicht abgeschätzt werden kann. Der Motor muss also frei beweglich sein und das mechanische System, das daran angekoppelt ist muss dies auch erlauben. Das ist gerade in der Anwendung in der automotiven Prüfstandstechnik nicht immer der Fall. Die Feststellung der Flussausrichtung durch Messung mit Hallsensoren ist in einfacher Art in ihrer Auflösung begrenzt und kann wiederum nur durch zusätzlichen Aufwand in der nötigen Genauigkeit erfolgen.

Bisherige Ansätze zur Rotorpositionsbestimmung im Stillstand arbeiten meistens mit Kalman Filtern, Beobachtern, Messung der Gegen-EMK oder Messung der magnetischen Unsymmetrie des Motors.

In einer industriellen kostengünstigen Anwendung wird man also normalerweise einen Inkrementalencoder als Winkelsensor am Motor finden, zusätzlich sind im Normalfall noch Phasenstrommessungen und die Messung der Zwischenkreisspannung gegeben.

In dieser Arbeit soll ein einfacher Ansatz zur Schätzung der Rotoranfangsposition ausgearbeitet und getestet werden, der zur Identifikation dieser Anfangsposition nur diese Standard-Messungen bzw. Schnittstellen verwendet und sich dabei auch auf einem kostengünstigen DSP ausführen lässt.

Zusätzlich soll der Ständerwiderstand und die Ständerinduktivität an der Maschine identifiziert werden. Aus den gefundenen Daten kann man danach einem PI-Regler parametrieren bzw. entwerfen und anschließend eine Stromregelung realisieren.

In [3] wird geschätzt dass 98% der Controller in der Papier- und Zellstofftechnik SISO PI-Controller sind und dass davon 65% schlecht eingestellt sind. Das hat meistens den Grund, dass über die zu regelnde Strecke wenig bzw. nicht genügend Wissen vorliegt. Nimmt man für eine permanenterregte Synchronmaschine einen magnetisch symmetrischen Aufbau an, kann durch die Bestimmung des Offset, des Widerstandes und der Längsinduktivität der Motor zum Großteil bedatet werden. Zieht man noch Wissen über die Geometrie und die verwendeten Magneten in Betracht, so wäre es möglich den Motor damit sogar komplett zu bedaten.

Zur Auslegung einer Drehmomenten/Drehzahl-Regelung einer PMSM, der Behandlung von Nichtlinearitäten und der Regelstruktur sei auf [4] verwiesen.

Im Folgenden wird kurz der Aufbau der Arbeit erläutert und die nötigen Grundlagen werden vermittelt.

1.1 Gliederung der Arbeit

In den folgenden Kapiteln wird versucht dem Leser ein komplettes Bild von der verwirklichten Idee und der Ausführung zu geben.

Im ersten Abschnitt wird kurz eine Einführung in die Raumzeigertheorie gegeben, dies soll jedoch nur als Auffrischung dienen. Dazu sei auch auf [1], [4], [5], und [6], verwiesen.

Danach werden die Differentialgleichungen für die allgemeine Drehfeldmaschine und daraufhin das Grundwellenmodell der PMSM beschrieben.

Die Ideen zu den Versuchen und ihre Struktur folgt in Kapitel 3, hier soll verständlich erklärt werden was die Grundgedanken beim Entwurf der Versuche waren und wie die Ausführung aussieht.

Im Kapitel 4: Software werden die entworfenen Funktionsblöcke, die zur Erstellung der Versuche notwendig sind, erklärt. Es wird genau auf die Auswertung des Inkrementalencoders eingegangen und die Genauigkeit der Drehzahlbestimmung diskutiert.

Es werden alle entwickelten Funktionen vorgestellt, erklärt wie man die nötigen Ein- und Ausgangsgrößen erhält und wie sie skaliert sind. Die Blockschaltbilder aus Simulink werden gezeigt und ihre Funktion erklärt.

Zum Abschluss des Kapitels wird gezeigt, wie die Parameteridentifikationsversuche implementiert wurden.

Im Kapitel 5 wird die richtige Konfiguration der Entwicklungsumgebung gezeigt.

Am Ende wird über die Reglerparametrierung gesprochen und die Messergebnisse präsentiert.

2 Grundlagen

In diesem Kapitel soll eine kurze Einleitung in die Grundlagen der Theorie hinter Drehfeldmaschinen erfolgen. Dazu gehört die Darstellung und die Definition von Raumzeigern, die Vorstellung der verschiedenen verwendeten Koordinatensysteme, eine kurze Einführung in die Theorie zur allgemeinen Drehfeldmaschine sowie letztendlich die Anwendung der aufgeführten Grundlagen auf die permanenterregte Synchronmaschine.

Daraus resultierend leitet sich das Grundwellenmodell der PMSM ab das am Ende dieses Kapitels als Ergebnis und Grundlage für die Modellierung der verwendeten Maschine dienen wird.

2.1 Raumzeiger

Raumzeiger sind ein mathematisches Hilfsmittel zur Beschreibung von Drehstromgrößen bzw. räumlich sinusförmig verteilten 3-Größen-Systemen und eignet sich unter bestimmten Voraussetzungen zur Modellierung von Vorgängen in Drehfeldmaschinen und Umrichtern.

Ein Raumzeiger in einer Drehfeldmaschine ist so definiert, dass er durch seine Lage in der komplexen Zahlenebene die Lage einer sinusförmigen Größe am Ankerumfang der Welle der Maschine beschreibt und seine Größe der Amplitude proportional ist (Durch Einführung eines Skalierungsfaktors wird später die Amplitude gleich der Länge des Raumzeigers sein).

Hier muss jedoch bei mehrpoligen Machinen eine Unterscheidung zwischen dem mechanischen und dem elektrischen Winkel gemacht werden. Der mechanische Winkel beschreibt den psysikalischen Winkel an der Welle während für den elektrischen Winkel ein Polpaar 360° entspricht.

Im Folgenden wird von einer Maschine mit nur einem Polpaar ausgegangen, die Erweiterung auf mehrere Polpaare wird danach kurz aufgezeigt.

Als Beispiel wird der Strombelag einer Drehfeldmaschine als Raumzeiger definiert:

Besitzt eine Drehfeldmaschine eine symmetrische dreisträngiige Wicklung sollen folgende Aussagen gelten:

- Es ist kein Nullstrom vorhanden. Mathematisch beschrieben durch:

$$I\mathrm{a}(t) + I\mathrm{b}(t) + I\mathrm{c}(t) = 0 \tag{2.1}$$

- Eine stromdurchflossene Wicklung erzeugt ein entlang dem Umfang sinusförmig verteiltes Magnetfeld B im Luftspalt. (Radialkomponente, Grundwelle)

- Die Überlagerung der drei Stränge resultiert in einem wiederum sinusförmigen Gesamtfeld Bges

Die Amplitude und die Phasenlage dieser Welle stellt man nun als komplexen Raumzeiger $\underline{I}$ dar.

Für jede Wicklungsachse gilt dabei:

$$I_a(t) = \hat{I} * \cos(\omega t) \tag{2.2}$$

$$I_b(t) = \hat{I} * \cos(\omega t - 120°) \tag{2.3}$$

$$I_c(t) = \hat{I} * \cos(\omega t - 240°) \tag{2.4}$$

$$I_a(t) + I_b(t) + I_c(t) = 0 \tag{2.5}$$

Die Summe der zeitlichen Größen in den Wicklungen ist gleich Null.

Der Raumzeiger $\underline{I}$ wird nun wie folgt definiert:

$$\underline{I} = \frac{2}{3} * (I_a + \underline{a} * I_b + \underline{a}^2 * I_c) \tag{2.6}$$

Dabei wird der komplexe Drehoperator $\underline{a}$ verwendet. Dieser ist durch $\underline{a} = e^{j120°}$ bzw. $\underline{a} = e^{j240°}$definiert.

Der Feldraumzeiger setzt sich also aus der Addition der drei Feldkomponenten in der Wicklungsachse ($\varphi_a = 0$) zusammen und somit wird die räumliche Verteilung berücksichtigt.

Nach Berechnung von $\underline{I}$ ergibt sich dann:

$$\underline{I}(t) = \hat{I} * e^{j\omega t} \tag{2.7}$$

Der Raumzeiger $\underline{I}$ hat also dieselbe Amplitude wie die Phasengröße.

Analog können nun Phasenspannungen sowie Flüsse definiert werden. Die Raumzeiger sind komplexe Größen und stellen das dreiphasige Wicklungssystem damit in einem zweiphasigen Wicklungssystem dar das zwei senkrecht aufeinander stehende Achsen aufweist. (Da der komplexe Zeiger in Imaginärteil und Realteil zerlegt werden kann)

Würde man auch höhere Harmonische berücksichtigen würde dies genauso passieren, jedoch wäre die Umlaufgeschwindigkeit entsprechend der Ordnungszahl der jeweiligen Harmonischen höher.

Wir haben zu Beginn vorausgesetzt dass es keine Nullkomponente gibt, dies kann jedoch in ähnlicher Weise passieren.

Die Momentanwerte in den Phasen erhält man aus den folgenden Gleichungen:

$$I_a(t) = Re\{\underline{I}\} = I_\propto \tag{2.8}$$

$$I_b(t) = Re\{\underline{I} * \underline{a}^{-1}\} = \frac{1}{2}(\sqrt{3}\, I_\beta - I_\propto) \tag{2.9}$$

$$I_c(t) = Re\{\underline{I} * \underline{a}^{-2}\} = \frac{1}{2}(-\sqrt{3}\, I_\beta - I_\propto) = -\underline{I_a} - \underline{I_b} \tag{2.10}$$

mit $$\underline{I} = I_\alpha + j * I_\beta \tag{2.11}$$

2.2 Koordinatensysteme

Bisher wurde der Raumzeiger in einem raumfesten Koordinatensystem betrachtet. Dieses Koordinatensystem wird im allgemeinen Statorkoordinatensystem genannt die α-Achse fällt meist mit der Achse der Statorwicklung A zusammenfällt.

Um im Weiteren die verschiedenen Koordinatensysteme unterscheiden zu können wird per hochgestellten Index am Beispiel des Stromraumzeigers das Statorkoordinatensystem wie folgt gekennzeichnet: $\underline{I}^S$. In derselben Art wird mit der Notation $\underline{I}^R$ der Stromraumzeiger im Rotorkoordinatensystem gekennzeichnet. Bei diesem Koordinatensystem fällt die reelle Achse mit der a-Achse der Rotorwicklungen zusammen. Das Rotorkoordinatensystem dreht sich dabei relativ zum Statorkoordinatensystem mit der Winkelgeschwindigkeit ω_R.

Ein weiteres Koordinatensystem wird zuerst einmal willkürlich eingeführt und kann ebenfalls genauso definiert werden mit z.B.: $\underline{I}^W$ (z.B. mit den Achsen k-l)

Dieses Koordinatensystem kann sich zum Beispiel am Luftspaltfluss oder auch am Rotorfluss orientieren.

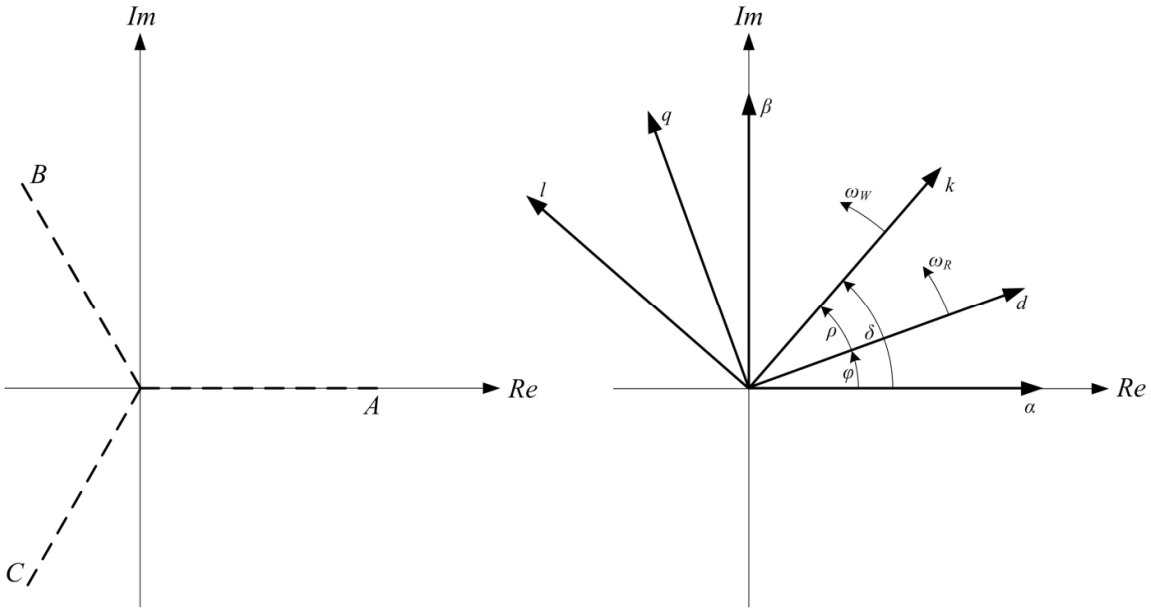


Abbildung 1: Koordinatensysteme

In Abbildung 1 werden die Zusammenhänge zwischen verschiedenen Koordinatensystemen dargestellt.

2.3 Allgemeine Drehfeldmaschine

Für die allgemeine Drehfeldmaschine lassen sich unter bestimmten Voraussetzungen bzw. Vereinfachungen relativ einfache Gleichungen zur Beschreibung der Vorgänge anschreiben.

Diese Voraussetzungen bzw. Vereinfachungen sind:

- Sättigungseffekte werden vernachlässigt, die Magnetisierungskennlinie ist linear
- Konzentrierte Wicklungen
- Alle Größen haben einen sinusförmigen Verlauf entlang dem Umfang der Maschine
- Rotorgrößen werden auf den Stator umgerechnet.

Im Allgemeinen kann man als Ausgangspunkt die Spannungsdifferentialgleichungen des Stator- und Rotorkreises anschreiben:

$$\underline{U_1}^S = R_1 * \underline{I_1}^S + \frac{d\underline{\psi_1}^S}{dt} \tag{2.12}$$

$$\underline{U_2}^R = R_2 * \underline{I_2}^R + \frac{d\underline{\psi_2}^R}{dt} \tag{2.13}$$

Und die dazugehörigen Flussverkettungsgleichungen:

$$\underline{\psi_1}^S = L_1 * \underline{I_1}^S + M * e^{i\varphi} * \underline{I_2}^R \tag{2.14}$$

$$\underline{\psi_2}^R = M * e^{-i\varphi} * \underline{I_1}^S + L_2 * \underline{I_2}^R \tag{2.15}$$

Mit den Variablen:

L1... Eigeninduktivität der Statorwicklung

L2… Eigeninduktivität der Rotor wicklung

M… lageabhängige Gegeninduktivität zwischen Stator und Rotor

φ… Wirkel zwischen stator- und rotorfestem Koordinatensystem

$M * e^{-i\varphi}$ … zeitvariante Gegeninduktivität

Das Luftspaltmoment lässt sich mit:

$$Mi = -\frac{3}{2} * p * Im\left\{\underline{\psi}_1{}^S * \underline{I}_1{}^{*S}\right\} = +\frac{3}{2} * p * Im\left\{\underline{\psi}_1{}^{*S} * \underline{I}_1{}^S\right\} \tag{2.16}$$

2.4 Permanenterregte Synchronmaschine

Bei der permanenterregten Synchronmaschine werden um das Erregerfeld aufzubauen anstatt einer stromdurchflossenen Erregerspule Permanentmagnete verwendet. Damit ist die Statorwicklung die einzige stromdurchflossene Wicklung in dieser Maschine.

Man unterscheidet prinzipiell die permanenterregte Synchronmaschine PMSM oder BLAC-Motor von der bürstenlosen Gleichstrommaschine BLDC-Motor.

Bei der PMSM werden die Statorwicklungen mit sinusförmigen Strömen beaufschlagt und bei dem BLDC-Motor handelt es sich um trapezförmige Stromverläufe.

Es gibt symmetrisch aufgebaute PMSM und unsymmetrische. Bei den symmetrischen PMSM kann prinzipiell die Aussage gemacht werden dass sowohl Widerstand als auch Induktivität des Motors im d-q-Grundwellenmodell in beiden Achsen gleich sind.

Bei den unsymmetrisch aufgebauten Maschinen versucht man durch geometrisch unsymmetrischen Aufbau gezielt verschiedene Induktivitäten im d-q-System zu erreichen um auch die Wirkung des dann auftretenden Reluktanzmoments ausnutzen zu können.

Wir können die allgemein bekannten Gleichungen zur Beschreibung des dynamischen Verhaltens der Synchronmaschine im dq-Koordinatensystem (rotorfest) wie folgt angeben:

$$\frac{d\psi_d}{dt} = Ud - R1 * Id + \omega_R * \psi_q \tag{2.17}$$

$$\frac{d\psi_q}{dt} = Uq - R1 * Iq - \omega_R * \psi_d \tag{2.18}$$

$$\psi_d = \psi_{PM} + L_d * I_d \tag{2.19}$$

$$\psi_q = L_q * I_q \tag{2.20}$$

Und das Drehmoment mit:

$$Mi = \frac{3}{2} * p * \left(\psi_{PM} * I_q + \left(L_d - L_q\right) * I_d * I_q\right) \tag{2.21}$$

$$J * \dot{\omega}_R = Mi - M_L \tag{2.22}$$

2.5 Grundwellenmodell einer PSM

Beschränkt man sich auf den Grundstellbereich und geht man davon aus, dass die Maschine einen isolierten Sternpunkt hat kann mit den oben angeführten Gleichungen schon das Grundwellenmodell der PMSM aufgestellt werden.

In dieser Arbeit wird das Grundwellenmodell in MATLAB/Simulink als Koppelplan erstellt und zur weiteren Entwicklung der Parameteridentifikationsmethoden in der Simulation genutzt. Da der Verfasser von der Übersichtlichkeit der Koppelpläne überzeugt ist wird hier anstatt eines Signalflussplans gleich der Koppelplan aus Simulink gezeigt.

Wichtig für die Modellerstellung in MATLAB sind noch die Koordinatentransformationen die zur Umwandlung zwische 3-phasigem-, Ständer- und Rotorkoordinatensystem gebraucht werden. Diese wurden ebenfalls in Simulink erstellt und werden hier kurz gezeigt.

- Transformation Ständerkoordinatensystem auf Rotorkoordinatensystem

$$I_d = I_\alpha * \cos\varphi + I_\beta * \sin\varphi \tag{2.23}$$

$$I_q = -I_\alpha * \sin\varphi + I_\beta * \cos\varphi \tag{2.24}$$

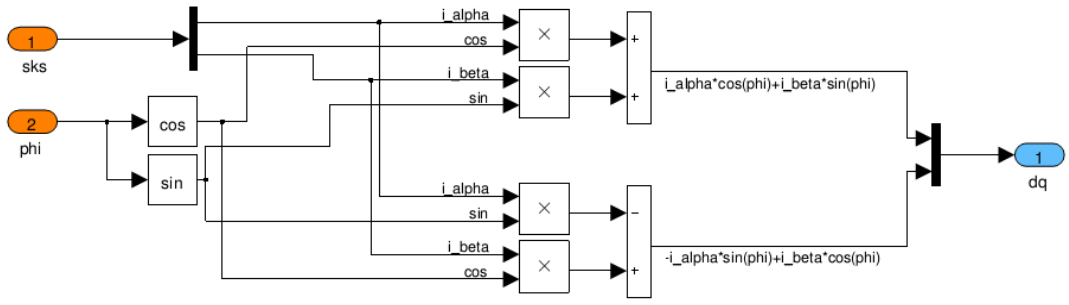


Bild 1: Koordinatentransformation sks -> dq in Simulink

- Transformation Rotorkoordinatensystem auf Ständerkoordinatensystem

$$I_\alpha = I_d * \cos\varphi - I_q * \sin\varphi \tag{2.25}$$

$$I_\beta = I_d * \sin\varphi + I_q * \cos\varphi \tag{2.26}$$

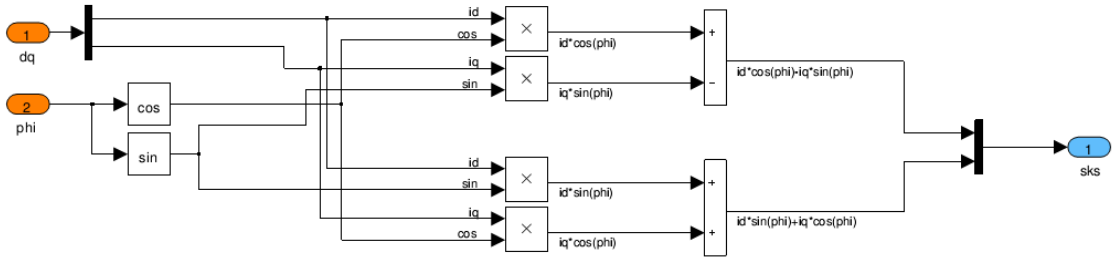


Bild 2: Koordinatentransformation dq -> sks in Simulink

- Transformation vom Ständerkoordinatensystem in das 3-phasige System:

  (Inverse Clarke Transformation)

$$T_{\propto\beta}{}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \tag{2.27}$$

- Transformation vom 3-phasigen ins Ständerkoordinatensystem

$$T_{\alpha\beta} = \frac{2}{3} * \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.28}$$

Der Signalflussplan/Koppelplan für den elektrischen Teil der PMSM in Simulink:

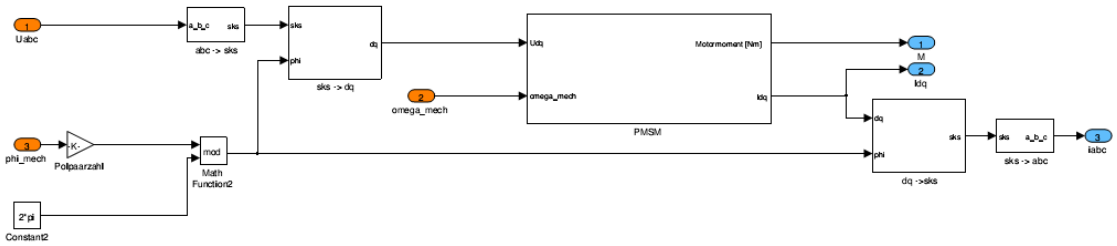


Bild 3: Transformationen rund um das Grundwellenmodell in Simulink

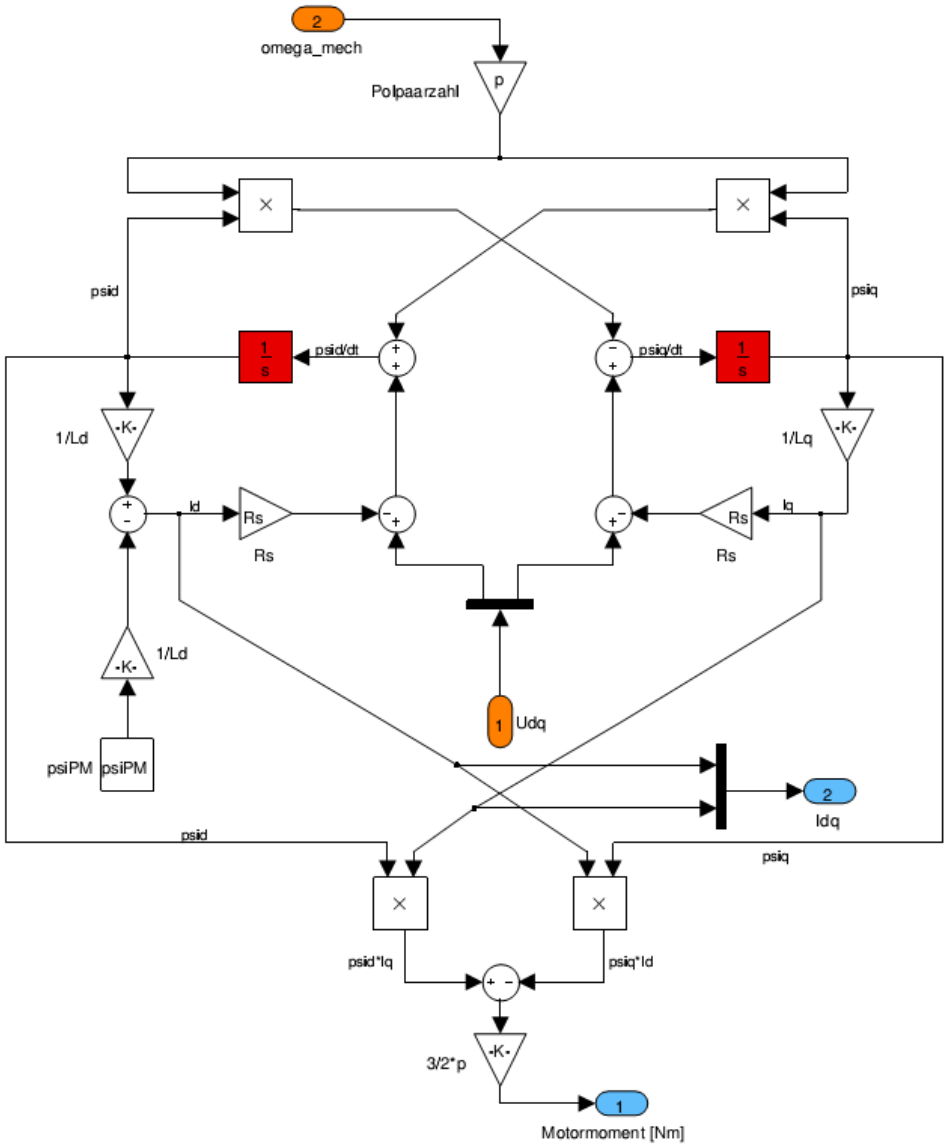


Bild 4: Koppelplan des Grundwellenmodells in Simulink

3 Parameterbestimmung der PMSM

3.1 Einführung

Um die Regelung der PMSM richig auslegen zu können muss Wissen über den Regelkreis, also die Parameter der permanenterregten Synchronmaschine existieren. Lässt man folgende Vereinfachungen zu:

- Keine Temperaturabhängigkeit der Parameter
- Keine Nichtlinearität in der Leistungselektronik
- Keine Sättigungserscheinungen im Eisen

können die Maschinengleichungen für eine PMSM im d-q-Koordinatensystem relativ einfach mit konstanten Koeffizienten angeschrieben werden: (siehe [1])

$$\frac{d\psi_d}{dt} = U_d - R_s * I_d + \omega_e * \psi_q \tag{3.1}$$

$$\frac{d\psi_q}{dt} = U_q - R_s * I_q - \omega_e * \psi_d \tag{3.2}$$

$$\psi_d = \psi_{PM} + L_d * I_d \tag{3.3}$$

$$\psi_q = L_q * I_q \tag{3.4}$$

$$M_i = \frac{3}{2} * p * (\psi_{PM} * I_q + (L_d - L_q) * I_d * I_q) \tag{3.5}$$

$$J * \frac{d\omega_m}{dt} = M_i - M_L \tag{3.6}$$

Die dabei auftretenden unbekannten Koeffizienten sind:

- Der Ständerwiderstand R_s
- Die Induktivitäten L_d und L_q
- Der Permanentmagnetfluss ψ_{PM}
- Das Trägheitsmoment J

Diese Parameter sind im d-q-Koordinatensystem beschrieben, das heißt die relative Stellung des Rotors zum Stator muss bekannt sein um die Koordinaten im d-q-Koordinatensystem angeben bzw. anwenden zu können.

