Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Felix Kolbe

Redundanzkonzept eines Time-Triggered Bussystems in einem Formula Student Rennwagen: Modellierung, Implementierung und Anwendung in der Antriebsschlupfregelung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Felix Kolbe

Redundanzkonzept eines Time-Triggered Bussystems in einem Formula Student Rennwagen: Modellierung, Implementierung und Anwendung in der Antriebsschlupfregelung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Franz Korf
Zweitgutachter : Prof. Dr. rer. nat. Stephan Pareigis

Abgegeben am 22. August 2008

Felix Kolbe

Thema der Bachelorarbeit

Redundanzkonzept eines Time-Triggered Bussystems in einem Formula Student Rennwagen: Modellierung, Implementierung und Anwendung in der Antriebsschlupfregelung

Stichworte

Redundanz, Verlässlichkeit, Sicherheit; TimeMaster, zeitgesteuert, Echtzeit; TTCAN, verteilte Knoten; Antriebsschlupfregelung, Traktionskontrolle, Fahrassistenzsystem; Telemetrie, Formula Student, Hawks Racing Team

Kurzfassung

Diese Bachelorarbeit beschäftigt sich mit der Verbesserung der Zuverlässlichkeit eines Time-Triggered CAN-Busses durch den Ansatz redundanter TimeMaster. Dazu wird das Verhalten der TimeMaster modelliert und in das bisherige System integriert. Dabei werden die Kompatibilität und die zeitlichen Anforderungen in einem Rennwagen beachtet. Der Ausfall von Komponenten wird analysiert und die Stabilität des Systems bewiesen. Auf Basis dieses verlässlicheren Bussystems wird eine Antriebsschlupfregelung implementiert, exemplarisch für eine busbasierte Steuerungsaufgabe.

Felix Kolbe

Title of the paper

Redundancy concept of a time triggered bus system in a Formula Student racing car: modelling, implementation and application in a traction control system

Keywords

Redundancy, reliability, safety; time master, time triggered, real time; ttcan, distributed nodes; anti-slip regulation, traction control system, driver assistance system; telemetry, Formula Student, Hawks Racing Team

Abstract

This thesis concentrates on the reliability improvement of a time triggered CAN bus, by means of implementing redundant time masters. Their behaviour is modelled and integrated into the existing system. Thereby compatibility and timing requirements are considered. Malfunction of components is analysed and system stability is proved. On the basis of this more reliable bus system a traction control is implemented, exemplarily for a bus based control task.

Inhaltsverzeichnis

1 Einführung

In den heutigen Automobilen befinden sich immer mehr elektronische Komponenten. Diese ersetzen ihre mechanischen Vorgänger und sparen so Raum, Energie und Gewicht, oder sie ermöglichen neue Funktionen. Einen Großteil dieser Funktionen bilden Fahrassistenzsysteme (FAS), die das Sicherheitsrisiko des Fahrzeugs, der Insassen und auch der Umgebung verringern. Die bekanntesten Systeme sind das Antiblockiersystem (ABS) und das Elektronische Stabilitätssystem (ESP).

Abbildung 1.1: Der Rennwagen Hawk08 in Hockenheim, 2008
(Hanselmann, Formula Student Germany)

Auch für einen Formula Student Rennwagen sind solche Fahrassistenzsysteme interessant: Beispielsweise spart eine Hochschaltautomatik für eine elektronisch gesteuerte Gangschaltung entscheidende Zehntelsekunden im Beschleunigungsrennen. Durch eine Antriebsschlupfregelung (ASR) kann noch mehr Leistung auf die Straße gebracht werden, und dem Fahrer wird in kritischen Situationen assistiert.

Damit der Fahrer sich auf das Fahrassistenzsystem verlassen kann, muss dieses außerst zuverlässig sein. Die Anforderungen Echtzeitfähigkeit und Ausfallsicherheit müssen unbe-

dingt erfüllt werden. Diese Arbeit beschäftigt sich mit der Erfüllung dieser Anforderungen für den Rennwagen des Hawks Racing Teams der Hochschule für Angewandte Wissenschaften Hamburg. Das Ziel ist, Fahrassistenzsysteme in diesen Rennwagen zu integrieren.