Das Problem, sowohl bei der Anwendung von Absolutpositionsgebern als auch bei Relativpositionsgebern, besteht dabei darin, dass die genaue Montagelage bekannt sein muss um diesen Winkel angeben zu können. Ist die Montagelage eines Resolvers bzw. eines Absolutpositionsgebers einmal vermessen so ist es möglich den Winkel anzugeben, er muss jedoch nach einer möglichen Wartung oder Demontage wieder vermessen werden da selbst kleine Messfehler hier Auswirkungen in der Drehmomentbildung und letztlich in der Effizienz des Motors nach sich ziehen.

Bei einem Inkrementalgeber bzw. Relativpositionsgeber zeigt sich eine noch schwierigere Ausgangslage: Da die Absolutposition erst bekannt sein kann wenn ein möglicherweise vorhandener Indexpuls gemessen wird muss hier der Motor gesteuert zur Rotation gebracht werden bis der indexpuls auftritt und selbst dann treten zusätzlich die selben Probleme wie beim Absolutpositionsgeber nach Demontage auf.

Da in dieser Arbeit im weiteren Verlauf ein Inkementalencoder verwendet wird soll der mechanische Versatz zwischen Nullposition des Statorkoordinatensystems und der Nullposition des Drehgebers im Folgenden als Encoder-Offset bezeichnet werden.

In diesem Kapitel soll eine Testmethode gefunden werden die es erlaubt den Motor bei erster Inbetriebnahme innerhalb kurzer Zeit zu vermessen und diese Parameter festzustellen. Es soll dabei auch dem wenig einschlägig vorgebildeten Benutzer die Möglichkeit zur Konfiguration der Testläufe geboten werden. Als Voraussetzung für die Testläufe sollen Maximalkennwerte der Maschine, sowie Nennstrom und grobe Schätzwerte für das Trägheitsmoment sowie den Permanentmagnetfluss angenommen werden.

Die Testmethode soll so entworfen werden dass es dabei zu einer möglichst geringen bis zu keiner wahrnehmbaren Bewegung des Rotors kommt.

Die entworfenen Methoden werden implementiert und überprüft, die Ergebnisse werden in Kapitel 7 dokumentiert.

3.2 Bestimmung des Encoder-Offset

Wie bereits erwähnt soll der Encoder Offset (ab hier wird von einem Inkrementalgeber ausgegangen) in einer solchen Weise bestimmt werden, dass dabei der Rotor eine möglichst geringe Bewegung durchführt.

Bei traditionellen Methoden wird im Normalfall einmal die Montageposition des Indexpulses gemessen und dann bei jedem Maschinenstart so lange der Rotor über eine Steuerung angedreht bis der Indexpuls das erste mal auftritt. Dies hat zur Folge dass man erst in eine Momentenregelung übergehen kann wenn der Indexpuls auftritt, d.h.: man kann den eigentlich größten Vorteil des Elektromotors: die sofortige Verfügbarkeit des Maximalmoment ab der Drehzahl 0 nicht bzw. nicht von Beginn an ausnützen. Die Ideallösung wäre hier ein Testlauf der den Motor bestromt und dabei keine Rotorbewegung auslöst.

Es existieren verschiedene Ansätz die sich eine ähnliche Funktion zum Ziel gemacht haben, die meisten beruhen dabei auf der Schätzung des Rotorwinkels durch einen Beobachter und sind vergleichsweise rechen- und speicherintensiv. In dieser Arbeit wird von der Existenz eines Inkementalgebers ausgegangen und als Ziel definiert einfache und praktikable Testläufe zu entwickeln die wenig speicher- und rechenintensiv sind. Die entworfenen Testläufe arbeiten daher ohne komplexe mathematische Funktionen und es werden keine Vektor- oder Matritzenmultiplikationen benötigt.

3.2.1 Idee

Da das Ziel der Encoder-Offset-Identifikation ein Prüflauf ohne Rotorbewegung ist besteht die prinzipielle Idee genau diesen Wert durch eine Regelung auf 0 zu halten.

Bei Bestromung der Maschine mit einem konstanten, nicht drehenden, Spannungsvektor wird sich ein Drehmoment ausbilden welches den Rotor nach Überwinden der Reibungen in eine Richtung ausrichten wird bzw. eine Bewegung hervorruft. Dieser Bewegung soll durch Nachregeln des Winkels der ausgegebenen Spannung entgegengewirkt werden und damit eine weitere Bewegung des Rotors so minimal wie möglich gehalten werden.

Um die Auswirkung eines Winkelfehlers durch Rastmoment-Auswirkungen zu verhindern soll der Prüflauf zumindest mit halbem Nennstrom realisiert werden.

Da man zu Beginn der Identifikation keine Information über den Ständerwiderstand der Maschine hat wird folgendes Vorgehen realisiert:

Es wird eine Rampe erzeugt die die Höhe des Spannungsvektors langsam steigert. Fängt der Rotor an eine Bewegung auszuführen wird mit einem PI-Regler versucht den Winkel des Vektors so nachzuregeln dass die Bewegung minimal bleibt.

Erreicht die Rampe den gewünschten Strom und ist der Encoder-Offset gefunden wird die Rampe wieder genauso hinuntergefahren. Würde man die Spannung sprunghaft auf 0 ändern könnten bereits geringe Fehler in der Ausrichtung eine Rotorbewegung auslösen.

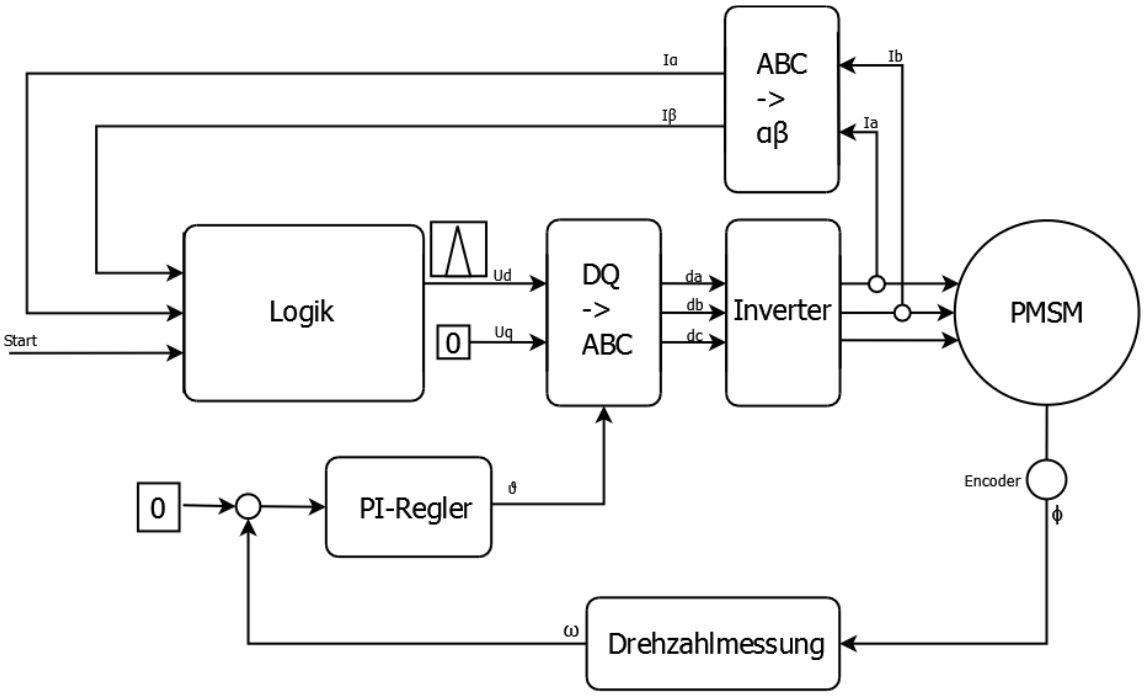


Bild 5:Blockschaltbild zur Encoder-Offset-Bestimmung

Im Bild 5 sieht man das Blockschaltbild zur Encoder-Offset-Bestimmung. Im Block „Logik“ befinden sich verschiedene Zähler und Umschaltbedingungen mit denen man das Ausgangssignal in Länge, Beschaffenheit und Steigung verändern kann. In diesem Block wird die Stromamplitude $I = \sqrt{{I_\alpha}^2 + {I_\beta}^2}$ gebildet und damit die Spannungshöhe begrenzt.

3.3 Bestimmung des Ständerwiderstandes

Der Ständerwiderstand der permanenterregten Synchronmaschine wird durch eine klassische indirekte Widerstandsmessung mit Spannungseinprägung identifiziert. Ist der Encoder-Offset erst einmal bekannt kann er durch Aufschalten einer Gleichspannung in d-Richtung einfach gemessen werden. Dies sollte bei einem nennenswert großen Strom passieren um den Messfehler möglichst gering zu halten. Da bei der Encoder-Offset-Bestimmung bereits eine Rampe bis zum halben Nennstrom der Maschine in ausgerichteter Position ausgegeben wird, wird der Encoder-Offset-Test mit dem Widerstandstest verbunden und zwischen dem An- und dem Abstieg der Rampe des Encoder-Tests noch eine Phase eingeführt in der die Spannung konstant gehalten wird.

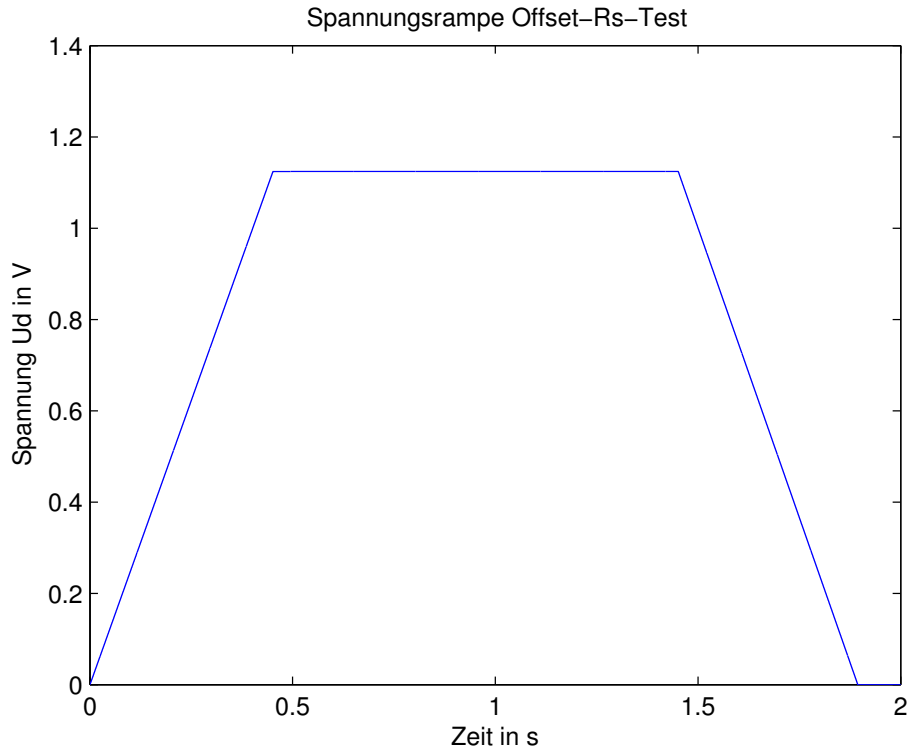


Bild 6: Spannungsrampe zur Offset- und Widerstandsidentifikation

In Bild 6 sieht man die verwendete Ausgangsspannungsrampe. Im ersten Bereich (bis Sekunde t=0.45) steigt die Spannung an und der PI-Winkelregler hält die Nulldrehzahl des Rotors. Ist der halbe Nennstrom der Maschine erreicht wird die Spannung konstant gehalten.

Der PI-Controller wird nun abgeschalten, der Offset-Wert gespeichert und der Spannungsvektor am Ausgang ab nun genau auf dem bestehenden Winkel gehalten. Der Ständerwiderstand wird über den Mittelwert der Messwerte in einem Ausschnitt der Konstanspannungsausgabe gemessen und gespeichert.

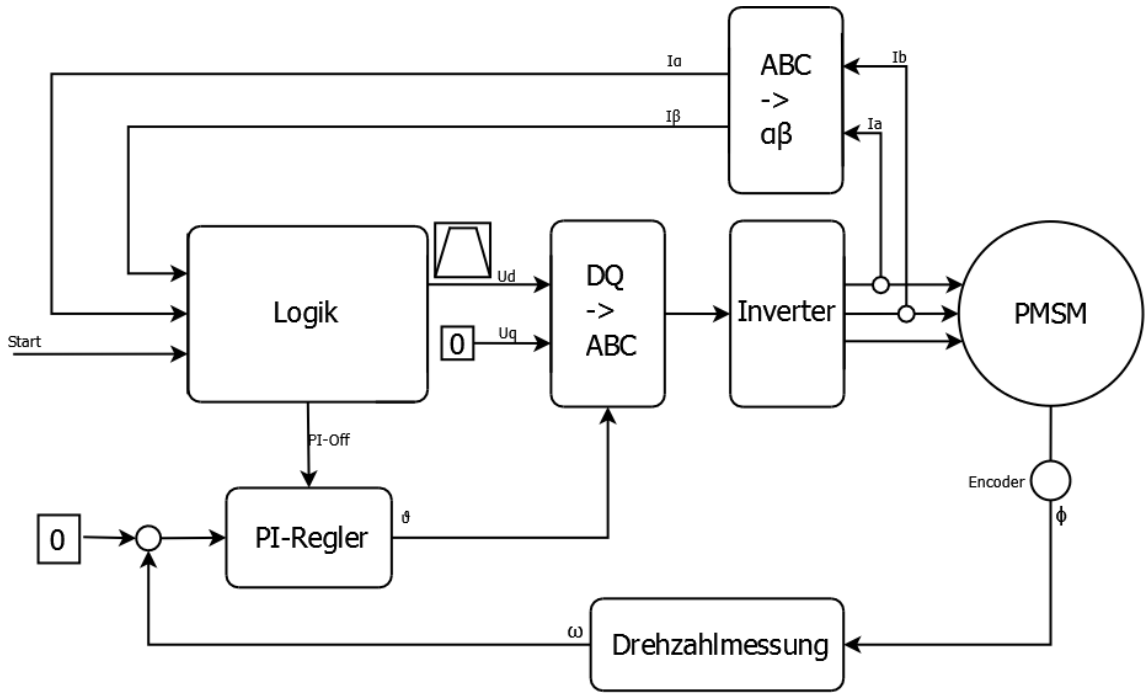


Bild 7: Erweiterter Regelkreis zur Offset- und Widerstandsidentifikation

Nach dieser kombinierten Messung stehen also die Nulllage des Winkelgebers und der Widerstand des Ständers fest. Diese Informationen werden in weiterer Folge zur Messung der Induktivität genutzt.

3.4 Bestimmung der Induktivität

Zur Bestimmung der Induktivität der Maschine gibt es wiederum viele verschiedene Ansätze. So kann man zum Beispiel durch die Erregung des Stators mit Wechselspannung und der Messung der Ströme mit der komplexen Wechselstromrechnung einen komplexen Widerstand berechnen. Durch das zuvor gewonnene Wissen über den Ständerwiderstand kann dann eine Induktivität berechnet werden. In dieser Arbeit wird aber ein anderer Ansatz zur Identifikation gewählt: Man legt an den Stator einen Spannungssprung in d-Richtung und misst die Stromantwort.

Es wird ein zeitdiskreter RL-Filter entworfen und der gemessene Spannungssprung an dessen Eingang gelegt, die Antwort dieses Systems wird genutzt um eine Fehlerfolge zu bilden.

Die Höhe des Spannungssprungs wird aus der Maximalspannung des Widerstandsversuchs U_{Rmax} abgeleitet. Um einen gut verwertbaren Stromanstieg zur Auswertung zu bekommen beträgt die Höhe des Spannungssprunges für die Induktivitätsmessung $U_{Rmax} * 1.5$.

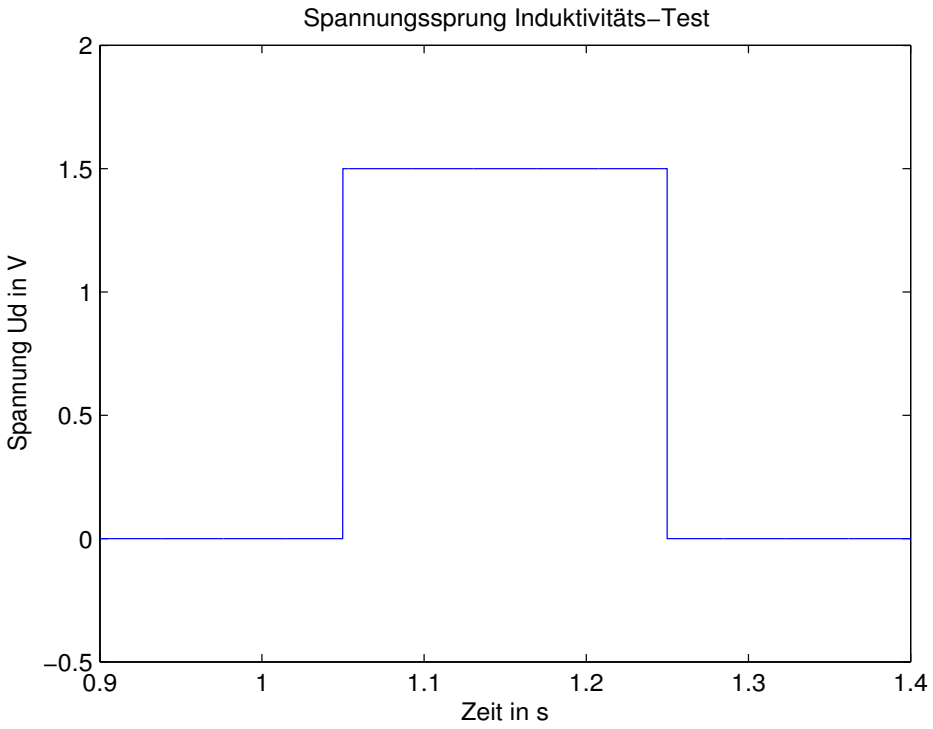


Bild 8: Spannungssprung zur Induktivitätsmessung

Wurde der Spannungssprung bzw. die Sprungantwort ein mal vermessen wird eine Optimierungsregelschleife gestartet die vom Anfangswert $L = 0$ ausgehend versucht den aufgenommenen Verlauf der Stromantwort des Systems anzunähern.

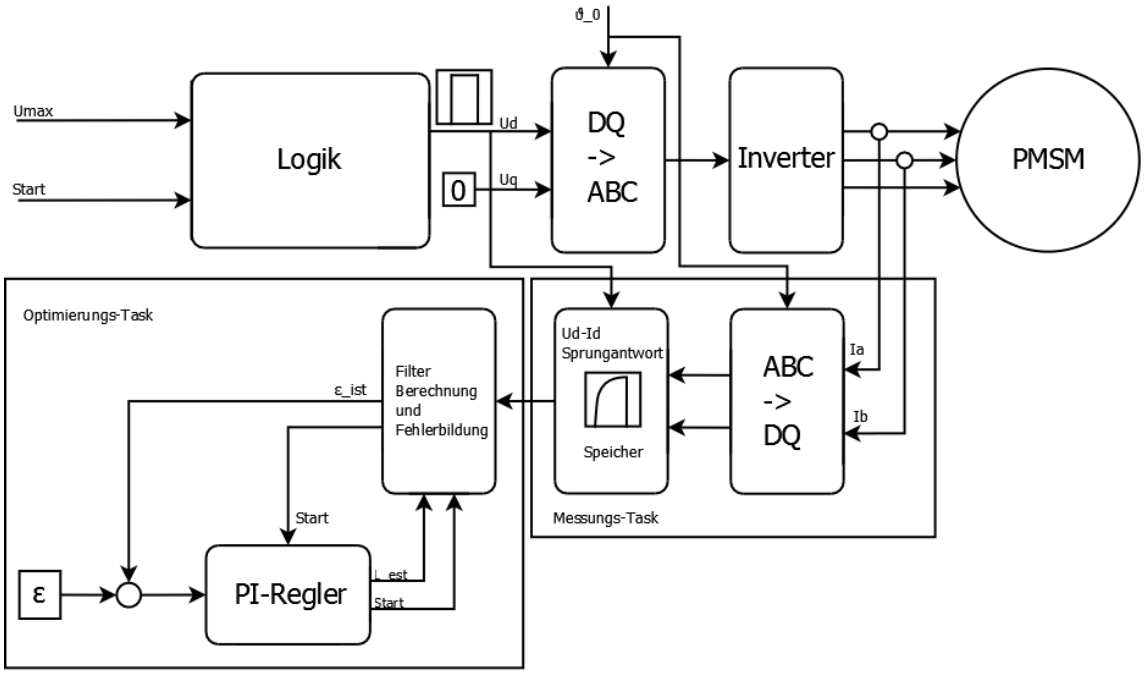


Bild 9: Blockschaltbild Induktivitätsidentifikation

Durch das Wissen über die Beschaffenheit der Strecke (RL-Strecke) kann davon ausgegangen werden dass die Fehlerminimierung hier genauere Ergebnisse liefert.

Diese Regelstruktur wird in Bild 9: Blockschaltbild Induktivitätsidentifikation dargestellt. Dazu anzumerken ist die Unterscheidung zwischen dem Messungs-Task und dem Optimierungs-Task.

Im Messungs-Task werden die sinnvollen Samples (in denen die Induktivität messbar ist) identifiziert und in einen Vektor geschrieben. Wurde dies erfolgreich durchgeführt wird der Task ab dann nicht mehr ausgeführt und erst dann wird der Optimierungs-Task gestartet.

In diesem wird der Filter im Zeitbereich für jeden Optimierungsschritt mit den neuen Werten neu berechnet, und zwar nur für die zeitliche Länge des übergebenen Vektors.

Durch dieses Vorgehen soll die Prozessorlast möglichst niedrig gehalten werden und es werden keine Berechnungen durchgeführt die nicht in das Ergebnis eingehen.

3.4.1 Verwendeter Filter

Der verwendete Filter wird nach der Trapezmethode berechnet.

Ausgehend von der zeitkontinuierlichen Gleichung:

$$U = R * i + L * \frac{di}{dt} \tag{3.7}$$

wird nach der zeitlichen Ableitung des Stroms gelöst:

$$\frac{di}{dt} = \frac{1}{\tau} * (\frac{1}{R} * U - i) \tag{3.8}$$

mit $\tau = \frac{L}{R}$

Wendet man darauf die Trapezmethode an bekommt man die zeitdiskrete Gleichung:

$$i_{k+1} - i_k = \frac{Ta}{2} * \frac{1}{\tau} * \left(\frac{1}{R} * u_k - i_k + \frac{1}{R} * u_{k+1} - i_{k+1}\right) \tag{3.9}$$

$$i_{k+1} = \frac{Ta}{2} * \frac{1}{\tau} * \frac{1}{R} * u_k + \frac{Ta}{2} * \frac{1}{\tau} * \frac{1}{R} * u_{k+1} - \frac{Ta}{2} * \frac{1}{\tau} * i_k - \frac{Ta}{2} * \frac{1}{\tau} * i_{k+1} + i_k \tag{3.10}$$

$$\mathrm{i_{k+1}} * \left(1 + \frac{Ta}{2} * \frac{1}{\tau}\right) = \frac{Ta}{2} * \frac{1}{\tau} * \frac{1}{R} * (u_k + u_{k+1}) + i_k * (1 - \frac{Ta}{2} * \frac{1}{\tau}) \tag{3.11}$$

Und damit:

$$i_{k+1} = \frac{\frac{Ta}{2} * \frac{1}{\tau} * \frac{1}{R}}{1 + \frac{Ta}{2} * \frac{1}{\tau}} * (u_k + u_{k+1}) + \frac{1 - \frac{Ta}{2} * \frac{1}{\tau}}{1 + \frac{Ta}{2} * \frac{1}{\tau}} * i_k \tag{3.12}$$

Mit

$$A = \frac{\frac{Ta}{2} * \frac{1}{\tau} * \frac{1}{R}}{1 + \frac{Ta}{2} * \frac{1}{\tau}} \tag{3.13}$$

und

$$B = \frac{1 - \frac{Ta}{2} * \frac{1}{\tau}}{1 + \frac{Ta}{2} * \frac{1}{\tau}} \tag{3.14}$$

Kann der Filter in der Form

$$i_k = A * (u_{k-1} + u_k) + B * i_{k-1} \tag{3.15}$$

angeschrieben werden.

Hier muss bei der Implementierung besonders bei der Berechnung der Koeffizienten in Fixkommadarstellung auf die vorhandene Auflösung Acht gegeben werden.

4 Software

In diesem Kapitel soll kompakt auf die spezifischen Eigenschaften der verwendeten Funktionsblöcke eingegangen werden. Danach wird gezeigt wie die verschiedenen Ideen aus den vorhergegangenen Kapiteln realisiert wurden.

In der Arbeit wurden zeitkontinuierliche sowie zeit- und wertdiskretisierte hardwarenahe Simulationsmodelle erstellt, diese werden ebenfalls kurz vorgestellt und verglichen.

Die entwickelten Funktionen wurden in einer Simulink Bibliothek hinterlegt und sind dort für den leichten Zugriff für den Anwender verfügbar.

Es wird durchgehend versucht die Vorteile des Fixkommaformats zu nutzen, d.h. es werden Eingangs- und Ausgangsgrößen immer dann normiert wenn dies sinnvoll ist.

4.1 Drehzahlbestimmung

Die Auswertung des Encoder-Signals bzw. die Drehzahlbestimmung nimmt eine bestimmende Rolle in der Regelstruktur ein. Da die Auflösung und Genauigkeit stark von der Abtastzeit des Encodersignals und der Ausführungszeit der Drehzahlbestimmung abhängt gilt es hier tiefergehende Überlegungen anzustellen und auch einen Blick auf die vorhandene Hardware zu werfen. Dazu werden das Encoderdatenblatt sowie die Auswerteeinheit des DSP herangezogen.

Auf der verwendeten PMSM wird ein Inkementalencoder mit Indexpuls genutzt. Zur Auswertung wird das Signal das nach der Vierfachauswertung durch die DSP zur Verfügung steht (Position counter).

Im folgenden Abschnitt wird erläutert wie man den TI F2808 richtig konfiguriert um dieses Signal ,für die im Folgenden diskutierte Auswertung, möglichst sinnvollen Art vorliegen hat. Prinzipiell kann man jedoch davon ausgehen, dass es möglich ist einen Zählerstand zu erhalten der sich zwischen einem Minimalwert und einem Maximalwert bewegen wird. Durch eine geeignete Verarbeitung ist es möglich daraus die Drehrichtung und die Drehzahl zu bilden.

Ebenfalls kann bei einem Wissen über die Maximal und Minimalwerte des Zählers auch der Überlauf behandelt bzw. berechnet werden. Das Ziel in den folgenden Abschnitten ist ein Signal zu erzeugen dass möglichst genau bzw. in ausreichender Genauigkeit die Drehzahl wiedergibt.

4.1.1 Test und Konfiguration des eQEP Moduls

Das eQEP-Modul (Enhanced Quadrature Encoder Pulse) ist ein konfigurierbares Hardware-Modul der C2000-DSP-Serie von TI das zur Vierfachauswertung der Encodersignale genutzt werden kann. Das Modul kann im Simulink Library Browser unter: Embedded Coder -> Embedded Targets -> Processors -> Texas Instruments C2000 -> F280x -> eQEP gefunden werden. Bevor man das Modul richtig konfigurieren kann müssen einige Randbedingungen diskutiert werden. So spielt die Maximal- und Minimaldrehzahl eine große Rolle in der Genauigkeitsbestimmung der Messung und beeinflusst auch die Wahl der Abtastzeit des Moduls.

Prinzipiell können bei Verwendung eines Encoders zwei Arten der Auswertung zur Drehzahlmessung unterschieden werden. Diese zwei Arten unterscheiden sich im Bezug auf ihr Anwendungsgebiet:

Das erste Prinzip beruht auf der Messung der Zeit die ein Encoderstrich zum passieren der Sensoren (Photodioden) braucht, die zweite Messung beruht auf der Messung der Anzahl von Encoderstrichen in einem bestimmten Zeitintervall.Dabei wird die Messung der Differenzzeit pro Encoderstrich im Low-Speed Bereich eingesetzt.Im Gegenzug dazu wird die Berechnung der Geschwindigkeit durch die Encoderstrichdifferenz pro Zeitintervall für die Messung bei höheren Drehzahlen verwendet. Die Gründe dafür liegen in der Genauigkeit der jeweiligen Auswertungsart, dies wird im Folgenden kurz aufgezeigt.

Die richtige Konfiguration des eQEP-Moduls ermöglicht es beide Auswertungsarten zu kombinieren und somit für jegliche Maximaldrehzahl und Abtastzeit die höchstmögliche Auflösung zu erreichen.

Zuerst wenden wir uns der Messung bei geringer Geschwindigkeit zu:

- Low speed measurement

Die Möglichkeit, die Zeit mit möglichst hoher Auflösung zu messen und trotzdem die Drehzahlauswertung mit der gewünscht niedrigen Abtastfrequenz zu betreiben bietet das eQEP-Modul durch die Quadrature edge-capture unit.

Dabei muss bei der Konfiguration im Reiter „Speed calculation“ zuerst die Verwendung der capture unit bestätigt werden (oberstes Häckchen in Abbildung 2).Bevor die weiteren Einstellungen erklärt werden kurz zur Funktion der edge-capture unit:

Die schnelle Zeitmessung wird hier durch die Messung der Frequenz des verwendeten Quarzes ermöglicht. Diese beträgt bei der verwendeten Hardware 100Mhz. Auf der Hardware wird dieser Takt durch einen Vorteiler, den sogenannten Prescaler heruntergeteilt, um in dem Bereich indem es sinnvoll ist den vorhandenen Datentyp möglichst gut ausnutzen zu können d.h keine sinnlosen Überläufe zwischen den Abtastungen zu erzeugen. Dabei sind binärwertige Teiler mit den Werten 1-128 zur Verfügung, also 1,2,4..etc (eQEP capture Timer prescaler). Des Weiteren kann ausgewählt werden, ob auf jeden Encoderpuls reagiert werden soll oder auf binärwertige Vielfache (1-2048 verfügbar, Unit position event prescaler). Damit kann die Genauigkeit bei dieser Auswertung über einen weiten Bereich variiert werden.

Der dadurch erzeugte Takt startet beim Erkennen eines vorbeifahrenden Encoderpulses (bereits durch Vierfachauswertung) und hat das Datenformat uint16.

Das bedeutet der Zähler läuft bei 2^16 bzw. 65535 über und startet, unter der Voraussetzung dass kein weiterer Encoderstrich gezählt wurde, neu. Wurde ein zweiter Encoderstrich gezählt, wird die Zeit, die zwischen den Flanken der beiden Pulse vergangen ist, in ein Register geschrieben das man beim nächsten Abtasten der eQEP-Einheit auslesen kann. Danach wird der Zähler neu gestartet. Das Speichern des Zählerstandes bei Auftreten eines Encoderpulses wird mit der Einstellung: Capture timer and position: On position counter read eingestellt. (Abbildung 2). Eine weitere Möglichkeit wäre das Auslesen nach einer Einheitszeit (Unit Time).

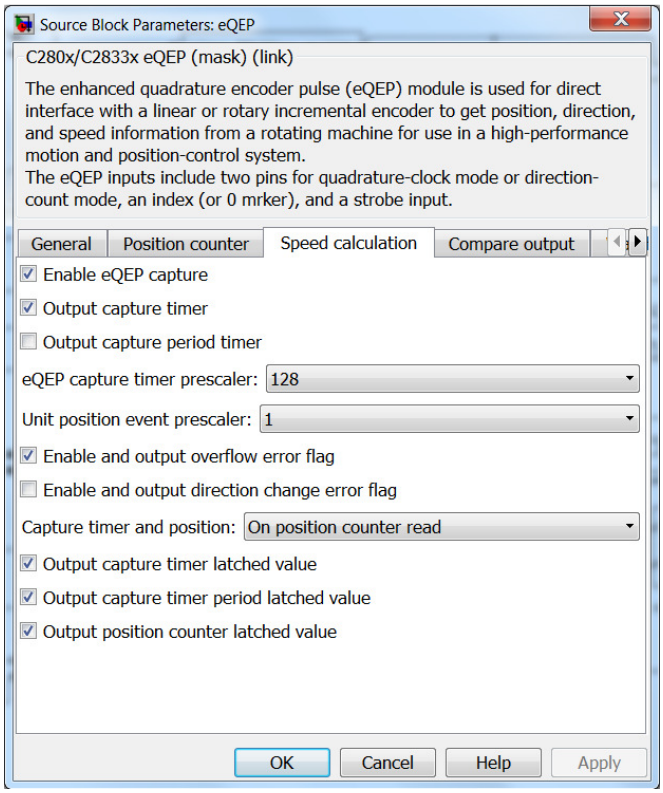


Abbildung 2: Konfiguration der edge-capture unit des eQEP-Moduls

Als Randbedingungen haben wir also den existenten Takt des Quarzes und die beiden Prescaler zu definieren. Im Bereich kleiner Drehzahlen ist auch der Einfluss der Abtastzeit dadurch sehr wichtig: Will man eine Drehzahlauswertung zur Verringerung der Prozessorlast mit geringer Abtastfrequenz betreiben muss darauf geachtet werden, dass der Zähler in der Zwischenzeit nicht überläuft bzw. eine Auswertung des Überlaufs implementiert werden.

Prinzipiell lässt sich zur Auswertung bei niedriger Drehzahl folgendes festhalten:Die Messung der Geschwindigkeit erfolgt über einen festen Weg und eine variable Zeit, die Berechnung erfolgt in einem abgetasteten System.

Dazu kann folgende Gleichung angeschrieben werden:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \tag{4.1}$$

Dabei ist X der Weg/Umfang der von einem bzw. mehreren Encoderstrichen beschrieben wird, ΔT der Zeitunterschied zwischen dem Auftreten der Encoderpulse und v die Geschwindigkeit.

Die Auflösung wird dabei mit steigender Drehzahl schlechter. Durch die endliche Encoderauflösung hat bei höheren Drehzahlen die Zeitauflösung eine größere Auswirkung da der Zählerstand dabei geringer ist. Die Auflösung lässt sich für ein Beispiel (Strichzahl=500) über die Drehzahl angeben:

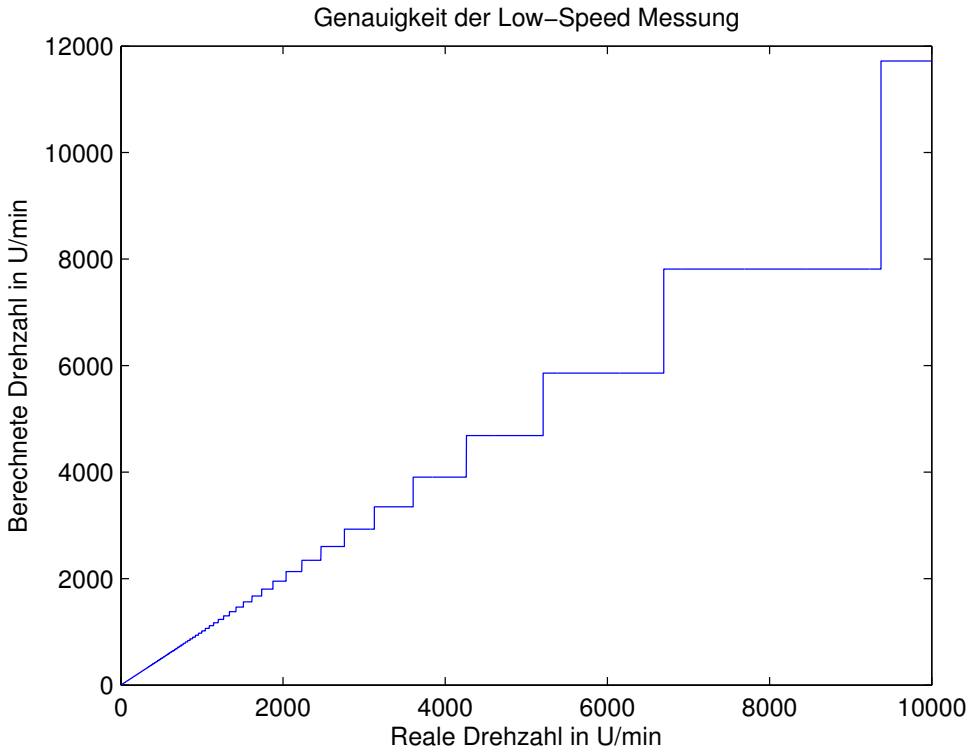


Abbildung 3: Auflösungsverschlechterung bei steigender Drehzahl

In Abbildung 3 wird beispielhaft für den Vorteiler 128 gezeigt wie sich die Auflösung bei steigender Drehzahl verschlechtert. Man erkennt dass der Einsatz dieser Auswertung nur bei niedrigen Geschwindigkeiten sinnvoll ist.

- High speed measurement

Im Gegensatz dazu steht die High-Speed Messung bei der die Abtastzeit fix gewählt wird (Abtastzeit des eQEP-Blocks) und die Anzahl der gemessenen Encoderpulse/striche die Variable Größe darstellt. Die Auflösung wird in diesem Fall von der endlichen Auflösung des Encoders und der Abtastzeit bestimmt.

Die eQEP-Einheit stellt hierbei einen Zähler zur Verfügung der einen frei konfigurierbaren Endwert von 0 bis 4294967295 bzw 32bit hat (Unit time base measurement).

Man kann für viele Anwendungen davon ausgehen, dass dieser Endwert praktisch nie erreicht wird und den Zählerüberlauf somit unbehandelt lassen. Im Rahmen dieser Arbeit wurde jedoch auch im Hinblick auf die Behandlung des Indexpulses entschieden, den Zählerüberlauf abzusichern.

Der Zähler kann bei vorhandenem Wissen über eine gewisse Stellung (z.B. ein mech. Anschlag) auch auf einen Wert initialisiert werden. In dem vorliegenden Fall wird er jedoch bei 0 gestartet.

Eine zusätzliche Funktion bildet das Index-Latch das beim Auftreten des Indexpulses den aktuellen Zählerstand in dieses Register schreibt. Dies lässt sich auch im Konfigurationsfenster der eQEP als Output definieren. Zusätzlich erwähnt sei noch das quadrature direction flag: Diese boolsche Variable kann ebenfalls zur Drehrichtungserkennung genutzt werden.

Für die Auswertung der Drehzahl bei hoher Geschwindigkeit wird also die Wegdifferenz pro Abtastzeit berechnet, dies kann in folgender Form angeschrieben werden:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \tag{4.2}$$

Wie bereits erwähnt hängt hier die Auflösung von der gegebenen Encoderauflösung ρ und der gewählten Abtastrate ab. Sind diese zwei Werte bekannt kann die Auflösung sofort berechnet werden, diese ist über den gesamten Drehzahlbereich konstant.

$$\delta = \frac{1 * \frac{60}{\rho}}{Ts} \text{ in U/min} \tag{4.3}$$

Berechnet man die Auflösung z.B. für eine gegebene Abtastrate von Ts=100Hz und einer Encoderauflösung von ρ=2000 Pulse in Vierfachauswertung so ergibt sich eine Auflösung von δ=3U/min. Die Abhängigkeit von der Abtastzeit Ts wurde zur Veranschaulichung in Abbildung 4 dargestellt.

In der Auswertung muss nun dafür gesorgt werden immer die höchstmögliche Auflösung zu erreichen, das heißt die Drehzahl festzustellen bei der die Auflösung der Low-Speed Messung geringer wird als die der High-Speed Messung.

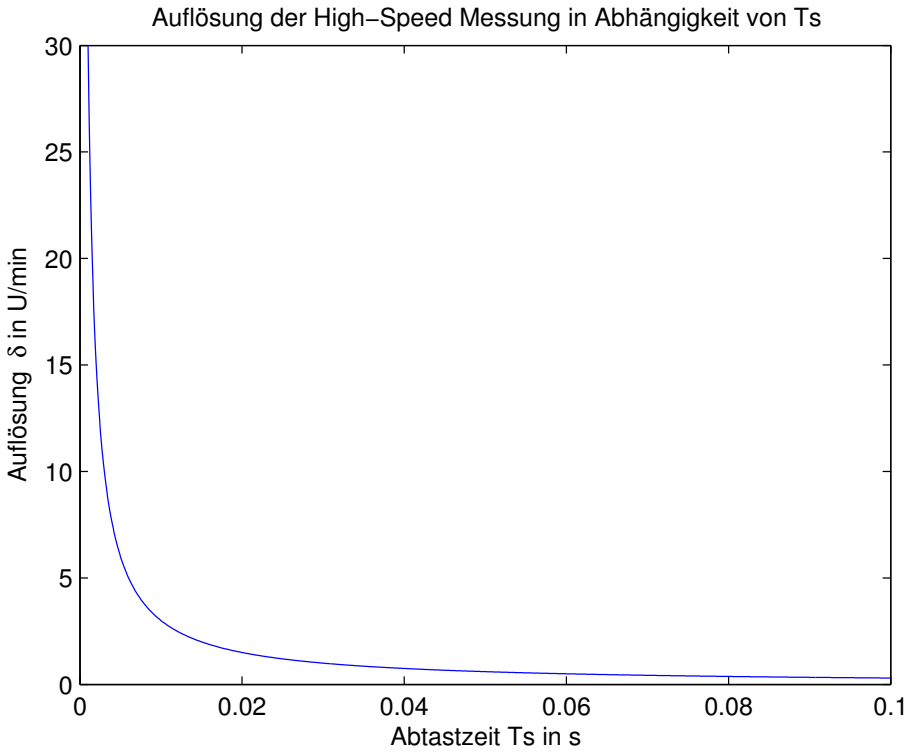


Abbildung 4: Auflösung der High-Speed Messung in Abhängigkeit von Ts

Dieser Wert kann ebenfalls für eine fixe Abtastzeit und gegebenem Vorteiler der Low-Speed Messung berechnet werden.

Die Drehzahlberechnung für die Low-Speed Messung erfolgt nach der Gleichung:

$$v(k) = \frac{X}{\Delta T} = \frac{\frac{60}{\rho}}{Ctr(k) * \frac{PS}{SYSCLK}} \text{ in U/min} \tag{4.4}$$

Dabei ist Ctr(k) der aktuelle dezimale Zählerstand, PS der gewählte Vorteiler und SYSCLK die Frequenz des verwendeten Quarzes in Hz. Die Auflösung berechnet sich durch den Unterschied zwischen den berechneten Drehzahlen für den minimalen Zählerstandunterschied von 1.

Mit$A = \frac{60}{\rho}$, und $T = \frac{PS}{SYSCLK}$ kann angeschrieben werden:

$$\frac{A}{Ctr * T} - \frac{A}{(Ctr + 1) * T} = \delta \tag{4.5}$$

Nach Umformen dieser Gleichung bekommt man die Formel zur Berechnung des gesuchten Zählerstandes in Form der Lösungsformel der quadratischen Gleichung in Normalform mit:

$$Ctr = -\frac{1}{2} \pm \sqrt{\frac{1}{4} + \frac{A}{T * \delta}} \tag{4.6}$$

Im konkret angesprochenen Fall wird z.B. die Drehzahl gesucht bei der die Auflösung δ=3 unterschritten wird, der berechnete Zählerstand ist dann Ctr=87.8898 (Hier ist natürlich nur die positive Lösung sinnvoll). Daraus berechnet sich eine Drehzahl von v=266.66 U/min.

Im Beispielfall muss also bei höheren Drehzahlen als 266U/min bzw. niedrigeren Drehzahlen als -266 U/min auf die Berechnung nach der High-Speed Methode umgeschalten werden.

In der späteren Anwendung wird für den Prescaler der Wert PS=128 verwendet, da es in dem angestrebten Anwendungsfall unbedingt notwendig ist die höchstmögliche Auflösung bei sehr niedrigen Drehzahlen zu erreichen. Da die Abtastzeit später für den Anwender konfigurierbar sein soll, werden die für die Umschaltung benötigten Berechnungen, in das Programm integriert. Genaueres zu den Einstellungen die hier nicht erklärt wurden findet man in [8].

4.1.2 Block Drehzahlbestimmung

Die Funktionen die im vorangegangenen Kapitel besprochen wurden wurden in einem Funktionsblock zusammengefasst und in Simulink mit einer Variableneingabemaske versehen. Der Anwender muss nun nurmehr die gewünschte messbare Maximaldrehzahl, die Anzahl physikalisch vorhandener Encoderstriche und die gewünschte Abtastzeit angeben und die Drehzahlmessung wird so konfiguriert dass die unter diesen Angaben maximal mögliche Auflösung am Ausgang erreicht wird.

Als Ausgänge prinzipiell nötig sind die Geschwindigkeitsinformation und die Positionsinformation, zusätzlich wird noch die Position des Indexpulses und Information über das Auftreten des Indexpulses als Ausgang definiert.

Der Funktionsblock ist in der Software in 2 verschiedenen Ausführungen vorhanden: Eine Version zur Integration auf der Hardware: hier ist der benötigte eQEP-Modul bereits integriert und wird ebenfalls konfiguriert. Die zweite Version wurde für die Simulation entworfen und benötigt am Eingang die Information die sonst das eQEP-Modul bereitstellt. Diese Informationen werden durch einen simulierten Inkrementalgeber-Funktionsblock zur Verfügung gestellt der dem Anwender ebenfalls die gleiche Variablenmaske anbietet.

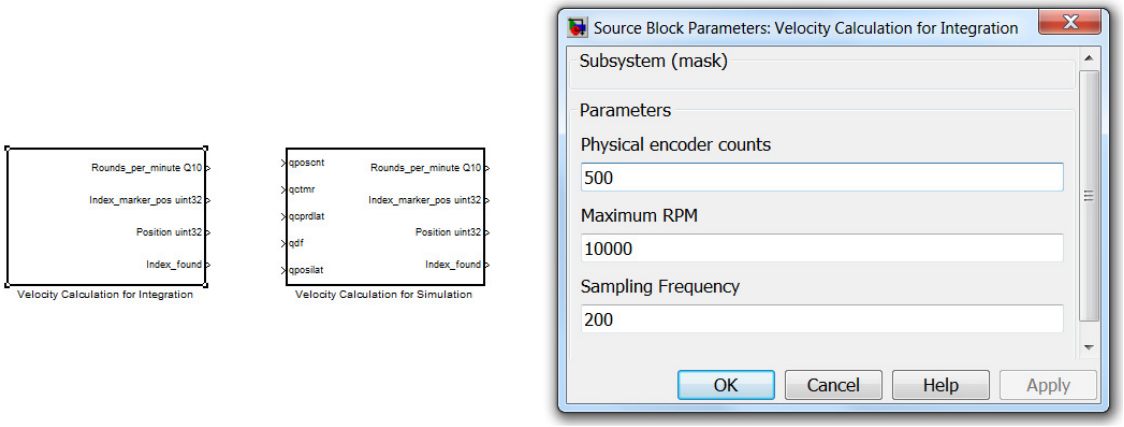


Abbildung 5: Drehzahlberechnung für Integration und Simulation mit Maske

In Abbildung 6 sieht man das Innenleben des Blocks. Wichtig dabei sind die konfigurierbaren Blöcke und die Berechnung der Werte aus den Variablen die aus der Maske gelesen werden.

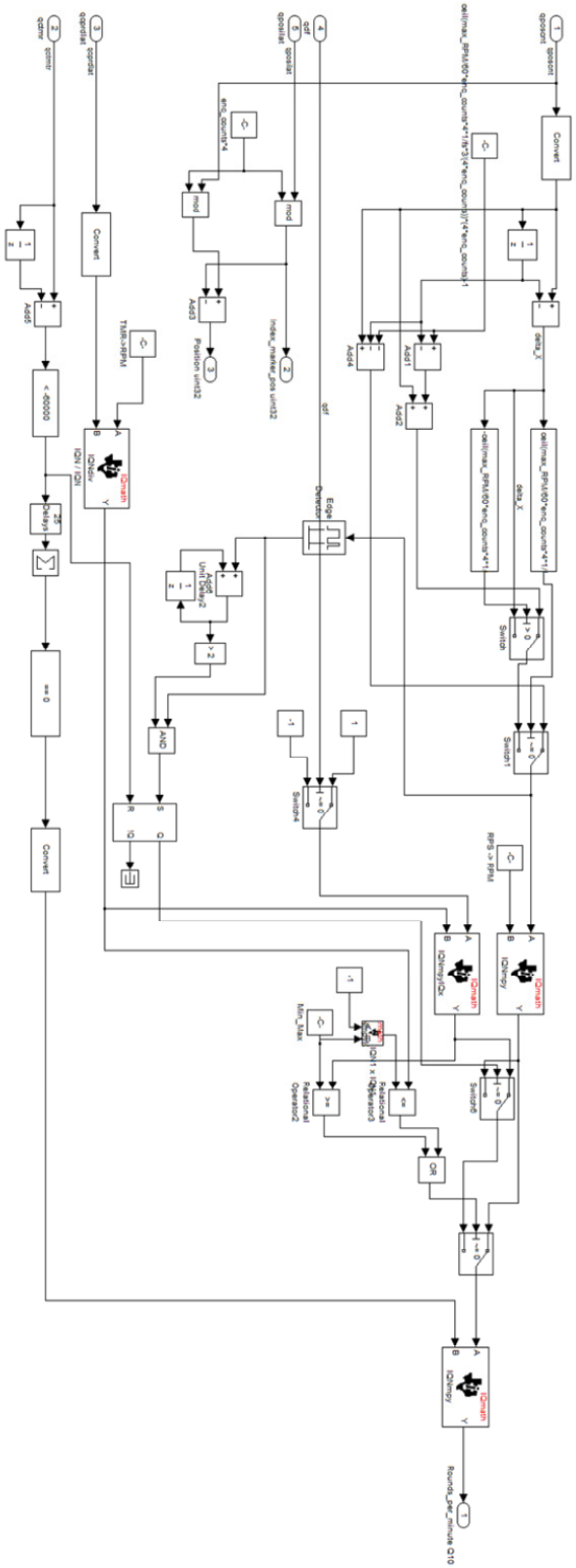


Abbildung 6: Funktionen der Drehzahlberechnung

Berechnung der Werte für die jeweiligen Blöcke:

- eQEP-Modul für High Speed Berechnung:

Die Grundeinstellungen (Prescaler, Ausgänge, etc) werden so konfiguriert wie im Kapitel 4.1.1 beschrieben. Der einzige Wert der noch bestimmt werden muss ist die Höhe des Encoderpuls-Zählers (Maximum position counter value).Dazu wird berechnet wie viele Encoderpulse bei der Maximaldrehzahl in einem Abtastschritt gezählt werden und ein Sicherheitsfaktor $\ominus$ eingeführt mit dem diese Pulszahl multipliziert wird. Damit soll sichergegangen werden dass bei einem möglichen temporären Überschreiten der Maximaldrehzahl durch ein äußeres Moment keine Fehlmessungen entstehen.

Um ausserdem die Positionsmessung und die Indexpulsplatzierung auf der Welle einfach angeben bzw. berechnen zu können wird der Zähler immer auf ein Vielfaches der Maximalen Encoderpulszahl ausgelegt. Dies wird durch Division durch die Anzahl der Encoderpulse, anschließenem Aufrunden und neuerlicher Multiplikation mit der Encoderpulsanzahl erreicht. Da der Zähler beim Index 0 startet muss noch die 1 abgezogen werden.

Die Formel zur Berechnung lautet wie folgt:

$$\max count = \left(\left\lceil \frac{\frac{\max RPM}{60} * \rho * 4}{\frac{Ts}{4 * \rho}} * \theta \right\rceil * 4 * \rho \right) - 1 \tag{4.7}$$

mit der Abtastzeit des Funktionsblocks Ts, dem ganzzahligen Sicherheitsfaktor $\ominus$ und der Encoderstrichzahl ρ. Der Faktor $\ominus$ wird mit 3 gewählt.

Der nun definierte und berechnete maximale Encoderpuls-Zählerstand *max count* wird als Grundlage für die Überlaufbehandlung benötigt.

- Überlaufbehandlung:

Beim Überlauf des Zählers soll eine Korrektur erfolgen und der Wert am Ausgang durch einen berechneten Zählerstandunterschied ersetzt werden.

Bei einem negativen Überlauf springt der Zähler von einem Abtastschritt zum nächsten von einem sehr niedrigen alten Zählerstand (count old) auf einen sehr hohen neuen Zählerstand (new count).

Die Berechnung des Korrekturwertes erfolgt mit:

$$count\ korr\ neg = new\ count - \max\text{count} - old\ count \tag{4.8}$$

Im Falle eines positiven Zählerüberlaufs springt der Zähler von einem hohen alten Zählerstand auf einen sehr niedrigen neuen Zählerstand. Die Korrektur erfolgt mit:

$$count\ korr\ pos = new\ count + \max\text{count} - old\ count \tag{4.9}$$

Der Überlauf wird dadurch erkannt dass der Unterschied zwischen den Zählerständen größer ist als der physikalisch mögliche bei der Maximaldrehzahl und je nach Drehrichtung wird die passende Korrektur an den Ausgang geleitet.

- Low Speed Berechnung

Berechnung der Geschwindigkeit im Low-Speed-Bereich wie im Kapitel 4.1.1 vorgestellt.

Gültigkeitsüberprüfung: Drehzahl wird bei Timerüberlauf auf 0 gesetzt und erst ab dem zweiten gültigen Wert an den Ausgang gelegt. Der erste Puls hat keinen Referenzpuls, der Zähler startet am Anfang direkt beim initialisieren.

Da aus dem Timer-Zählerstand keine Drehrichtungsinformation gewonnen werden kann wird das Quadrature Direction Flag der eQEP-Einheit zur Auswertung herangezogen

- Umschaltung zwischen Low-Speed und High Speed Berechnung.

Die Umschaltung zwischen den Berechnungsarten wird ebenfalls immer mitberechnet und auf den richtigen Wert eingestellt. Zum Zeitpunkt des Umschaltens werden keine besonderen Vorkehrungen getroffen.

4.1.3 Simulation der Inkementalgeberschnittstelle

Um später in der Lage zu sein die Funktionen der Drehzahlbestimmung in der Simulation mit einem Motormodell testen zu können wird der Inkrementalgeber ebenfalls modelliert. Dazu werden die Funktionen nachgebildet die die Eingänge für die Drehzahlberechnung zur Verfügung gestellt. Ausgegangen wird dabei von dem Simulationsmodell der PMSM. In diesem Modell wurde bereits die Möglichkeit eingebracht einen Versatz von Stator- und Rotorkoordinatensystem beim Simulationsstart einzubringen. Nun wird beim Inkrementalgeber-Modell der eigentliche Sensor-Offset, der Montageversatz der Nullposition zur Richtung der Phase a im Rotorkoordinatensystem, eingebracht. Zusätzlich wird der Block mit einer Variablenmaske versehen die wie schon bei der Drehzahlmessung die Funktion automatisch konfiguriert.