Die Module der Fahrassistenzsysteme werden untereinander und mit den Sensoren und Aktoren über Bussysteme verbunden. Dabei wird nicht jedes Modul mit jedem anderen direkt verbunden, sondern alle Module werden an eine Leitung angeschlossen. Die entstehende Verkabelung ist im Gegensatz zur direkten Verbindung übersichtlich und skalierbar, wodurch sich Aufwand, Geld und Gewicht sparen lassen. Als Herausforderung hierbei gilt die Zuverlässigkeit des Busses: Ist die Kommunikation gestört, können die Dienste ihre Funktion nicht erfüllen.

Damit diese Bussysteme eine solche verantwortungsvolle Aufgabe zuverlässig übernehmen können, werden zunehmend zeitgesteuerte Varianten eingesetzt, wie z. B. TTCAN[1], Flexray[2] und TTP[3]. Bei zeitgesteuerten Systemen wird im Gegensatz zu herkömmlichen ereignisgesteuerten Systemen die gemeinsame Zeit als Steuersignal für die Zuteilung des Busses genutzt. Es wird im Voraus festgelegt, welche Komponente zu welchem Zeitpunkt und mit welcher Dauer den Bus aktiv benutzen und Nachrichten versenden darf. Dazu wird eine sich wiederholende Folge von Zeitfenstern definiert. Die statische Zuweisung dieser Zeitfenster ergibt eine deterministische Auslastung und Antwortzeit des Busses.

In einer solchen Variante jedoch steht und fällt die Funktion mit dem Zustand des Zeitgebermoduls, TimeMaster genannt. Um diesem Single Point of Failure[4] vorzubeugen, werden mehrere dieser Zeitgebermodule redundant eingesetzt und als Reserven programmiert. Diese werden aktiv, wenn das zuvor aktive Modul außer Betrieb geht.

In dieser Arbeit wird dieses Konzept redundanter Zeitgeber in das bestehende Bussystem des studentischen Rennwagens integriert. Dadurch soll dieses Bussystem nicht nur passiven und unkritischen, sondern auch aktiven und sicherheitsrelevanten Diensten als Basis dienen können. So können zukünftig Signalleitungen, die bisher neben dem Bus verliefen, durch Busnachrichten ersetzt werden. Dies verringert das Gewicht und den Verkabelungsaufwand.

Um zu demonstrieren, dass sicherheitsrelevante Anwendungen von dem Bussystem verlässlich übernommen werden können, wird ein Fahrassistenzsystem im Rennwagen integriert. Da hierbei die Interessen des Rennteams die Möglichkeiten eingrenzen, wird hier die Antriebsschlupfregelung als zu implementierendes Fahrassistenzsystem gewählt. Wie bereits erwähnt ist dieses System für einen Rennwagen durchaus interessant und lässt sich verhältnismäßig einfach integrieren.

[1] TTCAN: Time Triggered Controller Area Network, http://www.ttcan.com
[2] Flexray: http://www.flexray.com
[3] TTP: Time-Triggered Protocol, http://www.tttech.com
[4] Single Point of Failure (SPOF): Der Ausfall einer Komponente bewirkt den Ausfall des Gesamtsystems

1.1 Das Hawks Racing Team und die Formula Student

Das Hawks Racing Team ist eine Gruppe von Studenten der Hochschule für Angewandte Wissenschaften Hamburg, die an der Fomula Student[5] teilnimmt. Es entwirft und baut den Rennwagen, in den die Entwicklung dieser Arbeit integriert wird.