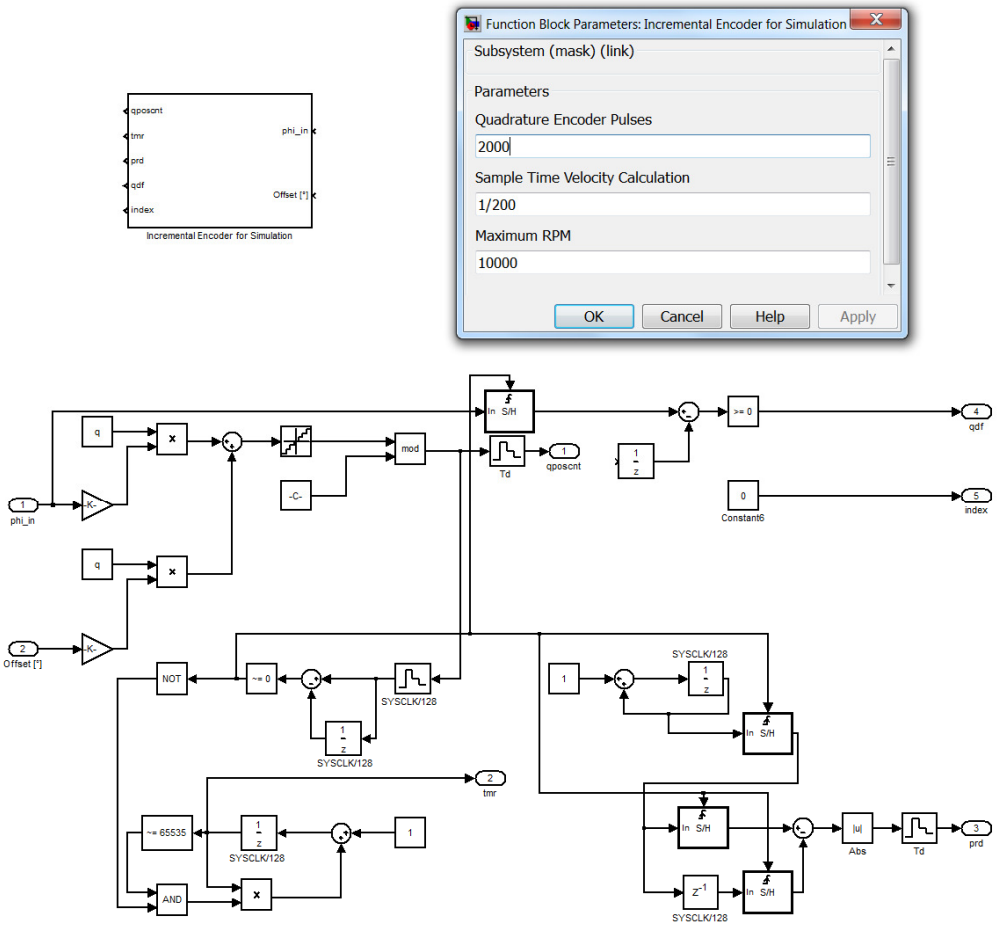


Abbildung 7: Funktionen und Maske der Inkrementalgebersimulation

Aus dem Grundwellenmodell der PMSM wird für die Inkrementalgebersimulation nur der mechanische Winkel ϕ benötigt. Am Eingang der Funktion werden die Winkel normiert und mit der Strichzahl multipliziert. Der Inkrementalgeberoffset wird zu dem mechanischen Winkel, in dem ein möglicher Versatz von Rotor und Stator schon eingebracht ist, hinzugezählt. Wird das Ergebnis noch auf ganzzahlige Vielfache beschränkt erhält man den simulierten Pulszähler ähnlich wie in der eQEP Einheit. Durch Anwendung der Modulo-Funktion mit dem Konfigurationswert des maximalen Encoderpuls-Zählerstandes auf diesen Pulszähler wurde die eQEP Ausgabe nachgebildet.

Der Timer-Zähler wird durch einen Zähler realisiert der wie der Zähler der eQEP-Einheit mit der Frequenz ft=SYSCLK/PS ganzzahlig hinaufzählt. Wird ein Encoderpuls erkannt (Differenz zwischen zwei mit ft abgetasteten Encoderpulszähler-Werten) oder der Zähler erreicht 2^16 wird der Zähler zurückgesetzt und man hat sofort auch den selben Timer wie er in der eQEP-Einheit zur Verfügung steht.

Um die Messung der Differenzzeit nachzubilden wird beim Autreten eines Encoderpulses jeweils der Zählerstand eines Zeitzählers gesampelt und die Differenz der letzten zwei gesampelten Werte gebildet. Es besteht auch noch die Möglichkeit einen Wert für die Indexpulsposition anzugeben. Dieser wurde (da nicht relevant) jedoch nicht genutzt. Das Quadrature Direction Flag wird durch den Vergleich der vorhergehenden kontinuierlichen Winkel mit dem aktuellen nachgebildet und ausgegeben.

In Abbildung 8 sieht man das Ergebnis der Simulation der Inkrementalgeber-Simulations-Blocks mit einer sinusförmigen Anregung: Die oberen 4 Diagramme zeigen die Ausgabe des Inkrementalgebers und die unteren 2 die Auswertung der simulierten Drehzahlmessung und den integrierten Drehzahlwert.

Man sieht dass alle Funktionalitäten erfüllt werden.

Es wird somit eine einfach konfigurierbare Drehzahlmessung, sowohl für die Simulation wie auch für die Integration auf der realen Hardware zur Verfügung gestellt.

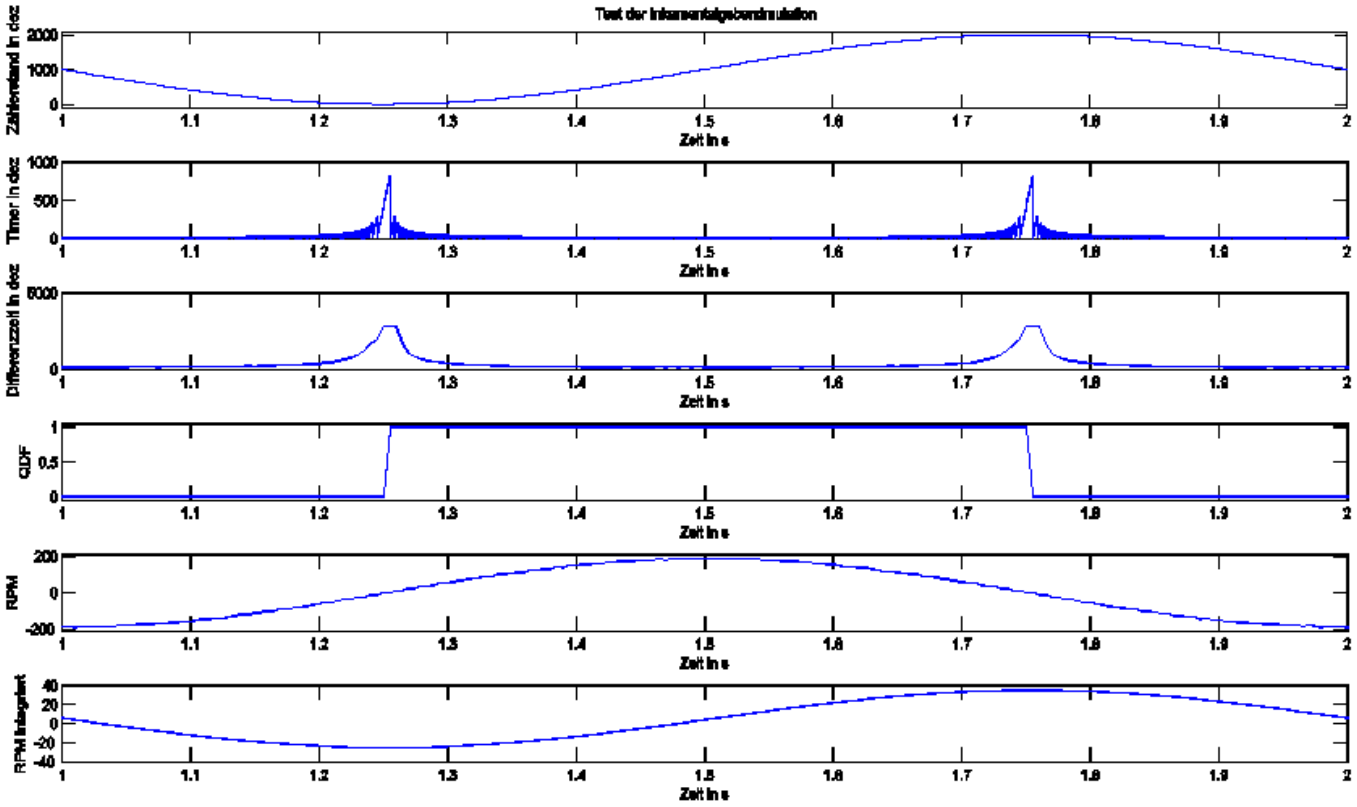


Abbildung 8: Test der Inkementalgebersimulation

4.2 Ein- und Ausgangsgrößen

Hier wird besprochen wie die Ausgangsgrößen in Rohform verfügbar sind und wie sie für die Weiterverarbeitung umgewandelt werden. Dabei werden die Skalierungen die sich durch die Hardwarebeschaltung der Messungen ergeben ebenfalls behandelt.

- Eingabetaster:

Auf der Hardware sind zwei Taster vorhanden die in der Software eingelesen werden können. Sie werden hier zum Starten der Testläufe verwendet.Die Taster sind an den digitalen Eingängen des TI F2808 angeschlossen und können durch den Block *Digital Input* aus der C280x-Library eingelesen werden.

Als Funktion wurden Öffner verwendet, darum werden die Signale sofort nach dem Einlesen invertiert. In der erstellten Library wird ein Block zur Verfügung gestellt der die Taster mit einer Abtastrate von $T = 50ms$ einliest und nach aussen Active-High Signale zur Verfügung stellt.

Im Bild 10 sieht man die erstellten Systeme aus dem Library-Block *Buttons.*

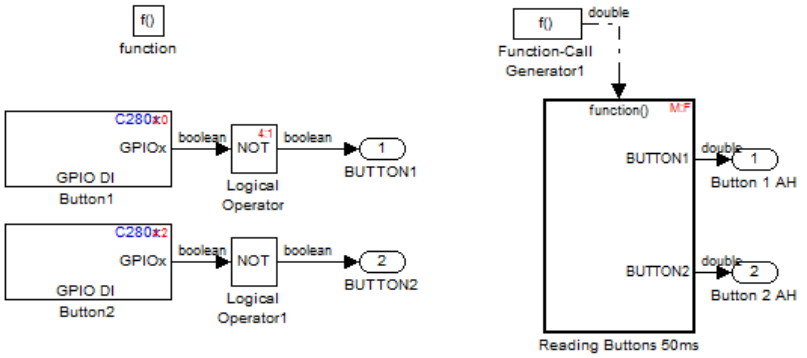


Bild 10: Einlesen der Taster

- Potentiometer:

Zur Durchführung von schnellen Experimenten mit Variablen stehen zwei Potentiometer zur Verfügung. Diese können zum Beispiel auch zur Skalierung und experimentellen Entwicklung der Reglerparameter dienen.Die Potentiometer werden über den Spannungsteiler am Potentiometerabgriff durch den ADC gemessen.

- Strommessung:

Der ADC des F2808 hat eine Auflösung von 12bit und arbeitet mit 3.3V-Technologie. Eine OPV-Verstärkerschaltung am Eingang skaliert den Strom und führt eine Nullpunktsverschiebung durch, daraus resultiert auf eine Auflösung von 12.77mA/digit mit einem daraus resultierenden Messbereich von -26.2 bis 26.2A.

In der Software wird dieser Wert zur Normierung der Ein- und Ausgangsgrößen genutzt.Dadurch wird sichergestellt dass unter der Voraussetzung einer richtig arbeitenden Regelschleife kein Wert über dem maximal Messbaren angeregelt werden kann.

Die Abtastzeitpunkte werden mit der PWM synchronisiert, genaueres dazu wird im Kapitel 5.5 beschrieben.

- Spannungsmessung:

Die Spanungsmessung wird ebenfalls über eine OPV-Verstärkerschaltung skaliert und man erhält in des DSP eine Auflösung von 21.3mV/digit. Um das Einlesen der Messwerte mit bestimmten Ereignissen koppeln zu können wird das System als *function-call-subsystem* ausgeführt, dies stellt ein Interrupt-getriggertes System dar.

Der Einleseblock und sein Innenaufbau ist in

Bild 11 zu sehen, dieser Block wird ebenfalls in der Library hinterlegt.

Genaueres zur Konfiguration des ADC findet man in [9].

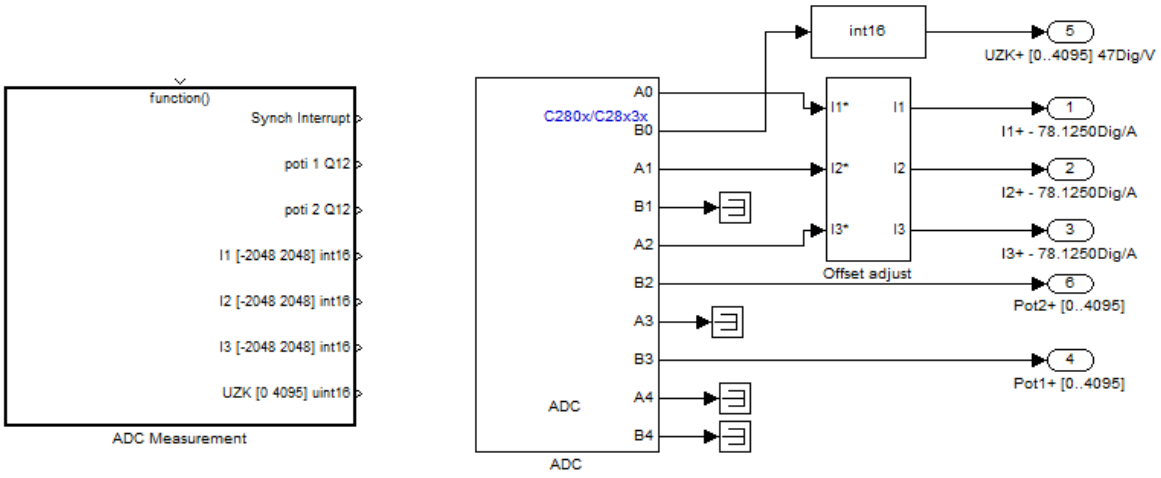


Bild 11: Einlesen der ADC-Werte, Funktionsblock und Aufbau

- Drehzahlmessung

Der Block zur Drehzahlmessung wird genauer imKapitel 4.1 vorgestellt.

- SVPWM bzw. Spannungsausgabe

Als Spannungsausgabe wird ein Block benötigt bei dem man Spannungen auf den Achsen eines Koordinatensystems vorgeben kann und den Winkel des Ausgangsspannungssystems ändern kann. Dazu werden wieder die Code-optimierten Blöcke von Texas Instruments herangezogen. Als Datenformet wird von TI das IQ-Format Q17 empfohlen und angewendet.

Im Bild 12 sieht man den Aufbau zum Block der Spannungsausgabe, der Block PWM scaling übernimmt dabei die Skalierung der Werte von -1 bis 1 auf die benötigten Ausgangsgröße. Genaueres dazu im Kapitel 5.5.

Die Spannungen am Eingang des Blocks müssen ebenfalls auf -1 bis 1 normiert sein. Die *Space Vector Modulation* von TI arbeitet mit einer Methode zur Nullpunktsverschiebung (addieren der 3. Harmonischen) zur besseren Ausnutzung der Zwischenkreisspannung.

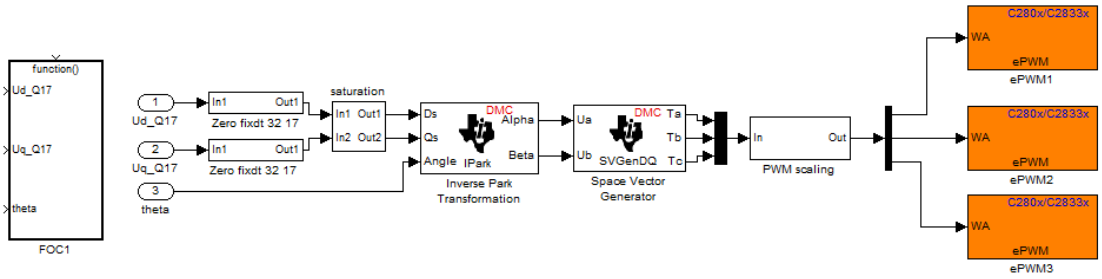


Bild 12: Spannungsausgabe, Funktionsblock und Aufbau

- CAN-Bus-Ausgabe

Um die nötigen Messwerte zu übermitteln werden verschiedene vorkonfigurierte Ausgabeblöcke zur Verfügung gestellt. In diesen, ebenfalls Interrupt-gesteuerten, Blöcken werden bis zu zwei 64 Bit Messages mit der möglichen Maximalfrequenz auf dem 1Mbit/s schnellen Bus übermittelt.

Einzelheiten zur Konfiguration findet man in [10].

Verfügbar sind folgende Blöcke:

- 8 x 16-bit Daten mit einer Frequenz von 2.5 kHz
- 4 x 32-bit Daten mit einer Frequenz von 2.5kHz
- 4 x 16-bit Daten mit einer Frequenz von 5kHz

Um die Blöcke in MATLAB einlesen zu können wurden dazugehörende m-Files geschrieben, diese können einfach mit der gewünschten Aufnahmezeit als Variable aufgerufen werden.

- Hardware-Interrupts

Um die verschiedenen Hardware-Interrupts des DSP nutzen zu können wird ebenfalls ein konfigurierter Block in der Library hinterlegt. In diesem Block ist *vorerst der ADC-End of Conversion-Interrupt* konfiguriert, dieser wird genutzt um den ADC rechtzeitig einzulesen.

Werden andere Interrupts benötigt müssen diese wie [7] beschrieben konfiguriert werden.

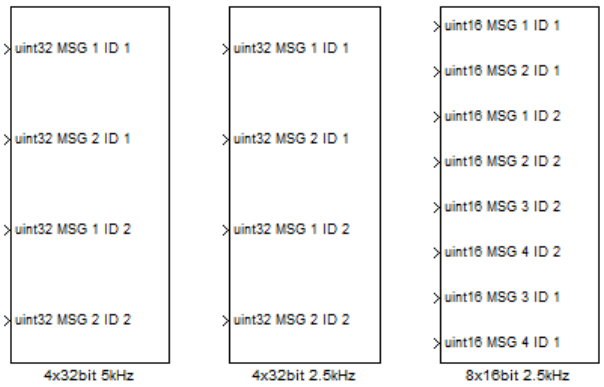


Bild 13: Verfügbare Blöcke für das CAN-Interface

- Alive-Signal

Auf der Hardware wurde zur Überwachung und zum Schutz der Leistungselektronik ein Controller integriert der die Ansteuerung der Leistungshalbleiter blockiert wenn von der DSP kein Alive-Signal gesendet wird. Das Senden dieses Signals wird ebenfalls in der Library als Block hinterlegt.

- PID-Controller

Es wurde in der gesamten Arbeit eine PID-Controller Architektur mit Anti-Windup angewandt. Der Aufbau des entworfenen Controllers ist in $Out = K * (ref - fb) * \left(Kp + Ki * Td * \frac{1}{z-1}\right)$ zu sehen. Der Controller erfüllt die Gleichung:

$$Out = K * (ref - fb) * \left(Kp + Ki * Td * \frac{1}{z-1}\right) \tag{4.10}$$

und ist für die Verwendung mit dem IQ Datenformat Q17 entworfen worden.

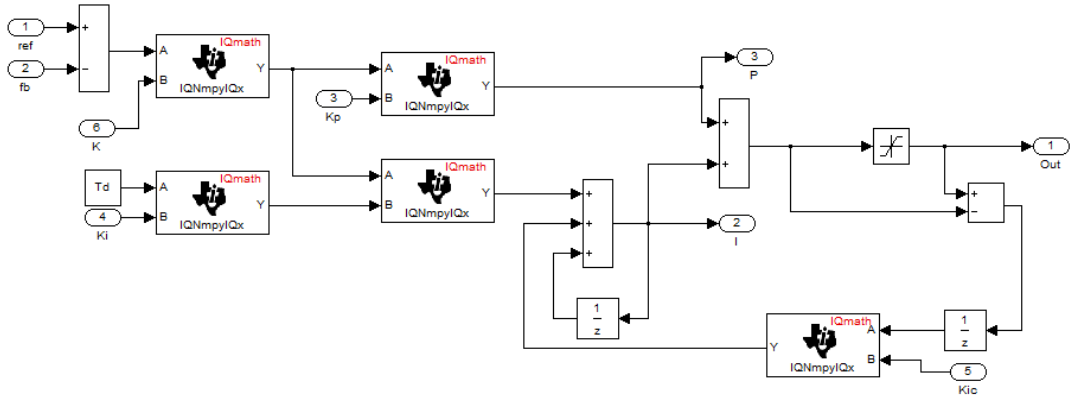


Bild 14: Aufbau des PID-Controllers

Der Controller-Block DMC PIC wurde mit einer Variablenmaske zur Bedatung der Abtastzeit und der Anti-Windup-Grenzen versehen und ist in der erstellten Library hinterlegt.

Damit ist die Betrachtung der allgemein nötigen Ein- und Ausgangsblöcke abgeschlossen, es wurde dargelegt wie die Größen eingelesen werden können, wie sie skaliert sind und welche Randbedingungen gelten, alle vorgestellten Blöcke können aus der erstellten Bilbliothek entnommen und direkt kompiliert werden. Im Folgenden werden die erstellten Versuche zur Parameteridentifikation gezeigt und beschrieben.

4.3 Programm zur Encoderoffset- und Widerstandsbestimmung

Wie in 3.3, Bestimmung des Ständerwiderstandes, schon beschrieben wurden der Versuch zur Identifikation der Encoderoffsetbestimmung und das Programm zur Widerstandsbestimmung zu einem Versuch kombiniert.

Im Bild 15 sieht man den gesamten Simulink-Koppelplan des Systems (ohne CAN-Messwertübergabe).

- Voltage Control

Im Block *Voltage Control* wurden die Funktionen entwickelt die imKapitel 3.3 mit dem Block *Logik* dargestellt sind. Eingangsgrößen sind die Ströme im Ständerkoordinatensystem, diese werden nur gebraucht um die Länge des Stromvektors zu bilden und zu begrenzen. Ausserdem ist ein Eingang für eine boolsche Variable zum Starten des Versuchs vorhanden. Bei gestartetem Versuch wird der Wert gehalten. Zum Nochmaligen Starten muss wieder eine Flanke erkannt werden.

Da der Versuch der Offsetbestimmung sehr robust sein soll wurde hier dem Anwender wenig Einstellmöglichkeit überlassen, er wird nur über die gewünschte Abtastzeit konfiguriert. Selbst die sollte nicht zu gernig gewählt werden da es mit gröberer Zeitdiskretisierung auch in der Rampenbildung zu größeren Spannungssprüngen kommen würde und damit dem Sinn der Rampe, durch den langsamen Anstieg die Möglichkeit zu haben schon bei geringem Moment zu regeln, entgegengearbeitet würde.

Die Ausgänge sind:

- die Spannungsrampe zur Vorgabe der Spannung auf der reellen Achse des Koordinatensystems, Zahlenformat Q17
- Eine boolsche Variable um den PI-Controller vor Einsetzen der Widerstandsmessung zu deaktivieren
- Eine boolsche Variable die den Zeitpunkt der Widerstandsmessung kennzeichnet
- Die Stromvektoramplitude zum Verringern der Verstärkung mit steigendem Strom.

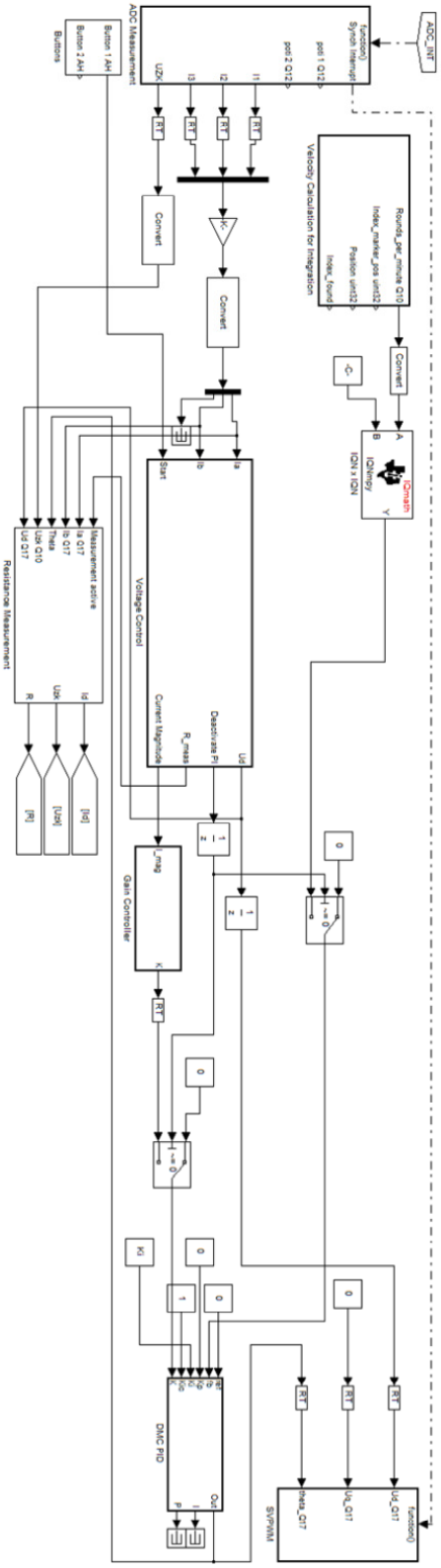


Bild 15: Offset- und Widerstandsbestimmung Koppelplan

- Gain Control

Durch den *Gain Controller* wird die Gesamtverstärkung des PI-Winkelcontrollers quadratisch mit dem steigender Stromamplitude I gesenkt.

Mit $x = \frac{I_{max}}{2} - I$ und $a = \frac{4}{{I_{max}}^2}$ wird nach der Formel

$$K = a * x^2 \qquad (4.11)$$

die Reglerverstärkung gesenkt.

Diese Absenkung hat sich nach empirischen Tests gegenüber einer linearen Absenkung als stabilere Lösung präsentiert.

- Resistance Measurement

Im Block Resistance Mesurement wird die Berechnung des Widerstandes durchgeführt.

Bei der Berechnung wurde darauf geachtet, dass es in den Bereichen in denen sich die Zwischenkreisspannung und die Ströme bewegen werden zu keinen Überläufen des Datentyps bei den Divisionen und Multiplikationen kommt.

Die Messung wird durch das Signal R_meas aus dem Block Voltage Control gestartet und es wird für n-Samples der Mittelwert nach

$$R = \frac{1}{n} * \sum_{1}^{n} \frac{U_{dn}}{I_{dn}} \qquad (4.12)$$

gebildet.

Wurde der Encoder-Offset und der Widerstand festgestellt werden die Werte an den Induktivitäts-Identifikationslauf übergeben. Ebenso erhält die Induktivitätsmessung den Spannungsmaximalwert um den Identifikationspuls auszulegen.

4.4 Programm zur Identifikation der Induktivität

Die Grundlagen bzw. die Idee zu diesem Programm werden im Abschnitt 3.4, Bestimmung der Induktivität, beschrieben. Dort wird gezeigt dass es einen Task gibt der den Versuch und die Messung durchführt und einen sogenannten Optimierungs-Task.

Den Koppelplan des Identifikationsversuchs für die Induktivität sieht man im Bild Bild 17. Es werden prinzipiell dieselben Aus- und Eingabeblöcke wie bei der Widerstandsidentifikation benötigt, die Funktionalität des Tests steckt in den Blöcken *Identification Pulse* und *Inductance Identification.*

Hier übernimmt, wie der Name verrät, der Block *Identification Pulse* die Aufgabe das Testsignal zu generieren und im Block *Inductance Identification* stecken die vorher besprochenen Tasks. Der Puls wird dabei durch einen einfachen Zähler und einige logische Abfragen generiert, Sobald der Puls ausgegeben wurde wird auch das System nicht mehr berechnet, dadurch kann wieder Prozessorauslastung für den Optimierungsprozess gespart werden. Die zwei verschiedenen Tasks werden im Folgenden aufgeschlüsselt.

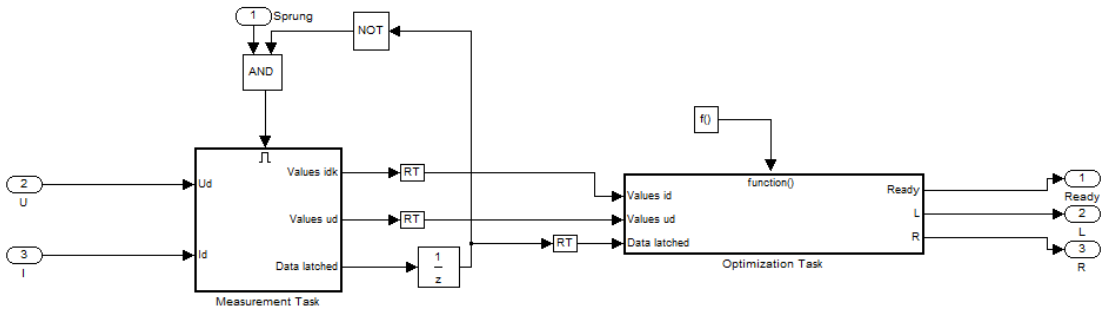


Bild 16: Aufbau Block Inductance Identification

Im Bild 16 sieht man das Innenleben des Blocks *Inductance Measurement.* Man sieht sofort dass das Aufnehmen der Messdaten nur 1-mal erfolgt und der Task danach nicht mehr ausgeführt wird da er mit dem Zustand *Data latched* verbunden ist. Nach dem Ausführen des Messtasks liegen am Ausgang des Blocks zwei Vektoren mit bestimmter Länge und den Messdaten vor.

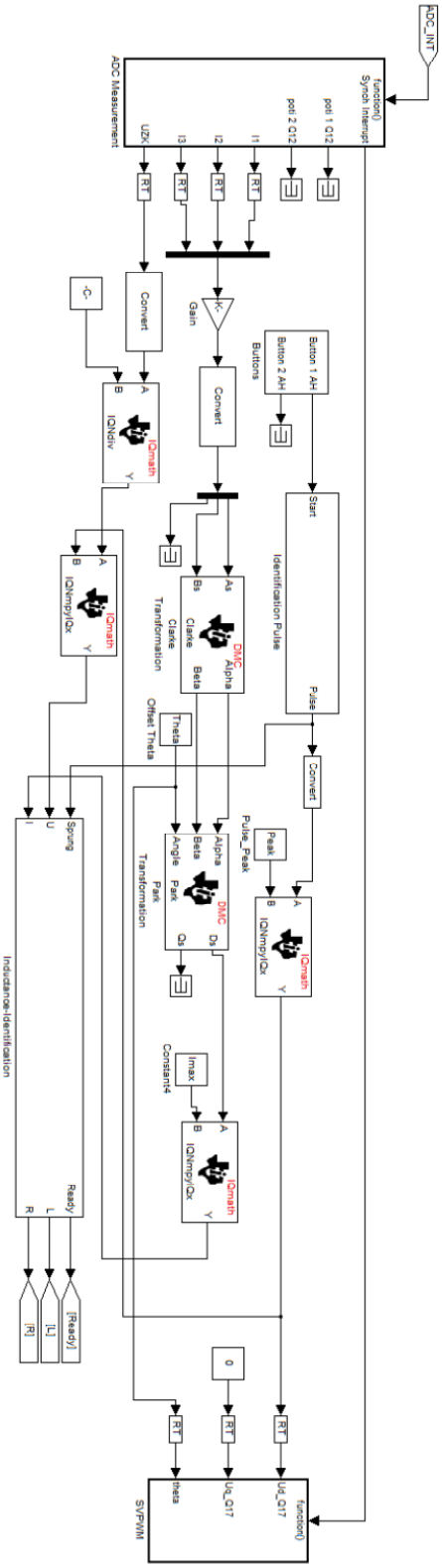


Bild 17:Induktivitätsbestimmung Koppelplan

Dabei wird im Messtask der Spannungsanstieg erkannt und der kommende Stromanstieg sowie der Spannungsanstieg selbst so lange aufgenommen bis der Strom seinen stationären Wert erreicht hat.

Selbiges gilt für den Optimierungs-Task, dieser hat als Vorgabe einen gewissen Parameterfehlertoleranzparameter Epsilon. Sobald diese Toleranzgrenze unterschritten wird ändert sich das Ausgangssignal nicht mehr. Es wird überprüft ob das Ausgangssignal für über 5 Optimierungsschritte konstant bleibt, ist dies gegeben wird die Optimierung abgebrochen. Somit wird auch abgebrochen wenn die Epsilon-Schranke nicht erreicht wird, jedoch durch z.B. die Begrenzung durch die Genauigkeit des Datentyps nicht erreicht werden kann.

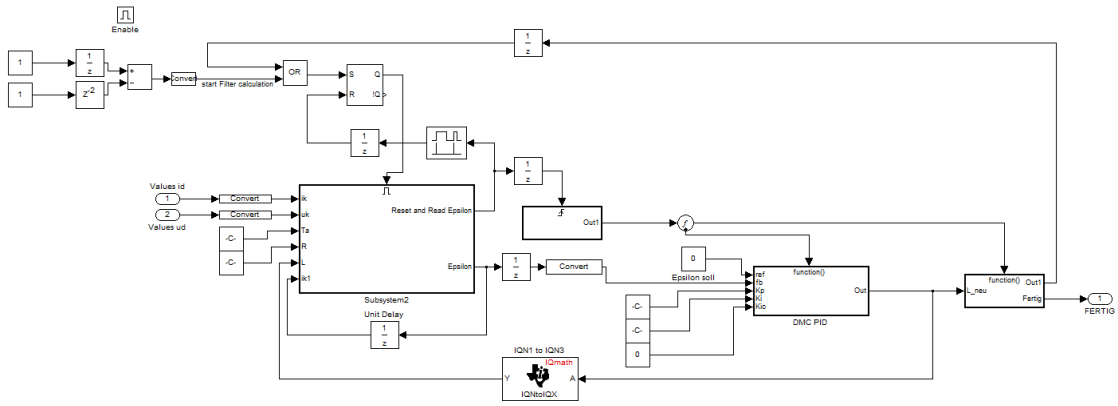


Bild 18: Aufbau der Parameteroptimierung

In Bild 18 sieht man den inneren Aufbau der Parameteroptimierung für den Induktivitätstest.

Man sieht hier dass für die genaue Bestimmung von L ebenfalls wieder ein PI-Controller eingesetzt wird. Dem Aufbau des verwendeten Filters wurde hier mehr Aufmerksamkeit geschenkt, durch die Verwendung des Fixkomma-Datenformats kann es hier zu großen Fehlern in der Berechnung kommen wenn man das falsche IQ-Format verwendet.

In den Bildern Bild 19, Bild 20 und Bild 21 sieht man noch einmal genau den Aufbau des Filters und wie die Berechnung der Koeffizienten A und B aus dem Kapitel 3.4 vorgenommen wird. Der Regler für die Optimierung ist sehr einfach einzustellen, weiß man die gewünschte Genauigkeit des Parameters wählt man als Feedback-Format und für die Verstärkungen Kp und Ki einfach Minimalwerte in diesem Format. Hat man Ansprüche an eine schnellere Konvergenz muss Ki entsprechend erhöht werden.

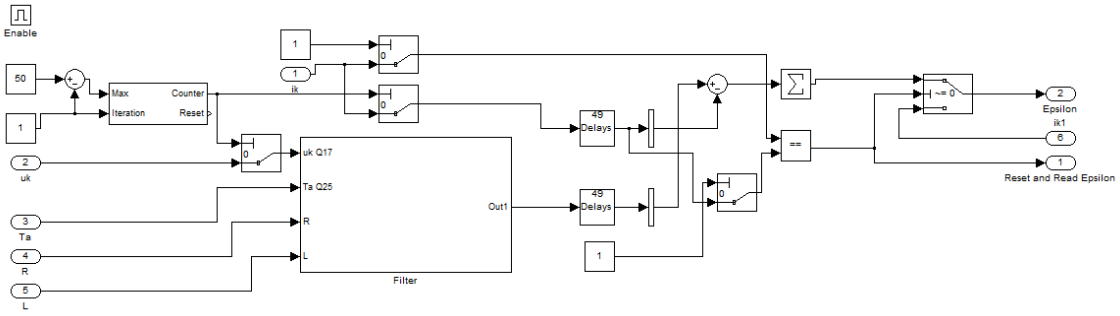


Bild 19: Berechnung des Filterergebnisses und der Toleranzschwelle

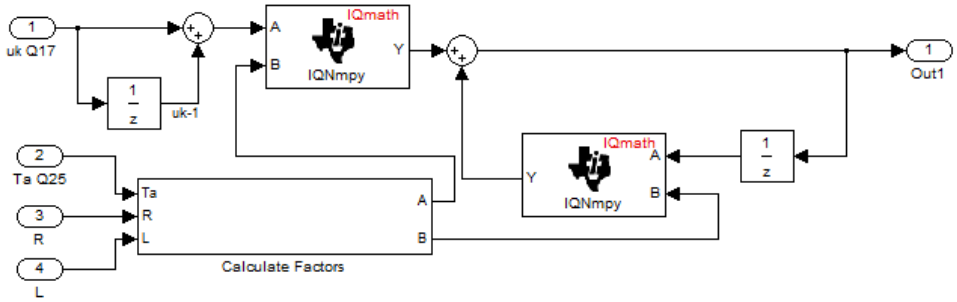


Bild 20: Block "*Filter*"

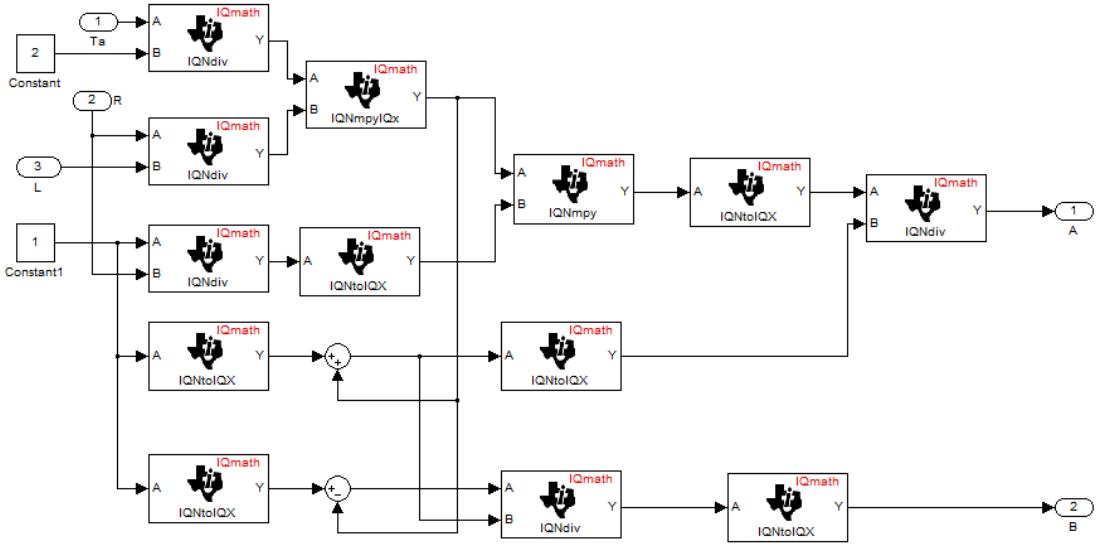


Bild 21: Block "*Calculate Factors*"

Kann durch zu groß gewählte Parameter die gewünschte Genauigkeit nicht erreicht werden bricht die Optimierung dann ab wenn keine Änderung der Optimierungsvariablen mehr auftritt.

4.5 Die Identification Library

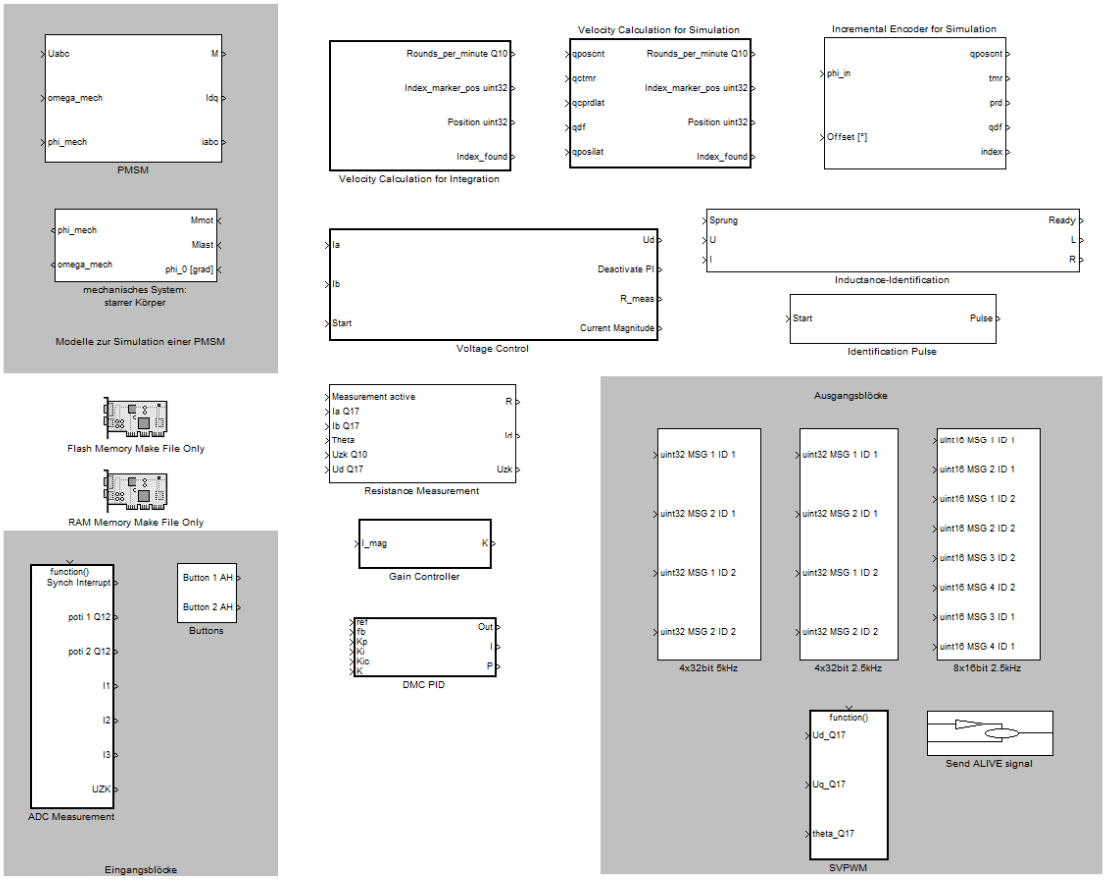


Bild 22: Die erstellte Library zur Motoridentifikation

5 Toolchain

5.1 Verwendete Softwarepakete

In diesem Kapitel werden die in dieser Arbeit benötigten Software-Pakete vorgestellt und beschrieben welche Schritte unternommen werden müssen um die benötigte Software-Werkzeugkette zu konfigurieren. Die verwendete Software ist:

- The Mathworks: MATLAB/Simulink R2011b 64-bit
- Teaxs Instruments: Code Composer Studio v5
- Windows 7 64-bit
- Windows SDK 7.1
- Java Runtime Environment

MATLAB ist ein Programm zur Lösung mathematischer Probleme und primär für die numerische Berechnung mithilfe von Matritzen ausgelegt. Die Abkürzung MATLAB steht für Matrix Laboratory. Im Softwarepaket Simulink können Systeme grafisch modelliert und zeitgesteuert simuliert werden. Dabei wird prinzipiell zwischen zeitkontinuierlichen und zeitdiskreten Systemen unterschieden. MATLAB stellt für diese Systeme eine Reihe von Lösungsalgorithmen zur Verfügung, dabei werden Einschritt- und Mehrschrittverfahren unterschieden. Auf die Unterschiede dieser Verfahren soll in dieser Arbeit nicht eingegangen werden.Simulink ist dadurch eine geeignete Plattform zur Systemsimulation und zur Reglerauslegung. Mit der Zusatzsoftware Emdebbed Coder ist es ausserdem möglich direkt Programmcode für eine bestimmte Zielhardware aus dem grafischen Systemmodell zu generieren. Dabei sind für verschiedenste Microcontroller Blöcke zur optimierten Codegenerierung erhältlich.

In dieser Arbeit wird die Software-Toolbox Embedded Coder als Grundlage zur Codegenerierung für einen fixed point Microcontroller von Texas Instruments verwendet. Embedded Coder

stellt für diese Controller verschiedene laufzeitoptimierte Funktionen zur Verfügung. Für die Anwendung in der Motor-Regelung sind dabei folgende Funktionsbibliotheken wichtig:

- Digital Motor Control-Bibliothek:

Die DMC-Library stellt laufzeitoptimierte Standardfunktionen für die Controllerserie TI C28x zur Verfügung. Darunter finden sich die benötigte Park- sowie die Clarke-Transformation (und ihre Inversen), die Raumzeigermodulation, sowie vorgefertigte PID-Regler-Konstrukte. Um während der Laufzeit die Möglichkeit offen zu halten auf die Stellgrößenbeschränkungen des Reglers Einfluss zu nehmen wird in dieser Arbeit ein in Simulink diskret aufgebauter PID-Regler verwendet.

- IQ-Mathematik-Bibliothek

Die IQ-Mathematik ist eine Sammlung von optimierten hochgenauen Funktionen zur Berechnung von mathematischen Operationen im fixed point Format.

Die Buchstaben I und Q bezeichnen hierbei das verwendete Datenformat bzw. die Nachkommastellen. Es wird für die verwendeten Variablen mit dem Q-Wert die Position der Kommastelle definiert.

Dies kann bei bestehendem Wissen über die Größenordnung der auftretenden Werte dazu genutzt werden die Auflösung bei fixed point Berechnungen möglichst hoch zu halten. Das Format einer 32-bit Zahl mit nur einem Bit als Kommastelle wird z.B. als Q1 bezeichnet.

In Tabelle 1 sind die verfügbaren Formate zusammengefasst und die jeweilige Auflösung aufgelistet.Die Umwandlung einer Zahl im Integer-Format auf eine fixed point Zahl im IQ-Format kann in MATLAB sehr einfach mit einer sog. Data Type Conversion anschaulich dargestellt und ausgeführt werden, dabei ist jedoch auch auf die Art der Umwandlung (Data Type Conversion Parameters) zu achten:

- Input and Output have equal Real World Value:

Die Wertigkeit der Ausgangs im Q-Datenformat entspricht soweit es die Genauigkei tzulässt dem Wert des Eingangs im jeweiligen Datenformat.

IQ-Format	Vorkomma-Auflösung	Nachkomma-Auflösung
Q1	1073741824.0	0.500000000000000000000000
Q2	536870912.0	0.250000000000000000000000
Q3	268435456.0	0.125000000000000000000000
Q4	134217728.0	0.062500000000000000000000
Q5	67108864.0	0.031250000000000000000000
Q6	33554432.0	0.015625000000000000000000
Q7	16777216.0	0.007812500000000000000000
Q8	8388608.0	0.003906250000000000000000
Q9	4194304.0	0.001953125000000000000000
Q10	2097152.0	0.000976562500000000000000
Q11	1048576.0	0.000488281250000000000000
Q12	524288.0	0.000244140625000000000000
Q13	262144.0	0.000122070312500000000000
Q14	131072.0	0.000061035156250000000000
Q15	65536.0	0.000030517578125000000000
Q16	32768.0	0.000015258789062500000000
Q17	16384.0	0.000007629394531250000000
Q18	8192.0	0.000003814697265625000000
Q19	4096.0	0.000001907348632812500000
Q20	2048.0	0.000000953674316406250000
Q21	1024.0	0.000000476837158203125000
Q22	512.0	0.000000238418579101562000
Q23	256.0	0.000000119209289550781000
Q24	128.0	0.000000059604644775390600
Q25	64.0	0.000000029802322387695300
Q26	32.0	0.000000014901161193847700
Q27	16.0	0.000000007450580596923830
Q28	8.0	0.000000003725290298461910
Q29	4.0	0.000000001862645149230960
Q30	2.0	0.000000000931322574615479

Tabelle 1: IQ-Formate und Auflösung

- Input and Output have equal Stored Integer

Entspricht einer Verschiebung der Kommastelle. Wird z.B. Die Zahl 2048 (2^11) mit dieser Einstellung in eine Q11-Zahl umgewandelt so ist ihre Wertigkeit 1.

Im Laufe der Arbeit wurden die meisten Blöcke dieser Bilibliothek bezüglich der Äquivalenz ihrer Funktion in der Simulation und auf der Hardware überprüft, einige Ergebnisse werden später angeführt.

5.2 Konfiguration von Code Composer Studio

Code Composer Studio (CCS) muss nach der Installation auf die verwendete Hardware konfiguriert werden. Dies soll hier kurz beschrieben werden.

Der verwendete Controller ist ein *TI F2808* und als Programmierschnittstelle wird JTAG mit einem *XDS510LC JTAG Emulator* verwendet.

Dies muss in CCS als „Target Configuration“ eingestellt werden. Eine neue Konfiguration wird nach starten von CCS unter dem Menüpunkt Target/New Target Configuration angelegt (Bild 23)

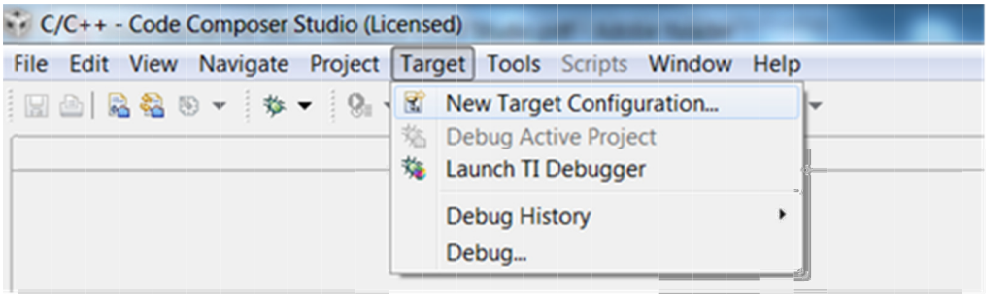


Bild 23: Anlegen einer neuen Konfiguration in CCS

Es wird nach einem geeigneten Speicherort gefragt, da man die generierte Datei jedoch später händisch in den Projektordner kopieren muss ist dieser frei wählbar. Als nächstes wird der JTAG-Emulator im Bild 24 zu sehenden Menü definiert. In selben Menü wird der Controller definiert (Bild 25). Im vorgestellten Fall wird als Device EZDSPF2808 gewählt da die vorliegende Pinbelegung mit dieser Konfiguration übereinstimmt.

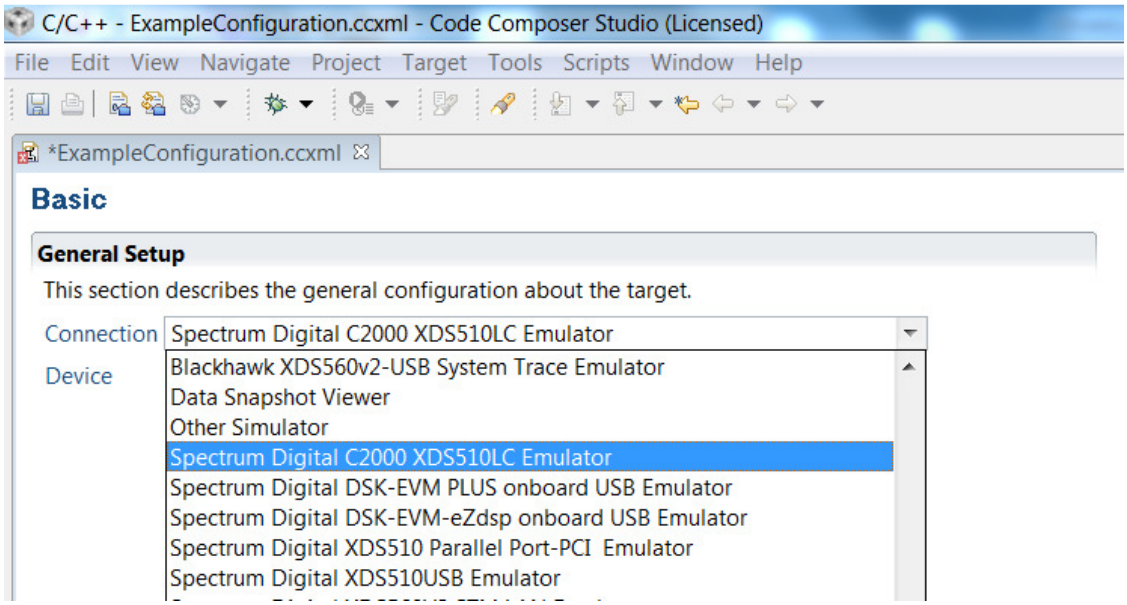


Bild 24: Schnittstellendefinition in CCS

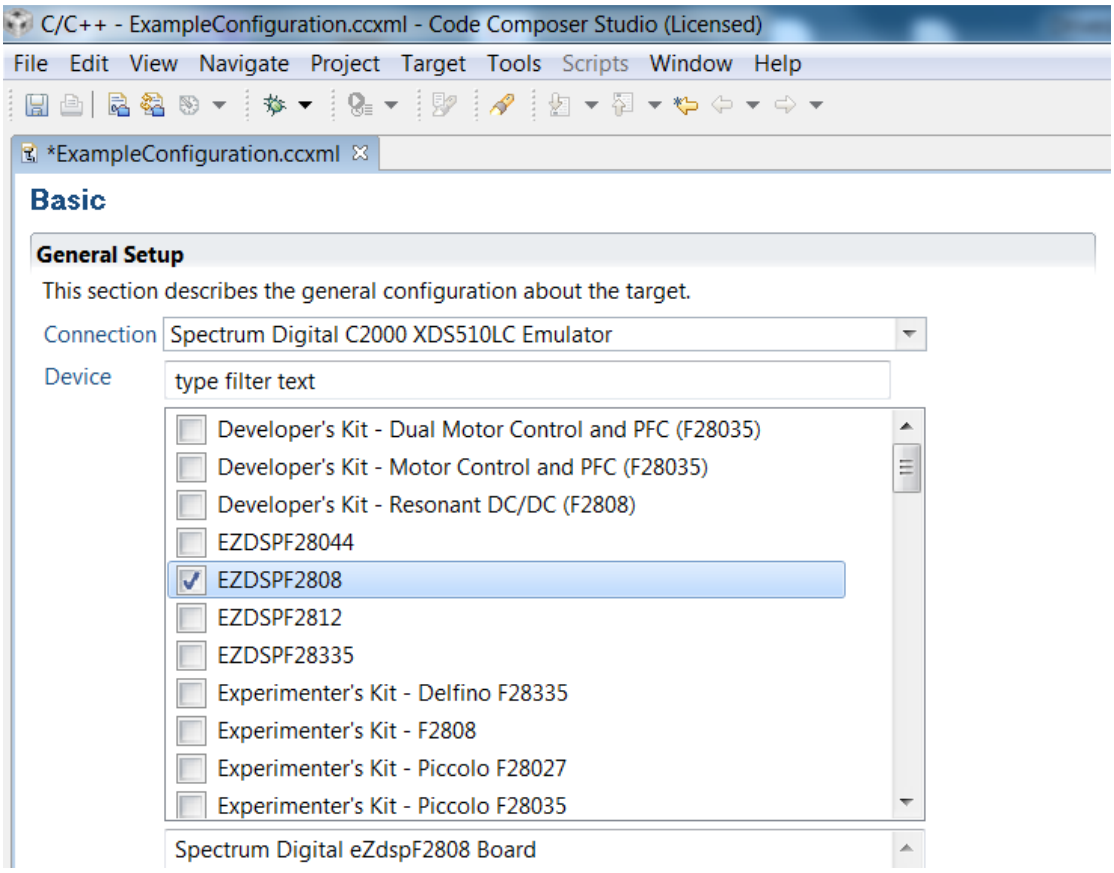


Bild 25: Controllerdefinition in CCS

Diese Konfiguration kann nun im angegebenen Ordner gespeichert werden. Die erstellte Datei (Endung .ccxml) muss später in den Pfad des Simulink-Modells das kompiliert werden soll gespeichert werden. Dieser Pfad wird im Folgenden mit <**Modelpath**> bezeichnet.

5.3 Konfiguration von MATLAB

In MATLAB muss nun der gewünschte Compiler eingestellt und konfiguriert werden. Ebenfalls kann durch das verwenden eines Java-Scripts der Ladevorgang eingebunden werden.

Führt man im MATLAB Command Window den Befehl "mex –setup" aus so kann der Standard-Compiler eingestellt werden. Wie bei der verwendeten Software schon angemerkt wird in dieser Arbeit das Microsoft Software Development Kit (SDK) 7.1 verwendet.

Dieselbe Einstellung muss für Standalone-Files mit dem Befehl „mbuild –setup" getroffen werden. Um dem Emdebbed Coder das Kompilieren und Laden von Makefiles zu ermöglichen muss man den Compiler, den Linker und das verwendete Script unter dem Befehl „xmakefilesetup" angeben. Es wird ein GUI geöffnet, zu sehen in Bild 26..

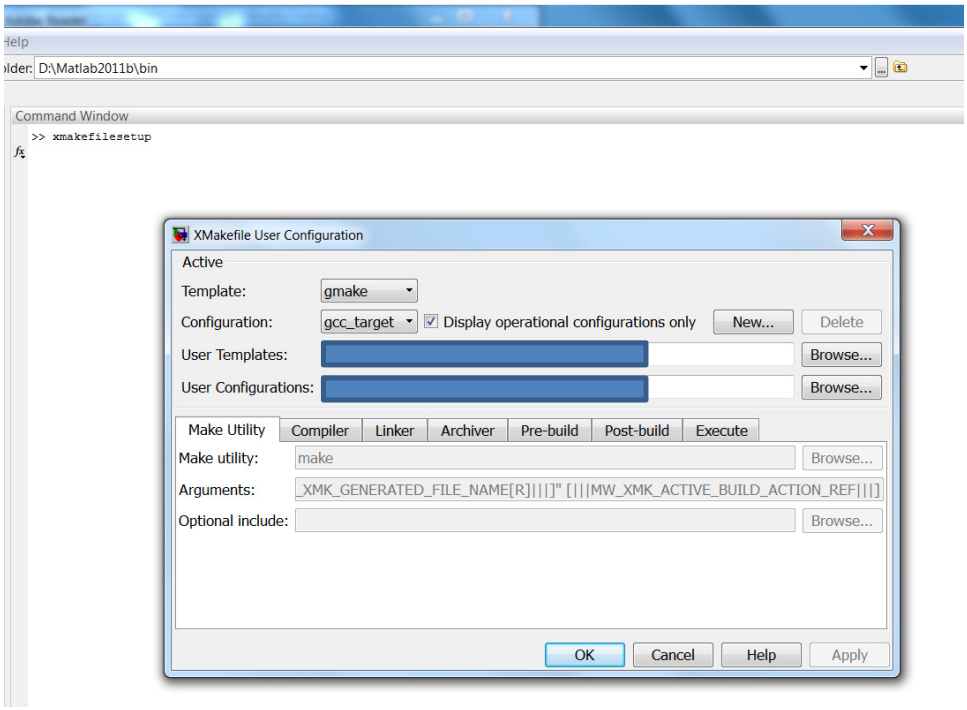


Bild 26: Makefile-Setup im MATLAB (xmakefilesetup)

Beschreibung der Schritte für die Makefile-Konfiguration:

- Die Markierung des Kästchens „Display operational configurations only“ aufheben.
- Unter Configuration: „ti_c2000_ccsv4“ auswählen (funktioniert genauso für CCSv5), “Apply” anklicken
- Falls das GUI vorher noch nie benutzt wurde wird jetzt nach dem Installationspfad von CCS gefragt (z.B..: C:\TI\ccsv5).
- Nun werden unter dem Reiter „Tool directories“ das nun verfügbar ist die CCS5-Installationspfade angegeben:

    - CCS Installation: C:\TI\ccsv5\
    - Code generation tools: C:\TI\ccsv5\tools\compiler\c2000\
    - DSP/BIOS Installation: C:\TI\ccsv5\

- Auf “Apply” klicken: GUI nicht schließen!
- Nun wird die neue Konfiguration abgespeichert:
  Auf „New“ klicken, und z.B.: die Konfiguration unter "ticcs_c2000_ccsv4_clone" speichern.

Nun sind alle Reiter (Compiler, Linker, etc..) editierbar und können wie folgt eingestellt werden: (Der MATLAB-Installations-Pfad wird dabei als <ROOT> angegeben, der PFAD von CCS als <CCSPATH>, der Pfad in dem die in CCS erstellte Konfiguration liegt mit <CONFIGPATH>

- Reiter „Make utility“

Make utility: <ROOT>\bin\win64\gmake

Optional include: <CCSPATH>\ccsv5\tools\compiler\c2000\include

- Reiter “Compiler”:

Compiler: <CCSPATH>\ccsv5\tools\compiler\c2000\bin\cl2000

Arguments: -I"include" -fr"[|||MW_XMK_DERIVED_PATH_REF|||]"

Source: .c,.asm,.abs,.sa

Header: .h

Object: .obj

- Reiter “Linker”:

Linker: <CCSPATH>\ccsv5\tools\compiler\C2000\bin\cl2000

Arguments: -o [|||MW_XMK_GENERATED_TARGET_REF|||]

File extensions for library files: .lib,.cmd

Generated output file extension: .out

- Reiter “Archiver”:

Archiver: <CCSPATH>\ccsv5\tools\compiler\C2000\bin\ar2000

Arguments: -r [|||MW_XMK_GENERATED_TARGET_REF|||]

Generated output file extension: .lib

- Reiter “Execute”

Hier wird der Ladevorgang über ein Java-Script automatisiert.

Execute tool:<CCSPATH>\ccsv5\scripting\bin\dss.bat

Arguments: "<ROOT>\toolbox\idelink\extensions\ticcs\ccsdemos\runProgram.js"
"<CONFIGPATH>\NewTargetConfiguration.ccxml"
"[|||MW_XMK_GENERATED_TARGET_REF[E]|||]"

5.4 Konfiguration von Simulink

Wenn die vorher beschriebenen Tätigkeiten genau so durchgeführt und damit die Software richtig konfiguriert wurde wird in MATLAB ein Modell in Simulink erstellt das auf dem Controller ausgeführt werden soll. Der Modellpfad wird als <MODELPATH> bezeichnet.

Um Code für die Zielhardware generieren zu können muss diese ebenfalls in dem Modell definiert werden. Dies kann einfach durch das Hinzufügen des Blocks „Target preferences“ aus dem Simulink Blockset Embedded Coder gemacht werden. Wird der Block das erste Mal zu einem Model hinzugefügt öffnet sich automatisch ein GUI, zu sehen in Bild 27. „IDE/Unter Toolchain“ ist hier wie schon zuvor nur CCSv4 verfügbar, diese Konfiguration funktionier in CCSv5 jedoch ebenfalls. Wie schon im Abschnitt 5.2 wird auch hier die Einstellung SD F2808 eZdsp ausgewählt. Bestätigt man die Auswahl werden die Parameter in der Konfiguration von Simulink bereits richtig gesetzt.

Hat man eine andere Hardware oder will man nicht den ganzen RAM für Programmcode verwenden etc. kann dies im Target preferences Block im Reiter „Memory“eingestellt werden. Dort kann man auch definieren ob man den Programmcode in den RAM oder in den Flash-Speicher laden möchte um ihn permanent zu speichern.Während der Entwicklung der Programme wird normalerweise nur der RAM-Speicher verwendet da der Flash-Speicher eine begrenzte Anzahl an Schreibezyklen aufweist und normalerweise erst bei einem fertigen Produkt zum Einsatz kommt.

Auch die Grundeinstellungen der ADC-Einheit, der PWM-Einheit sowie das CAN Timing werden in diesem Block unter dem Reiter „Peripherals“ vorgenommen. Darauf wird im Kapitel 5.5 genauer eingegangen.

Ab diesem Zeitpunkt ist die Software-Entwicklungskette voll funktionsfähig und es kann Code aus Simulink direkt auf den Mikrprozessor kompiliert werden.

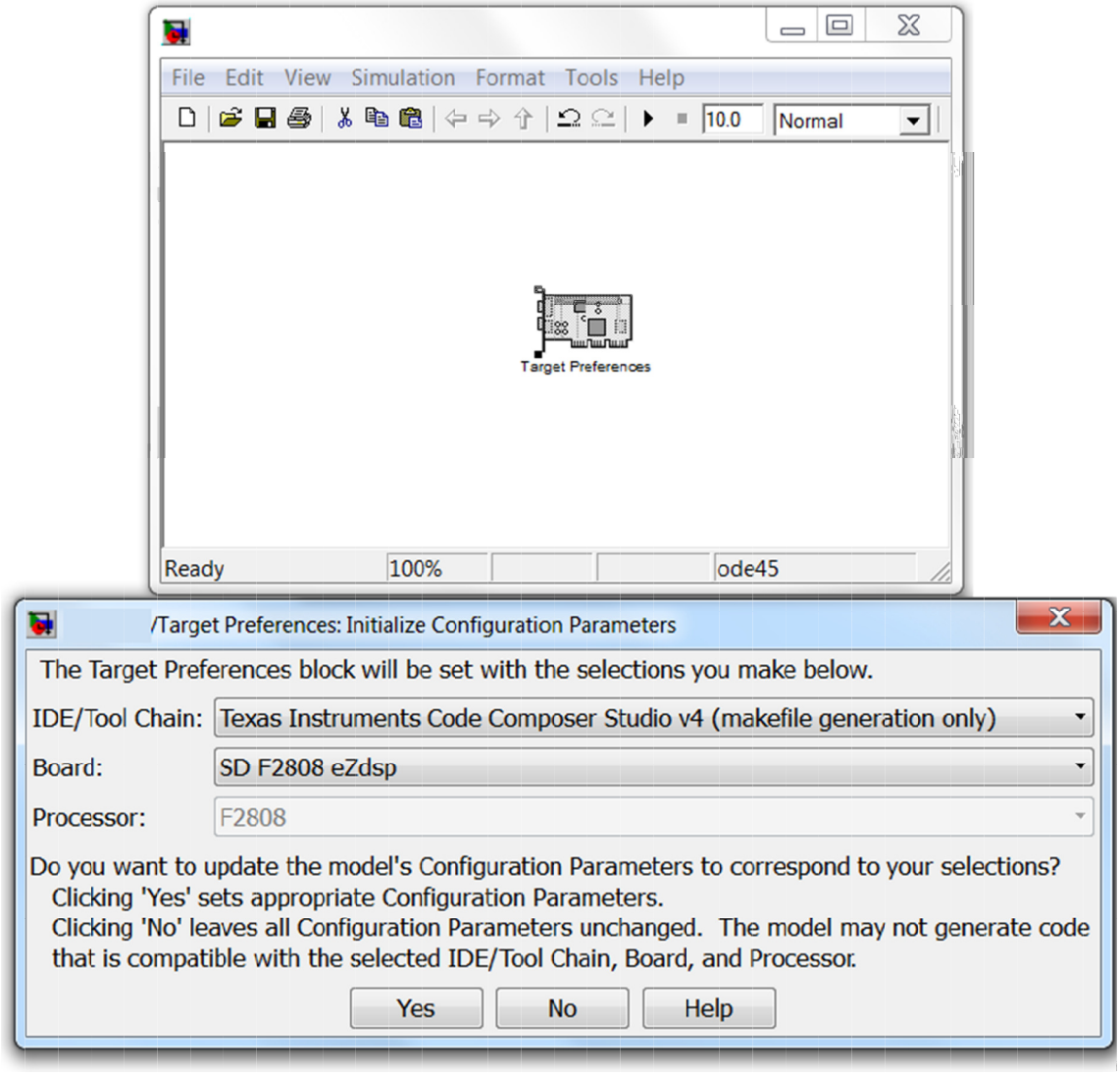


Bild 27: Definieren der Zielhardware in Simulink

5.5 Konfiguration der Controller-Peripherie

Wie im vorhergehenden Kapitel erwähnt werden die Grundeinstellungen für die Konfiguration der Controller Peripherie in Simulink durch den Block *Target preferences* vorgenommen. Gerade bei Anwendungen im Gebiet Motor Control ist wegen der ohmsch-induktiven Lasten ein besonderes Augenmerk auf ADC-Abtastzeitpunkte und die PWM-Generierung gelegt, deshalb wird hier darauf eingegangen:

- ADC-Einheit:

Die ADC-Einheit wird durch 3 Parameter spezifiziert: durch den Acquisition-Prescaler ACQ_PS, den ADC Clock Prescaler ADCLKPS und den Clock Presclaer CPS. Diese Parameter werden zur Ableitung des ADC-Messfensters und der ADC-Abtastgeschwindigkeit von der High Speed Peripheral Clock HISPCLK benötigt.

Die HISPCLK wird ebenfalls über einen Parameter vom Systemtakt SYSCLK abgeleitet. Dieser high-speed peripheral prescaler ist in Embedded Coder standardmäßig auf 1 gesetzt und das resultiert in einer Teilung des Systemtakts durch 2. Will man hier einen anderen Teiler einsetzen müssen die von Embedded Coder generierten Dateien in einem Zwischenschritt in CCS bearbeitet werden.

Im vorliegenden Fall ist HISPCLK also $\frac{SYSCLK}{2} = \frac{100MHz}{2} = 50MHz.$

Die weiteren Teiler sind im Bild 28 grafisch dargestellt. Die Prescaler sind dabei nullindiziert.

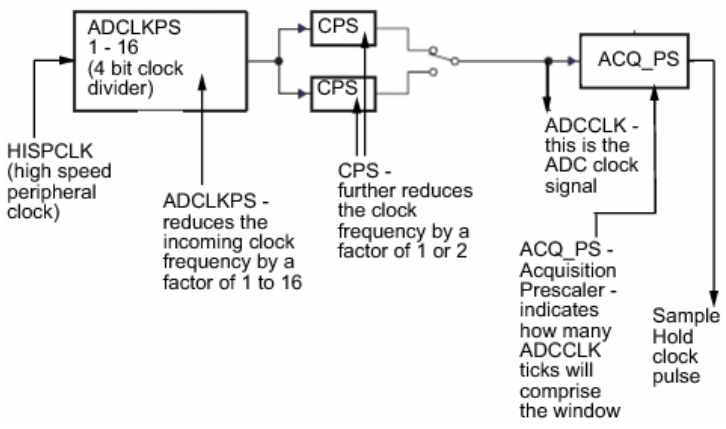


Bild 28: Teiler zur ADC-Konfiguration

Beispielhafte Berechnung:

Beträgt die HISPCLK z.B.: 100MHz und ADCLKPS=3 (was einem Teiler von entspricht) wird der Takt auf 12.5MHz heruntergeteilt

Is CPS=1 wird dieser Takt noch einmal geteilt und enspricht 6.25MHz.

Mit dem ACQ_PS wird nun festgelegt wieviele Takte das Messfenster des ADC offen bleibt. Ist ACQ_PS in diesem Fall z.B.: gleich 4 so erhält man ein Fenster von 0.64µs.

Weiters ist noch zu berücksichtigen dass es eine Zeitverzögerung zwischen dem Sampling und dem Bereitstehen des Messergebnisses sowie die Verzögerung durch das Sampling selbst gibt. Die verschiedenen Verzögerungen sind in Bild 29 zu sehen.

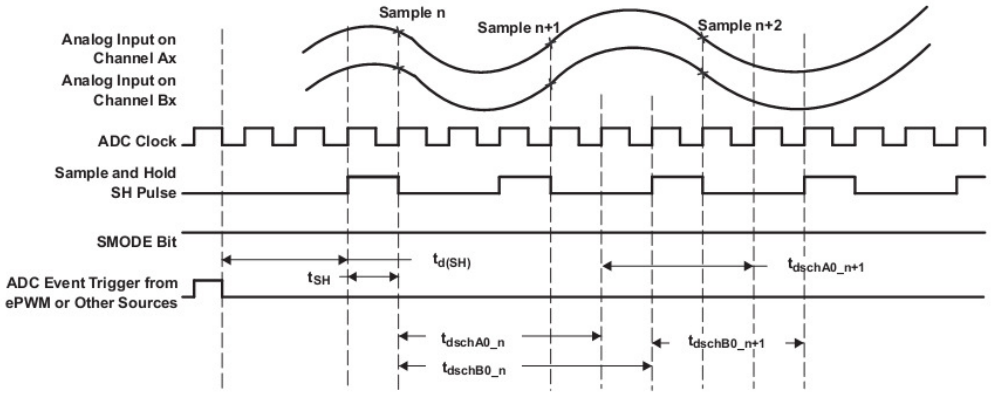


Figure 6-27. Simultaneous Sampling Mode Timing

Table 6-42. Simultaneous Sampling Mode Timing

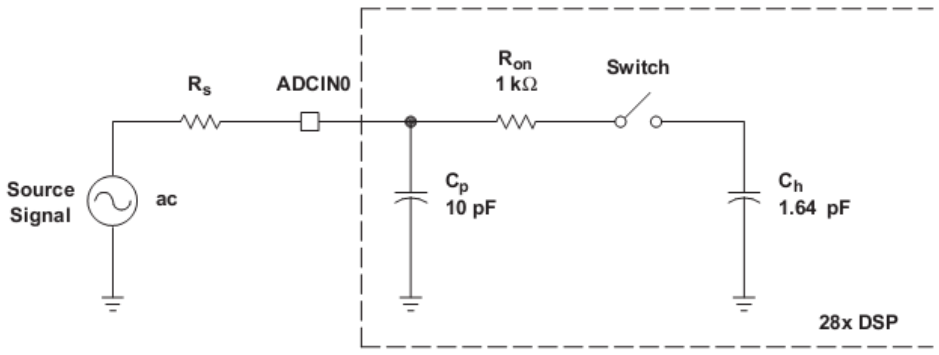
		SAMPLE n	SAMPLE n + 1	AT 12.5 MHz ADC CLOCK, $t_{c(ADCCLK)}$ = 80 ns	REMARKS
$t_{d(SH)}$	Delay time from event trigger to sampling	$2.5t_{c(ADCCLK)}$			
t_{SH}	Sample/Hold width/Acquisition Width	(1 + Acqps) * $t_{c(ADCCLK)}$		80 ns with Acqps = 0	Acqps value = 0–15 ADCTRL1[8:11]
$t_{d(schA0\_n)}$	Delay time for first result to appear in Result register	$4t_{c(ADCCLK)}$		320 ns	
$t_{d(schB0\_n)}$	Delay time for first result to appear in Result register	$5t_{c(ADCCLK)}$		400 ns	
$t_{d(schA0\_n+1)}$	Delay time for successive results to appear in Result register		(3 + Acqps) * $t_{c(ADCCLK)}$	240 ns	
$t_{d(schB0\_n+1)}$	Delay time for successive results to appear in Result register		(3 + Acqps) * $t_{c(ADCCLK)}$	240 ns	

Bild 29: Verzögerungen bei der ADC-Messung

Bei der Wahl der Fenstergröße sollte zum einen auf die Eingangsimpedanz der ADC-Einheit geachtet werden und zum anderen muss beachtet werden wie der maximal mögliche Strom- bzw Spannungsanstieg am ADC aussehen wird.

Das Ersatzschaltbild für den Eingang des ADC sieht man in Bild 30.Die Wahl der Fenstergröße spielt in diesem Anwendungsfall eine untergeordnete Rolle da bei der Beschaltung der ADC-Eingänge Kondensatoren vorgesehen wurden die die Ladung zur Aufladung des S/H-Kondensators schnell genug zur Verfügung stellen können.Eine maximale Messfehlerabschätzung lässt sich durch die Berechnung des maximalen Stromanstiegs durchführen. Die Maximalspannung des Systems beträgt Umax=50V und eine grobe Schätzung der Zeitkonstante und des Widerstandes liegen mit τ=1ms und R=0.2Ω vor. Daraus kann ein maximaler Stromanstieg von ΔI=157.5kA/s errechnet werden.

In dem berechneten Sampling-Fenster würde der Stromanstieg also 0.1 A betragen und somit einen maximal möglichen Messfehler (bei linearer Steigung) von 0.05A nach sich ziehen.



Typical Values of the Input Circuit Components:

Switch Resistance (R_{on}):	1 kΩ
Sampling Capacitor (C_h):	1.64 pF
Parasitic Capacitance (C_p):	10 pF
Source Resistance (R_s):	50 Ω

Bild 30: Eingangsimpedanzmodell für den ADC des TI F2808

- CAN-Einheit

Die CAN-Schnittstelle wird zur Übertragung der Messdaten aus der Steuerung verwendet. Diese Schnittstelle muss ebenfalls im Block *Target preferences* vorgenommen werden. Im Bild 31 sieht man die Grafik zum Bit-Timing des CAN-Busses. Standardeinstellungen für einen CAN-Bus mit 1Mhz bei einer System-Taktfrequenz von 100MHz sind:

- TSEG1=5
- TSEG2=4
- Baud Rate Prescaler=10
- Synchronisation Jump Width=2

Mit dem Parameter SAM kann noch angegeben werden wie oft am bzw. vor dem Sample Point abgetastet werden soll um den CAN-Level festzustellen.

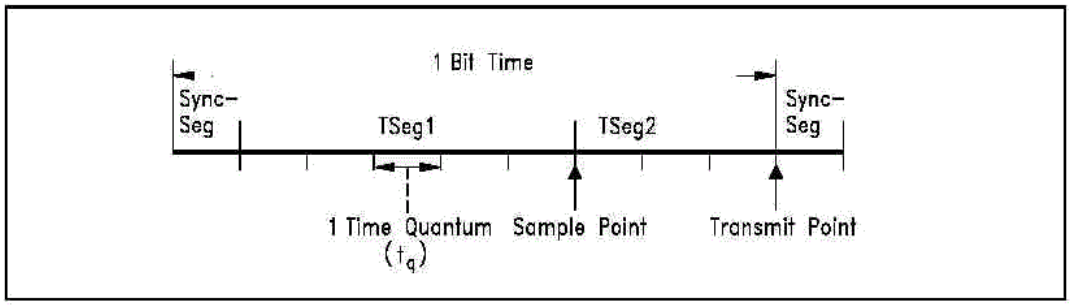


Bild 31:CAN-Timing

Wurde der CAN-Bus richtig konfiguriert kann man berechnen wie schnell man welche Daten über diese Schnittstelle auslesen kann:

Zum Senden einer Nachricht über den CAN-Bus wird die Nachricht an einen *eCAN Transmit*-Block übergeben, dieser ist auch in der C280x-Bilbliothek von Embedded Coder enthalten (Bild 32).

Berechnung des möglichen Datendurchsatzes:

Der CAN-Bus gilt im allgeimeinen bis zu 70% Buslast als echtzeitfähig.

Wie zuvor erwähnt wird in dieser Arbeit ein 1Mbit/s-CAN-Bus verwendet, das bedeutet als Berechnungsgrundlage für den Datendurchsatz wird 700kbit/s verwendet.

Eine CAN-Message mit 64bit Daten setzt sich aus eben diesen 64 Datenbits und 66 Steuerbits zusammen. 64 bit werden hier deshalb zur Berechnung genommen weil das die maximaale Datenpaketgröße für den CAN-Bus ist und somit den geringsten Overhead aufweist.

Zur Berechnung der maximalen Frequenz: Aus den vorhergegangenen Überlegungen folgt also:

$$f_{maximum} = \frac{\frac{700kbit}{s}}{2 * 130bit} = 2600\ Hz \tag{5.1}$$

Man kann also 128 bit Daten auf dem CAN-Bus mit 2,5kHz problemlos übermitteln.

- PWM-Einheit

Die PWM Einheit stellt verschiedene Funktionen zur Bildung von symmetrischer und unsymmetrischer PWM und zur Synchronisation mehrerer PWM-Ausgänge, sowie zur Einführung einer Totzeit an den PWM-Flanken zur Verfügung. Auch die Möglichkeit zur Synchronisation der PWM mit der Abtastung des ADC ist vorhanden.

Der Konfigurationsblock für die PWM findet sich wie alle weiteren Hardware-Blöcke im Embedded Coder Blockset in der Library C280x:

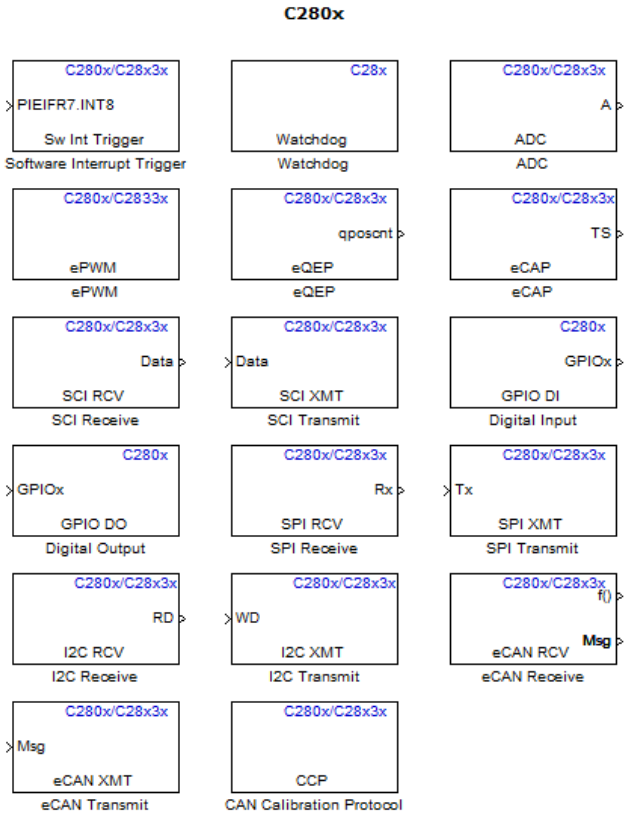


Bild 32: C280x Library im Embedded Coder

Da die PWM-Einheit des C280x sehr umfangreich ist wird hier nur auf die benötigten Einstellungen eingegangen:

- Module

Je nachdem welcher Prozessor in Verwendung ist muss hier das verwendete Hardware-Modul ausgewählt werden

- Timer period units und Timer period

Durch diese Einstellungen wird die Frequenz der PWM bestimmt!

Die Eingabe sollte in clock cycles erfolgen da eine andere Art der Eingabe Umrechnungsfehler und Berechnungszeit mit sich bringt.

Die Anzahl Clock cycles gibt einen maximalen Zählerstand an bevor dieser Zähler rückgesetzt wird. Die Frequenz mit der gezählt wird hängt von diversen Prescalern ab (TB, HSP, CLKINDIV, DIVSEL, PLLCR). In dieser Arbeit beträgt die Frequenz 100 MHz.

- Counting Mode

Hier kann man einstellen ob der Zähler sägezahnförmig zählen soll (resultiert in asymmetrischer PWM) oder hinauf und hinab zählen soll. Ausserdem lässt sich noch konfigurieren Ob der Ausgang nach Erreichen eines bestimmten Zählerwertes gesetzt oder rückgesetzt wird bzw. was beim Hinauf- und was beim Hinabzählen geschieht.

- Sync output

Hier kann eingestellt werden ob ein Synchronisationspuls für die nächste PWM-Einheit generiert werden soll und auf welches Ereignis (z.B.: Zähler gleich 0).

Falls mehrere PWM-Module synchronisiert werden kann der Sync-Puls von einer Einheit zur nächsten durchgeschleift werden.

- Deadband Unit

Stellt die Möglichkeit zur Verfügung vor bzw. nach einer Flanke des PWM-Signals eine Totzeit in Software einzufügen.

- Event Trigger

Hier kann man das SOC-Signal (Start of conversion) für den ADC Triggern.

Bei symmetrischer PWM ist es dabei sehr einfach z.B. gezielt genau in der Mitte des Pulses den ADC abzutasten.

6 Reglerentwurf

6.1 PI-Regler zur Offset-Bestimmung

Die Regelstrecke zur Offsetbestimmung ist nicht nur unbestimmt, selbst wenn man die Parameter für den Zusammenhang schätzen könnte wäre sie nichtlinear.

Nach der Drehmomentengleichung für die PMSM

$$M = \frac{3}{2} * p * \psi_{PM} * I_q \tag{6.1}$$

und der Bewegungsgleichung für Rotationskörper (Reibung vernachlässigt)

$$J * \ddot{\varphi} = M \tag{6.2}$$

kann man versuchen eine Gleichng für die nichtlineare Strecke aufzustellen.

Mit dem Differenzwinkel zwischen Rotorkoordinatensystem φ und Reglerausgangswinkel α

$$\gamma = \alpha - \varphi \tag{6.3}$$

Kann man Iq durch

$$I_q = I * \sin\gamma \tag{6.4}$$

ausdrücken.

Schreibt man jetzt

$$J * \dot{\omega} = \frac{3}{2} * p * \psi_{PM} * I * \sin\gamma \tag{6.5}$$

und wendet die LaPlace-Transformation an so erhält man:

$$\omega(s) = \frac{\frac{3}{2} * p * \psi_{PM} * I * \int_0^\infty \sin\gamma(t) * e^{-st}\, dt}{J * s} \tag{6.6}$$

Man kann Aufgrund der Sinus-Funktion bzw. der Nichtlinearität keine Übertragungsfunktion aufstellen. Stattdessen wird versucht einen stabilen Regler durch geeignete Randbedingungen zu erzwingen. Man sieht dass das Drehmoment proportional zu Iq ist, kennt man die Rotorlage nicht lässt sich jedoch hier auch schwer etwas Genaues annehmen.

Es ist bekannt dass das maximale Drehmoment bei einem fix eingeprägten Strom bei 90° elektrischem Versatz der Stator und Rotorkoordinatensysteme auftritt. Ohne Wissen über die Rotorlage ist jedoch auch diese Aussage ohne Wert.

Wie schon in den vorhergehenden Kapiteln gezeigt, wurde mit der Annahme begonnen: Sollte der Spannungsanstieg langsam genug sein sollte selbst ein relativ langsam ausgelegter Regler fähig sein den Encoderoffset ohne nennenswerte Rotorbewegung festzustellen.

Dies erweist sich jedoch als falsch sobald man sich auch das Reibmoment und den Stick-Slip-Effekt vor Augen führt.

Das bedeutet der Regler muss zumindest schon bei kleinem Strom bzw. kleinem Drehmoment eine große Verstärkung aufweisen um auf die sprunghaften Veränderungen reagieren zu können.

Aus dem gleichen Grund muss die Abtastzeit hier sehr niedrig gewählt werden.

Um den Offset auch sicher richtig festzustellen muss man die Maschine mit einem nennenswert großen Strom beaufschlagen, man muss damit das Reibmoment und das Rastmoment durch die Permanentmagnete überwinden.

Durch die Forderung nach hohem Strom und einem langsamen Stromanstieg wird die Strecke mit einer Veränderlichen Verstärkung beaufschlagt. Dieser Verstärkung muss man in irgendeiner Weise entgegenwirken.

In der vorliegenden Arbeit wurde entschieden die Gesamtverstärkung des PI-Reglers umgekehrt quadratisch proportional zur Stromstärke zu reduzieren.

Die Auslegung dieser Abschwächung wurde schon in Kapitel 4.4 aufgeführt.

Der Regler selbst ist in dieser Ausführung nur durch seinen I-Anteil zu spezifizieren, die Größenordnung der Verstärkung richtet sich nach der gewünschten Ausregelzeit nach einer einfachen Annahme: Will man eine Verdrehung von 90° in 0.1 Sekunde ausregeln so muss sich die Reglerausgangsgröße auch so schnell ändern können. Beispiele zur Auslegung werden in den Messergebnissen aufgezeigt.

Durch diesen Ansatz konnte eine sehr robuste Variante zur Reglerauslegung zur Offsetbestimmung vorgestellt werden.

Messergebnisse zu verschiedenen Reglerauslegungen werden ebenfalls in den Messergebnissen gezeigt.

Zusatzüberlegung zum Regler:

Der Regler arbeitet mit der Regelabweichung zwischen Drehzahl Null und der aktuellen Drehzahl.

Da der Winkel die integrierte Drehzahl ist ein I-Regler schon stabilisierend.

Würde man einen I-Regler mit der Verstärkung 1 als Regler wählen so würde, die Trägheit vernachlässigt) der Spannungszeiger immer den halben Weg des Rotors in die entgegengesetzte Richtung machen. Da aber die Trägheit im Spiel ist passiert auch das schon schneller in diesem Fall. Deshalb würde ein I-Regler mit der Verstärkung 1 in jedem Fall stabilisierend wirken und auch schnell ausregeln so lange die Abtastzeit hoch genug ist.

7 Mess- und Simulationsergebnisse

7.1 Offset-undWiderstandsbestimmung

7.1.1 Auslegung des Offsetreglers

Als grundlegender Forderung an den Offest-Regler kann wie schon unter 0 der Wunsch nach einer bestimmten Geschwindigkeit gestellt werden. Am Ausgang des Reglers wird auf -180° bzw. 180° Winkel begrenzt, auf dem DSP normiert stellt sich die Begrenzung im Q17-Format mit den Zahlenwerten -0.5 bzw. 0.5 dar.

Will man nun z.B. wie vorher angesprochen einen Versatz von 90° elektrisch innerhalb von 0.1 Sekunden ausgleichen muss der Regler das bewerkstelligen können. Dazu muss man eine Schätzung über die Winkelgeschwindigkeit zu Beginn des Versuchs anstellen. Zu diesem Zeitpunkt kann man mit dem eingeprägten Strom gerade das Reibmoment überwinden, kennt man dieses aus einem Datenblatt der Lager hat man hier schon Schätzwerte.

Eine sehr grobe aber zielführende Annahme ist: Die Drehzahl wird unter 60U/min bleiben. Man kann eine solche Annahme auch durch einen entsprechend langsamen Spannungsanstieg erzwingen.

Wir werden aber sehen dass dies nicht nötig ist.

Unter dieser Annahme müsste also bei einer Regelabweichung von 60U/min bzw. $\pi \frac{rad}{s}$ der I-Regler einen Ausgangswert von 0.25 in 0.1 sek erreichen.

Daraus ergibt sich Ki zu

$$K_i = \frac{2.5}{\pi} = 0.7958 \tag{7.1}$$

Auf den nächsten Seiten wird für verschiedene Werte von Ki das Ergebnis des Identifikationslaufs gezeigt.

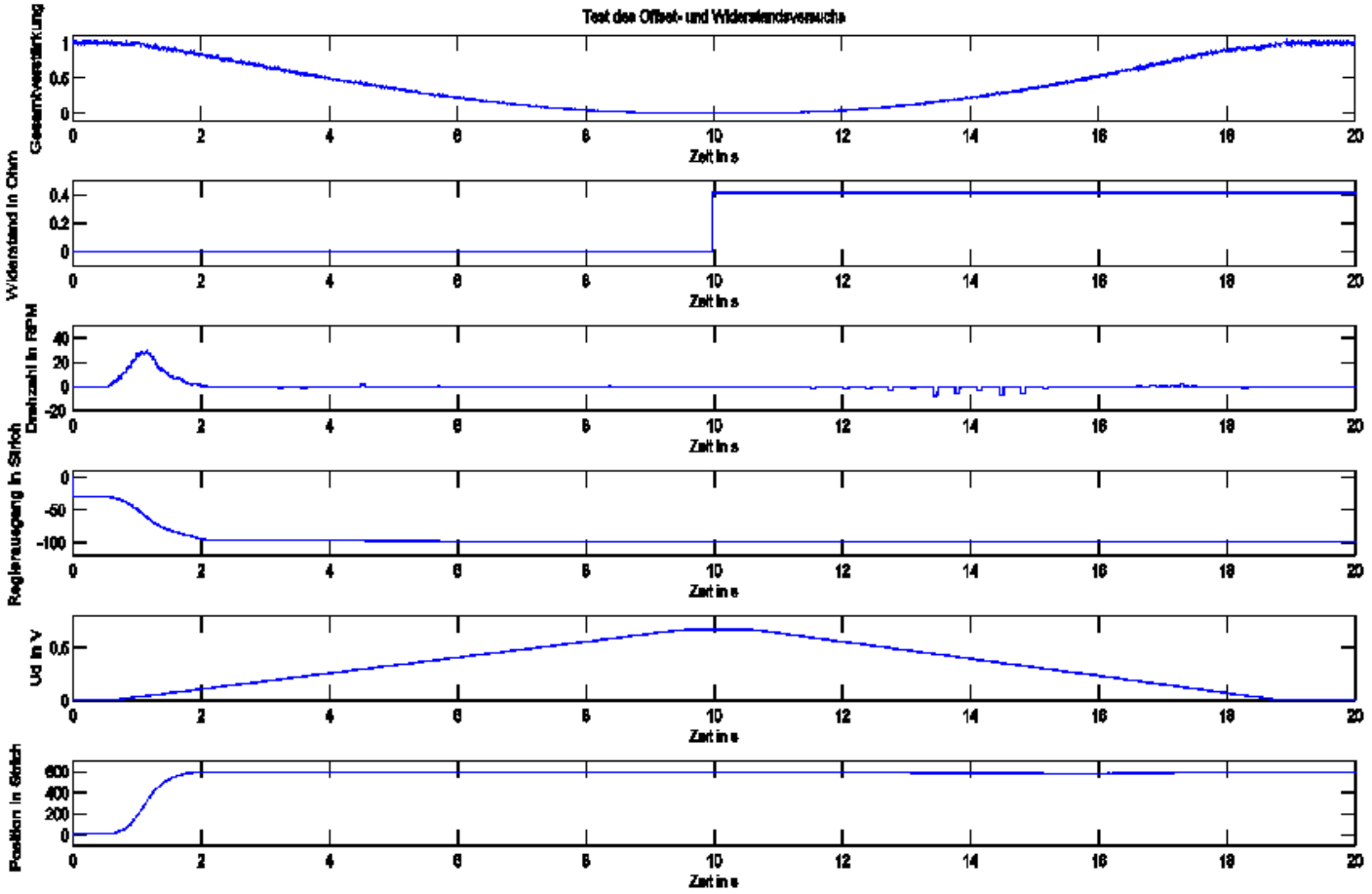


Bild 33: Offset- und Widerstandsbestimmung mit Ki=0.06

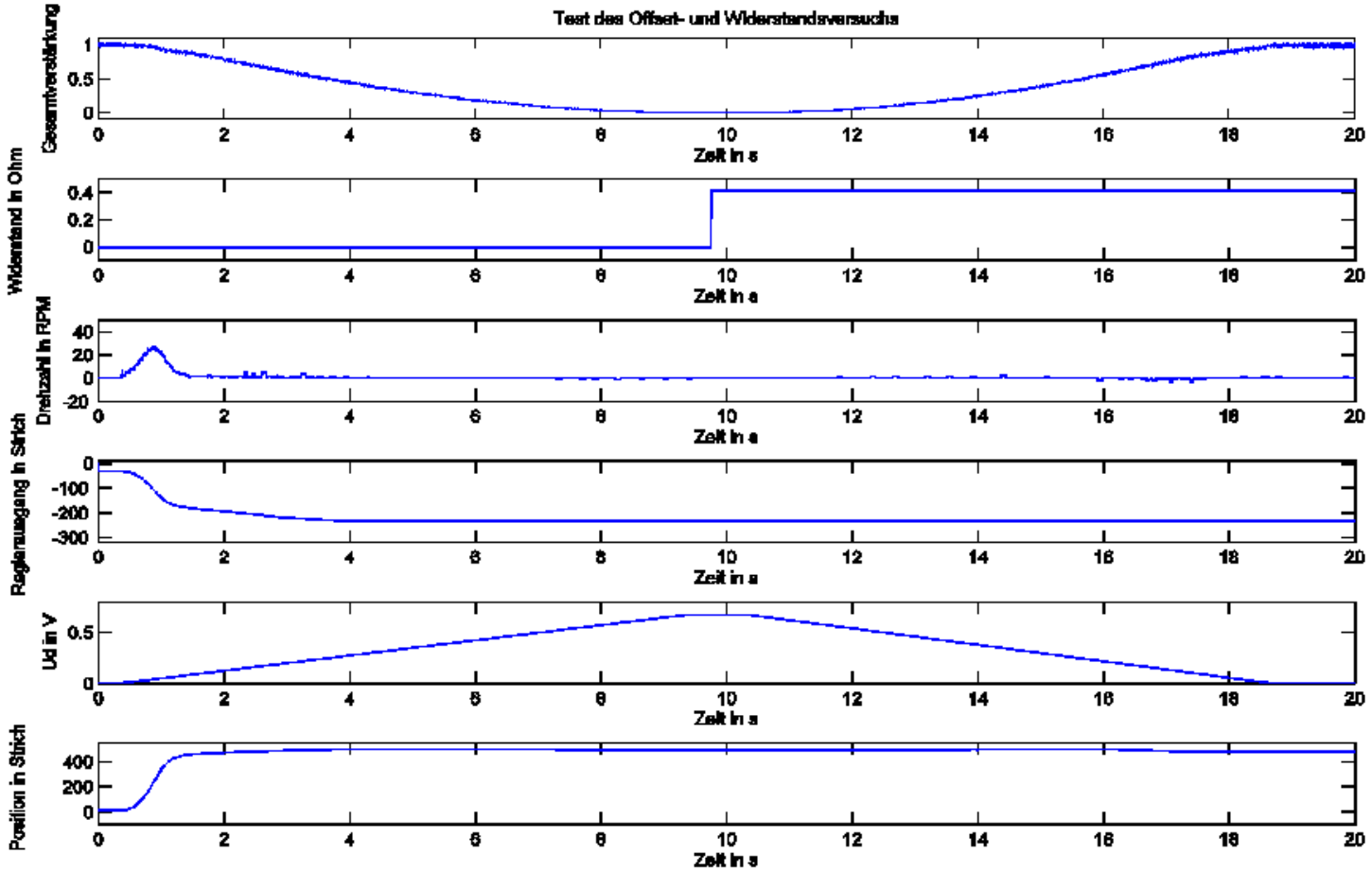


Bild 34: Offset- und Widerstandsbestimmung mit Ki=0.17

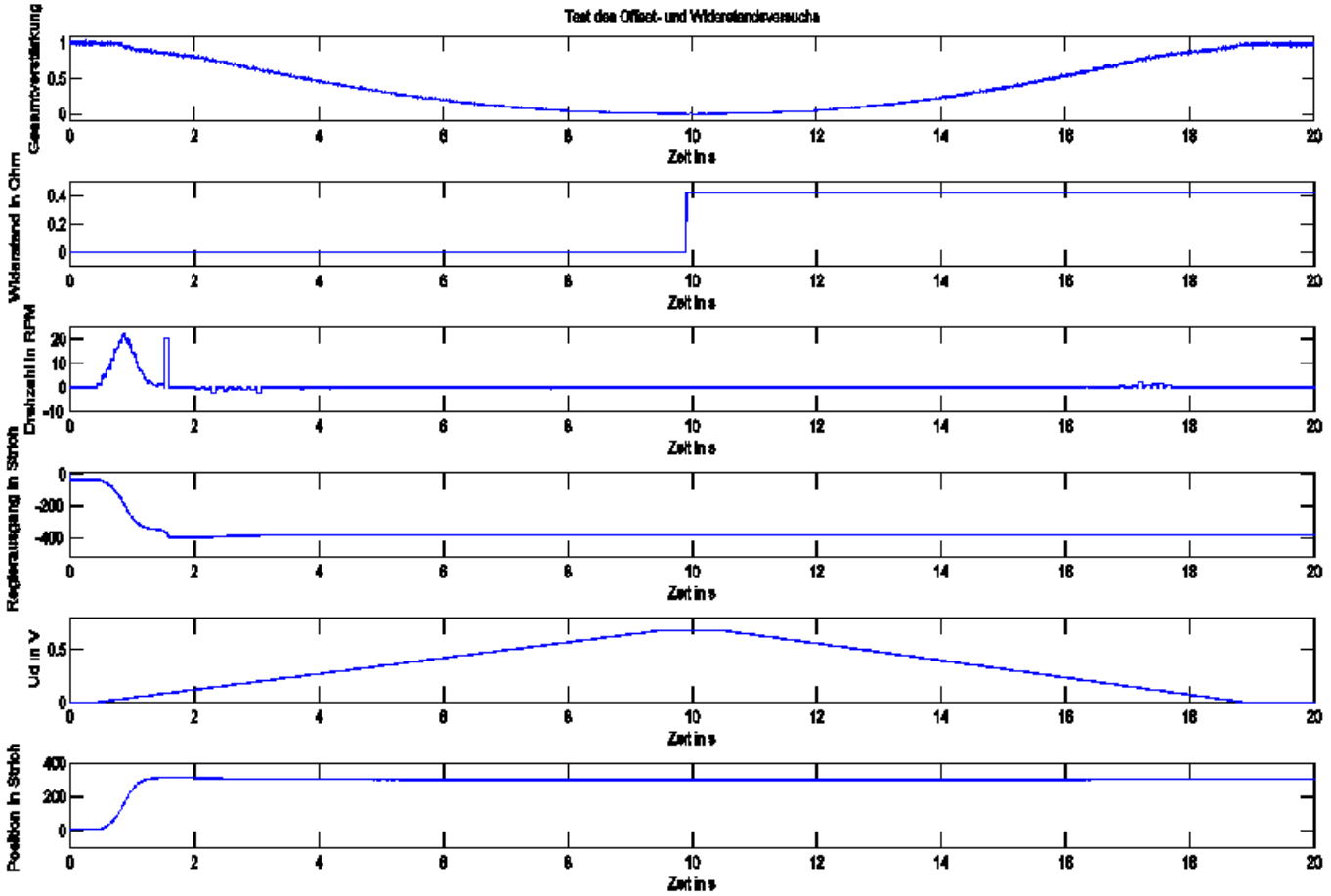


Bild 35: Offset- und Widerstandsbestimmung mit Ki=0.51

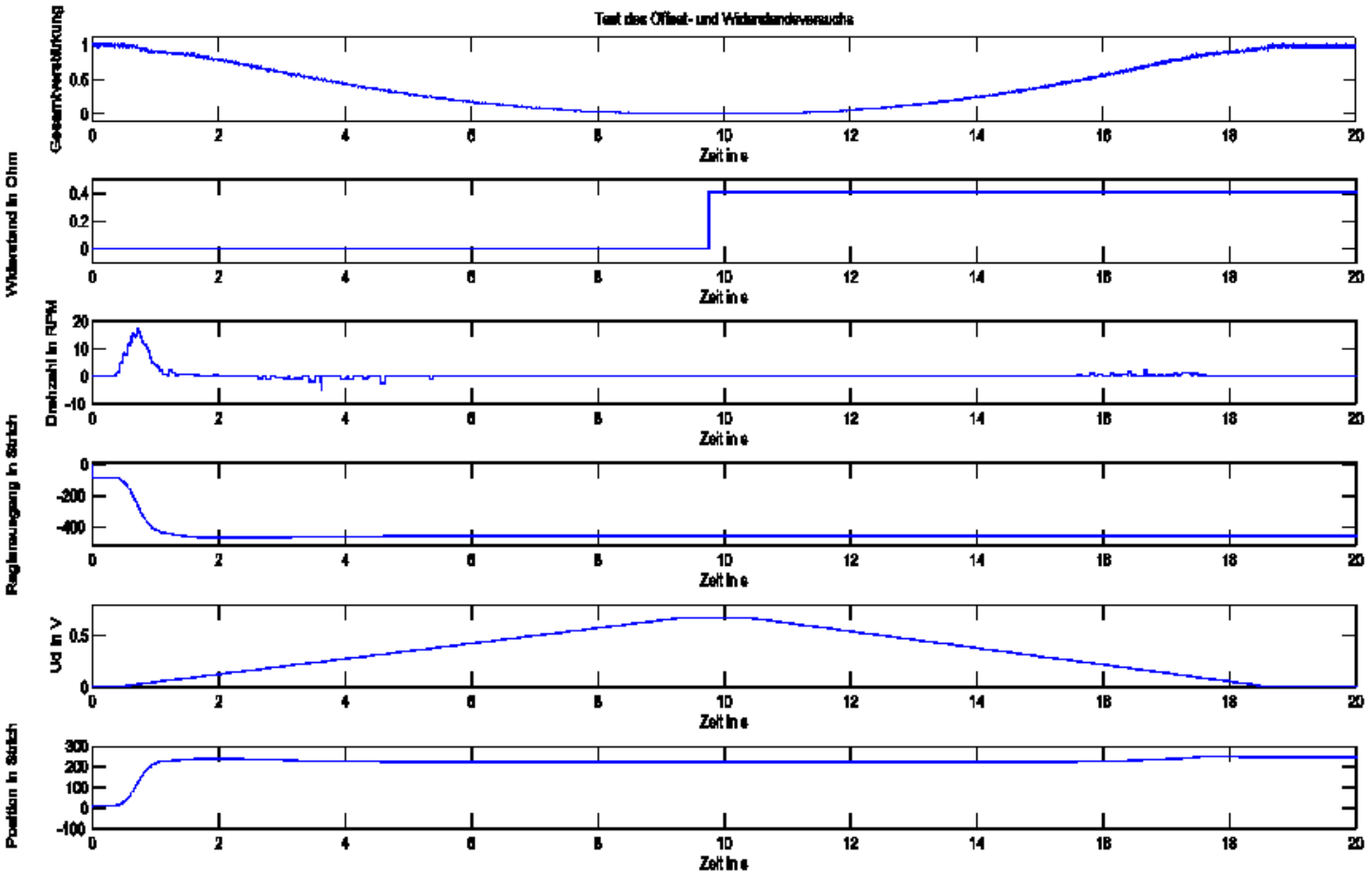


Bild 36: Offset- und Widerstandsbestimmung mit Ki=0.84

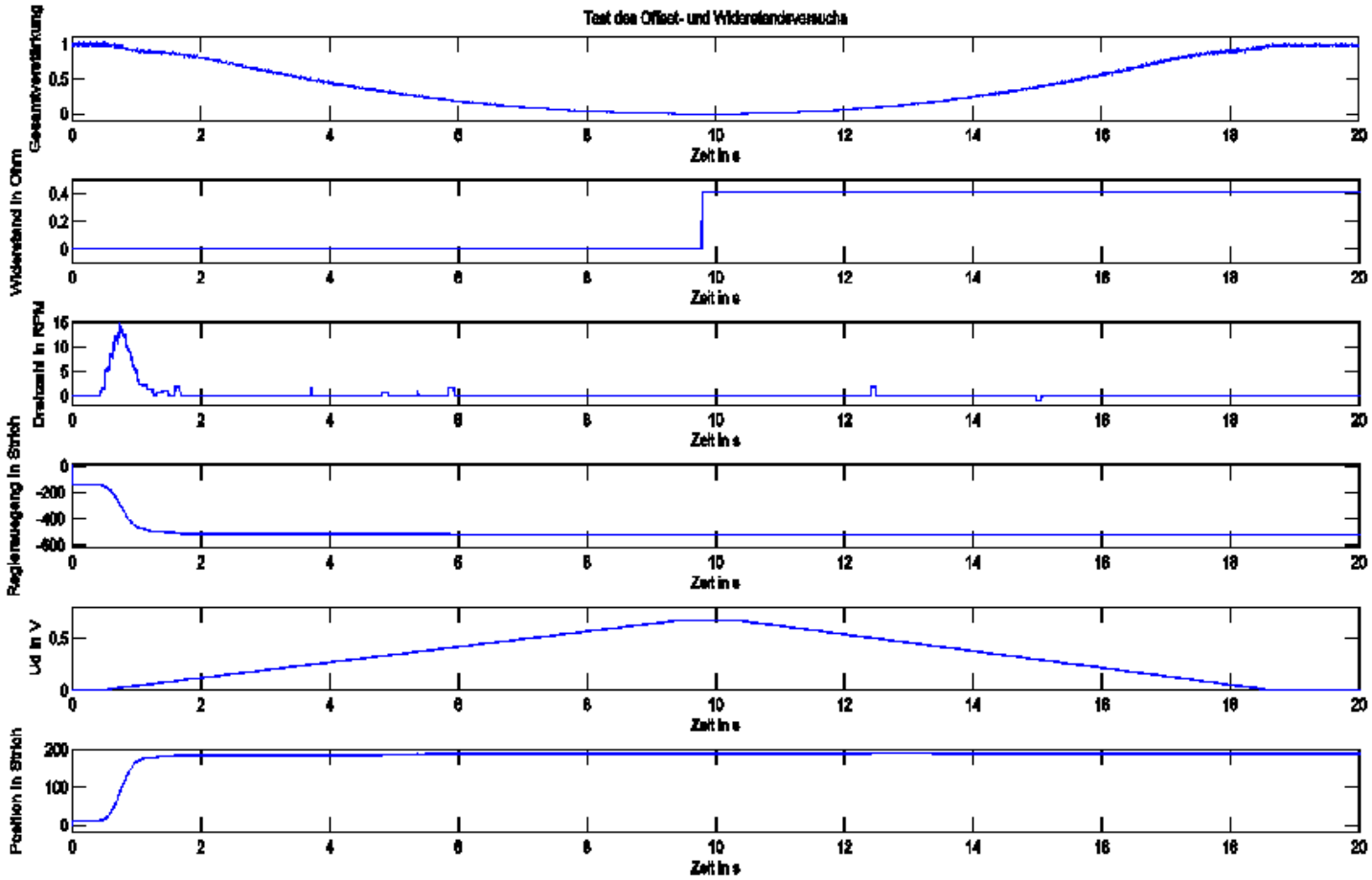


Bild 37: Offset- und Widerstandsbestimmung mit Ki=1

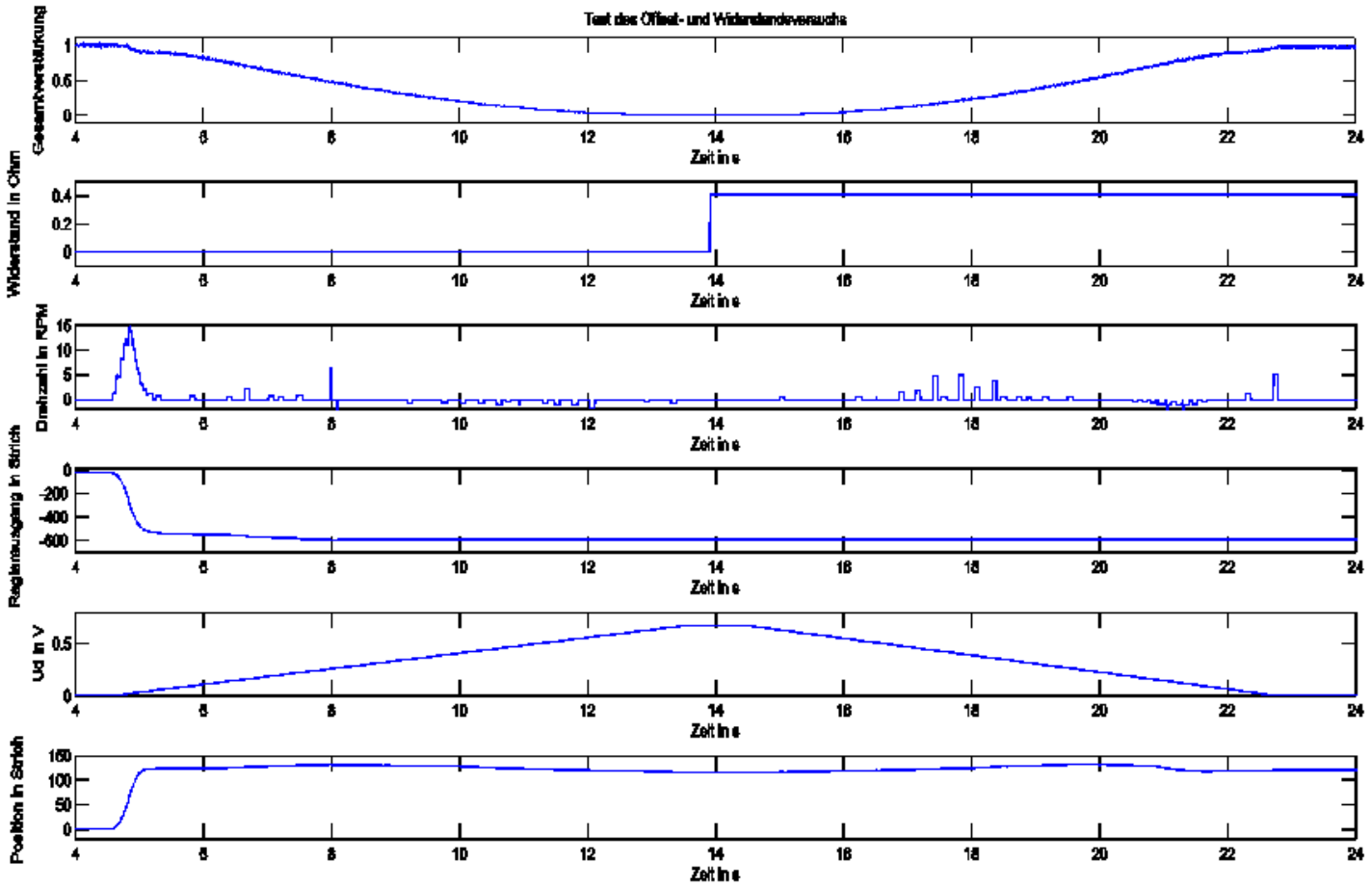


Bild 38: Offset- und Widerstandsbestimmung mit Ki=2

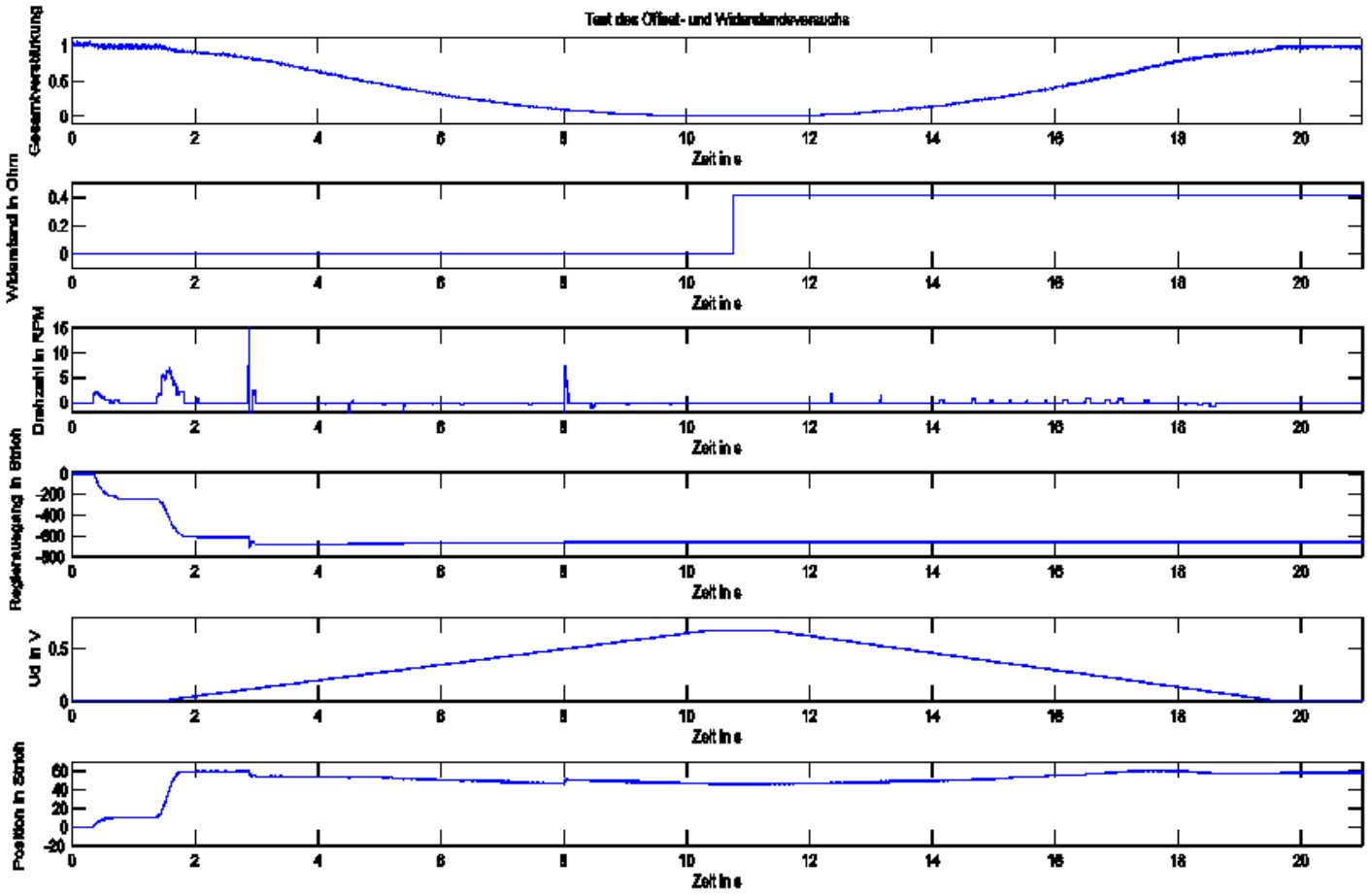


Bild 39: Offset- und Widerstandsbestimmung mit Ki=3.4

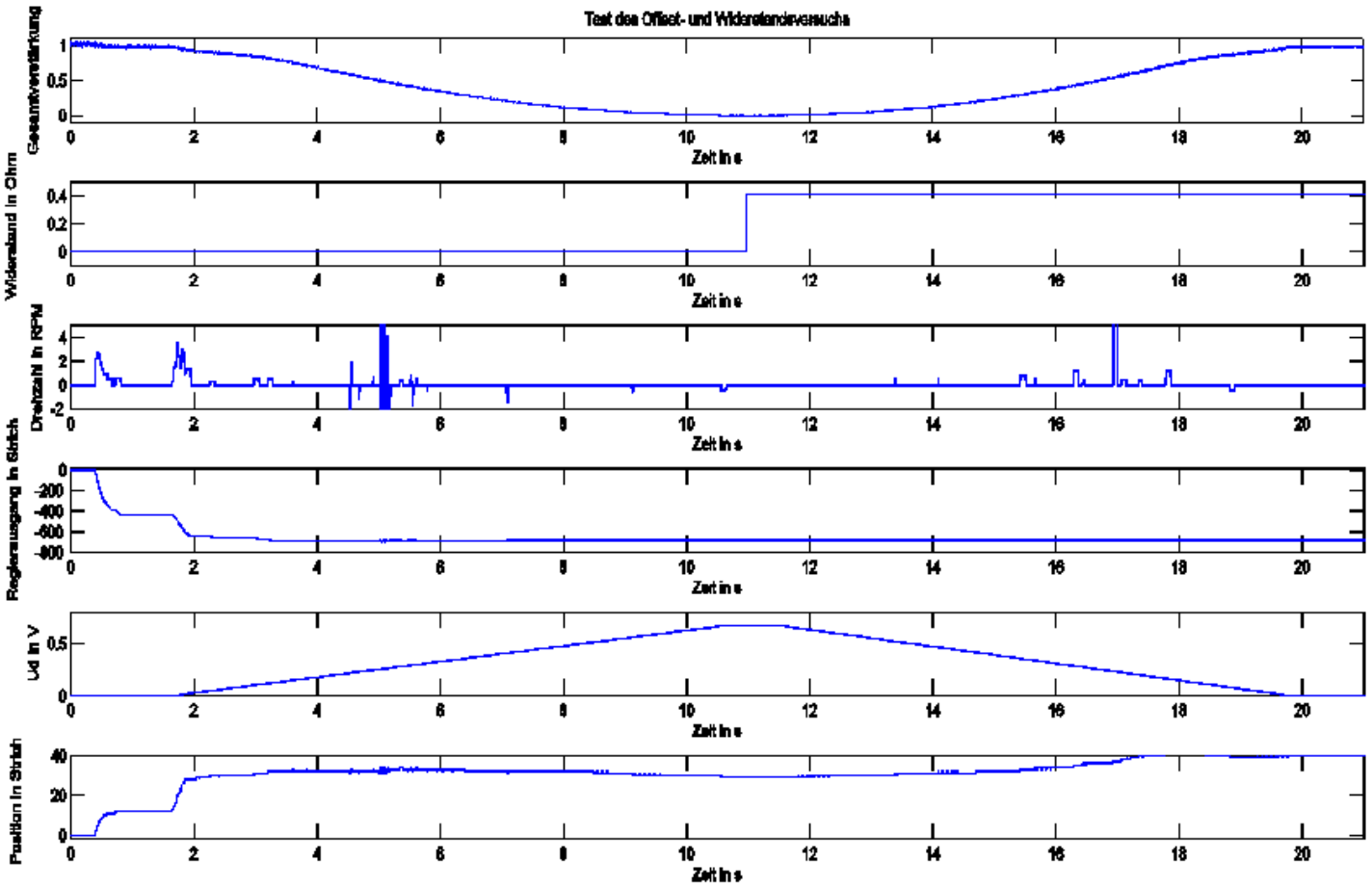


Bild 40: Offset- und Widerstandsbestimmung mit Ki=5.2

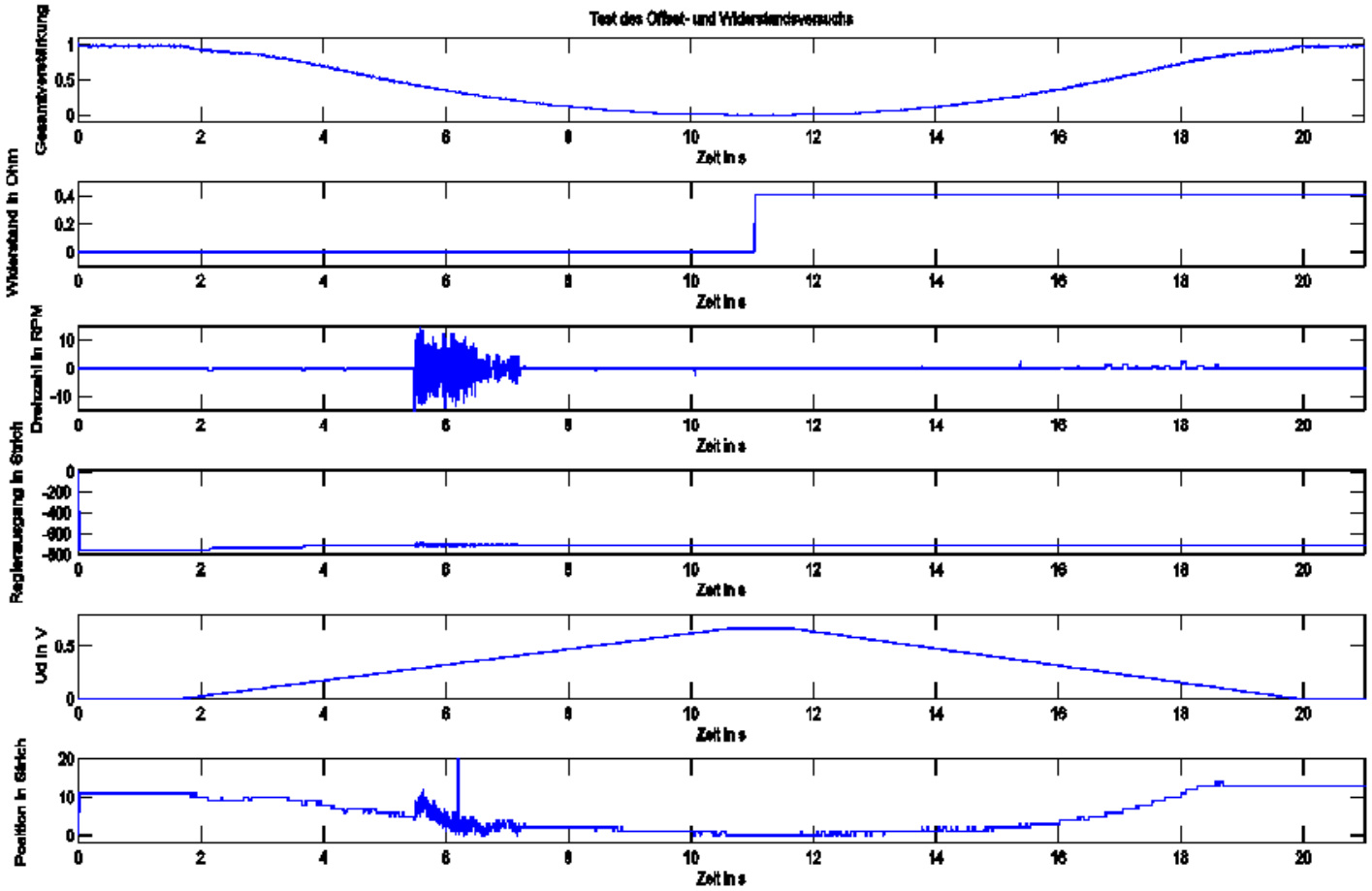


Bild 41: Offset- und Widerstandsbestimmung mit Ki=8

In den Bildern Bild 33 bis Bild 41 sieht man die Messergebnisse aus Versuchen mit Ki von 0.06 bis 8, in den Abbildungen ist jeweils im untersten Teil die Rotorpositionsveränderung zu sehen.

Man kann schlussfolgern dass mit höherer Verstärkung die Rotorbewegung natürlich geringer bleibt. Bei Ki=8 ist schon deutlich ein Schwingen des Reglers zu beobachten, dies wird jedoch von der quadratischen Abschwächung eingebremst bzw. unterdrückt. Es kommt erst ab Verstärkungen von Ki>15 zur Instabilität.

Man kann zusammenfassend sagen dass durch die quadratische Abschwächung eine sehr robuste Einstellung des Reglers möglich ist.

Eine mögliche Verbesserungsvariante des Testlaufs liegt zum Beispiel darin die Spannungsrampe beim ersten Auftreten einer Bewegung langsamer steigen zu lassen.

Das wurde auch durch empirische Versuche dazu festgestellt.

Im Bild 42 sieht man zur besseren Vergleichbarkeit noch einmal die verschiedenen erreichten Rotorbewegungen bei den unterschiedlichen Reglereinstellungen zusammengefasst.

In Bild 42 sieht man das Ergebnis der Optimierungsregelschleife zur Induktivitätsbestimmung.

Parameter	Datenblatt	Identifikation
R in Ohm	0.345	0.396
L in mH	0.273	0.206

Tabelle 2: Vergleich der Messwerte mit dem Datenblatt

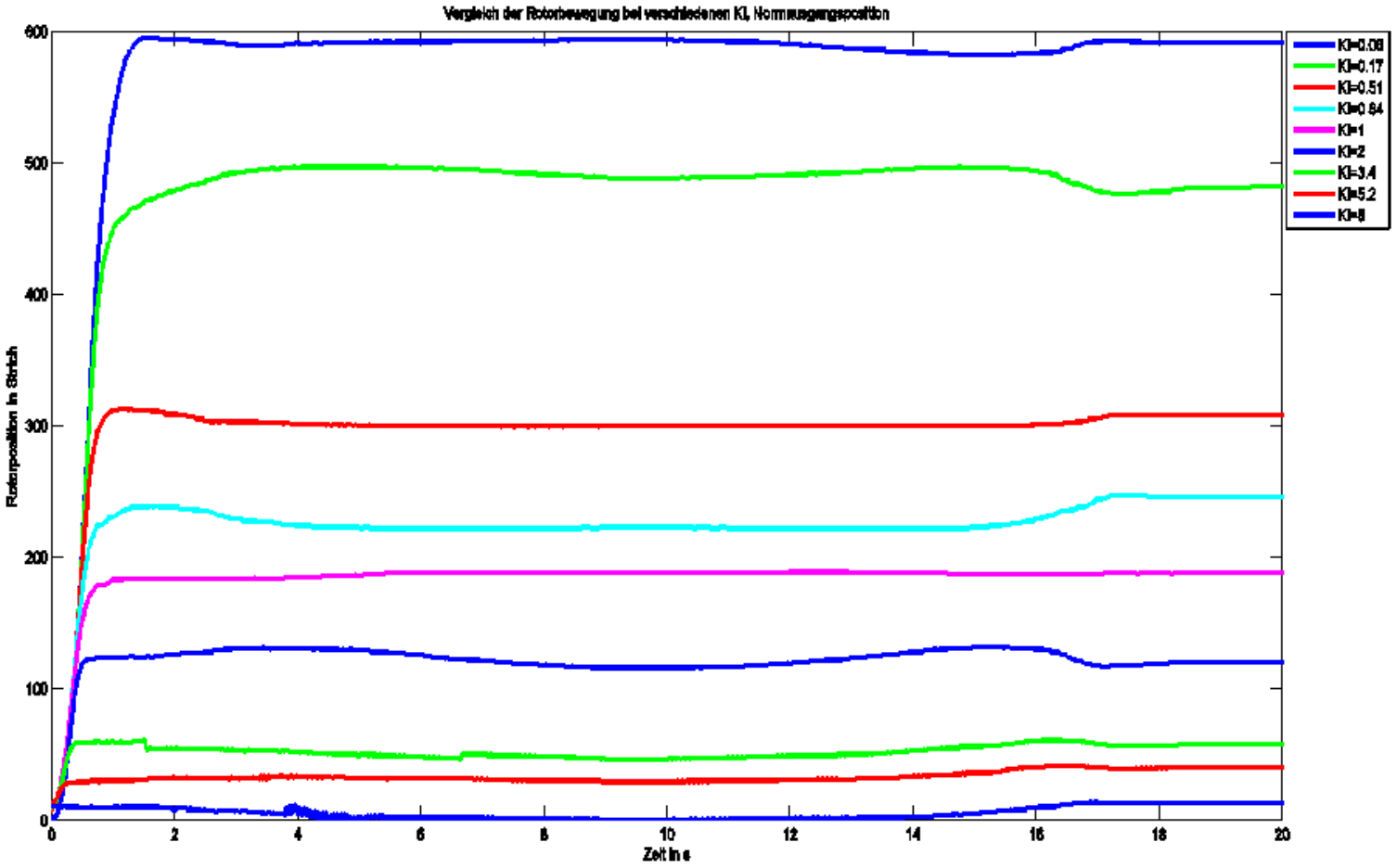


Bild 42: Vergleich der Rotorpositionsbewegung zu den gezeigten Reglern

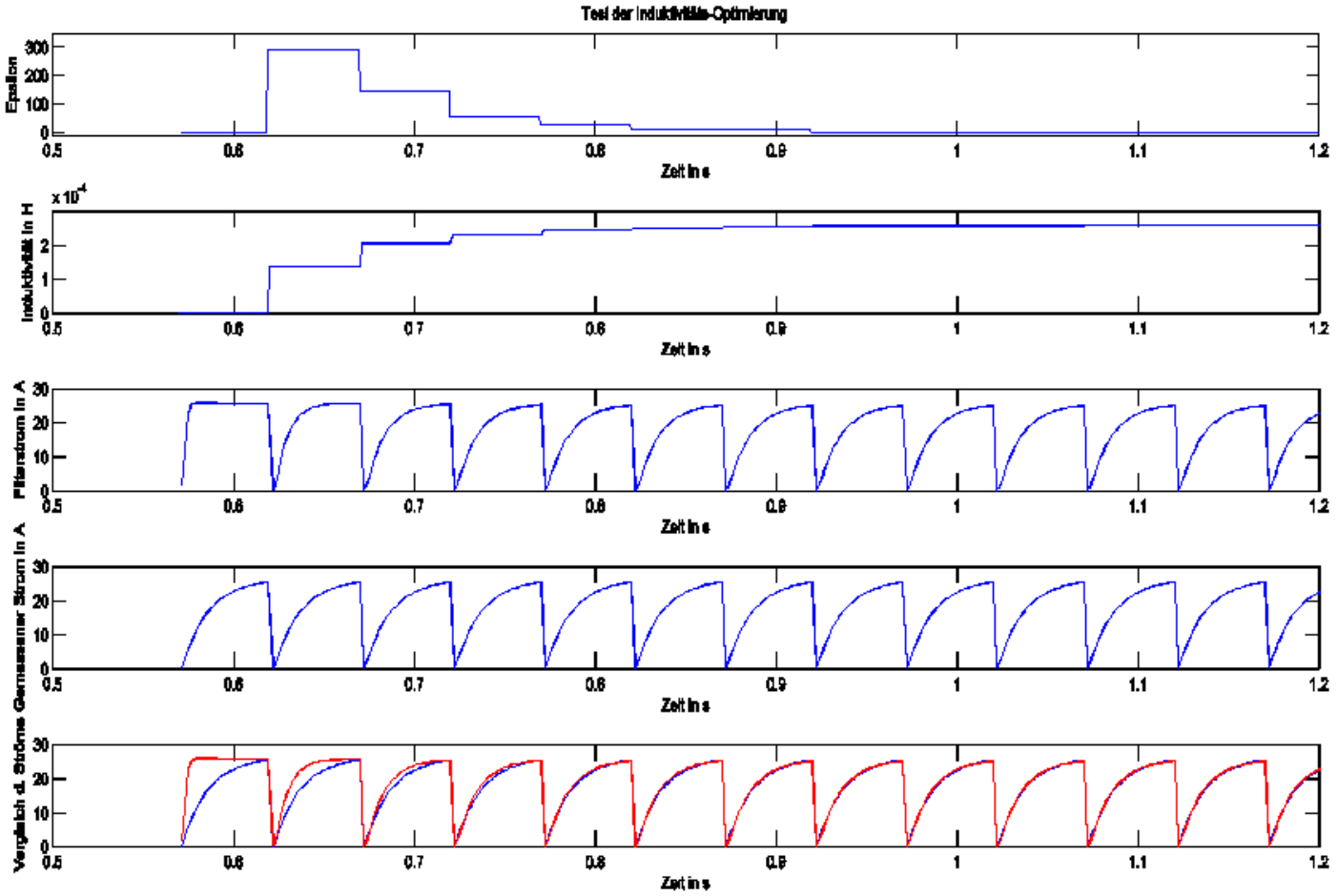


Bild 43: Ergebnis des Induktivitäts-Optimierungsversuchs

Literaturverzeichnis

[1] **Dierk Schröder:** *Elektrische Antriebe - Grundlagen*, Aachen, Springer-Verlag Berlin Heidelberg 1994, 2000, 2007, 2009

[2] **Dumitru Daniel Popa, Liviu Mario Kreindler, Raducu Giuclea, Aurelian Sarca:** *A novel method for PM synchronous machine rotor position detection*, CH-2022 Bevaix, Switzerland, Polytechnic University of Bucharest, Romania, 2007

[3] **A. O`Dwyer:** *PI and PID controller tuning rules for time delay processes: a summary*, Technical report A0D-00-01, Edition 1, School of Control Systems and Electrical Engineering, Dublin Institute of Technology Dublin 8, Ireland, 2000

[4] **Bassel Sahhary:** *Elektrische Antriebe mit sauermagneterregten Maschinen im dynamischen sensorlosen Betrieb*, Fakultät für Elektrotechnik, Helmut-Schmidt Universität, Hamburg, 2008

[5] **Dierk Schröder:** *Leistungselektronische Schaltungen, Funktion, Auslegung und Anwendung*, Springer-Verlag Berlin Heidelberg, 2008

[6] **Héctor D. Perassi:** *Feldorientierte Regelung der permanenterregten Synchronmaschine ohne Lagegeber für den gesamten Drehzahlbereich bis zum Stillstand*, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität Ilmenau, 2007

[7] **Texas Instruments:** *TMS320x280x DSP, System Control and Interrupts Reference Guide*, Literature Number: SPRU712C, 2004, 2006

[8] **Texas Instruments**: *TMS320x280x Enhanced Quadrature, Encoder Pulse (eQEP) Module Reference Guide*, Literature Number: SPRU790A, 2004, 2006

[9] **Texas Instruments**: *TMS320x280x DSP: Analog-to-Digital Converter (ADC) Reference Guide,* Literature Number: SPRU716B, 2004, 2005

[10] **Texas Instruments:** *TMS320x281x, 280x DSP Peripheral Reference Guide Reference Guide, Literature Number*: SPRU566C, 2003, 2006