Abbildung 1.2: Formula Student Germany und Hawks Racing Team Logos

1.1.1 Die Formula Student

Die Formula Student ist ein Projekt für Studenten der Ingenieurswissenschaften zur Ergänzung des Studiums, um Erfahrung in Produktion und Marketing zu sammeln. Das Ziel ist es, einen einsitzigen Rennwagen zu entwickeln, zu konstruieren und einen Prototypen zu fertigen. Dabei sind gewisse Regeln einzuhalten um die Sicherheit zu erhalten. Hypothetisch soll der Wagen einmal tausendfach pro Jahr produziert, und mit einem Stückpreis von maximal 25.000 $ verkauft werden.

In vielen Ländern auf mehreren Kontinenten gibt es bereits jährliche Wettbewerbe, an denen Teams vieler Hochschulen und Universitäten teilnehmen. Diese müssen neben ihrem Rennwagen auch technische Dokumentationen und einen Finanzierungsplan vorlegen, als gäbe es einen Investor für ihren Prototypen.

1.1.2 Das Hawks Racing Team

Das Hawks Racing Team wurde 2003 gegründet und besteht momentan aus etwa 60 Studenten. Das Team hat bereits 3 Rennwagen gebaut und war mit diesen mehrfach erfolgreich. Es nimmt üblicherweise an den Wettbewerben in Hockenheim (Deutschland), Silverstone (England) und Fiorano (Italien) teil. Parallel zur Entstehung dieser Arbeit wird mit dem Hawk08 der vierte Wagen gefertigt.

[5] http://www.formulastudent.de & http://www.formulastudent.com

1.2 Das bestehende Telemetriesystem

Für den Hawk07, den Rennwagen des Vorjahres, gibt es bereits ein Telemetriesystem. Es wurde in zwei Bachelorarbeiten von Haase (2007) und Schuckert (2007) entwickelt und integriert. Da das System auf Erweiterbarkeit und Flexibilität ausgelegt wurde, ist es problemlos möglich, die bisherigen Komponenten weiterzuverwenden und neue hinzuzufügen. Bestehende Komponenten, die funktional von neuen Komponenten unabhängig sind, müssen nicht angepasst oder neu entwickelt werden.

1.2.1 Der Aufbau

Das System besteht im Wesentlichen aus verteilten Knoten auf Basis des AVR Mikrocontrollers AT90CAN128 (siehe Abschnitt 2.5), die über einen CAN-Bus miteinander verbunden sind. Jeder Knoten arbeitet weitgehend unabhängig von den anderen und ist nur für eine bestimme Teilaufgabe zuständig. Das Gegenprinzip wäre ein entsprechend größerer Hauptrechner, der das Auslesen der Sensoren, das Auswerten der Daten und alle weiteren Funktionen übernimmt.

Durch den CAN-Bus besteht die Möglichkeit, die Knoten unabhängig voneinander im Fahrzeug zu platzieren, um z. B. den Mikrocontroller nahe an den jeweiligen Sensor zu setzen. Je kürzer die Signalleitung zum Sensor ist, und an je weniger Störquellen die Leitung vorbeiläuft, etwa der Zündelektronik des Motors, desto fehlerfreier ist das erhaltene Signal.

Dies ist besonders wichtig, wenn es sich um ein analoges Signal handelt. Wenn nur Strom- und CAN-Leitungen durch das gesamte Fahrzeug gelegt werden, statt alle Sensorleitungen zu einem Hauptrechner zu führen, wird außerdem Gewicht gespart, ein wichtiges Kriterium bei einem Rennwagen.

Die Aufteilung in physikalisch kleine Knoten macht eine akzeptable Integration in das Fahrzeug erst möglich: Ein Rennwagen dieser Größe besitzt innerhalb des Rahmengerüstes nicht ohne weiteres einen großen ungenutzten Raum, in den sich ein kleiner Hauptrechner einsetzen ließe. Die kleinen Module dagegen, im Hawk07 sind es 7 an der Zahl, lassen sich in jedem beliebigen Freiraum unterbringen.

Folgenden Schlussgedanken hat Schuckert (2007) für die Wahl der Systemarchitektur verfasst:

> „Die Wahl fällt auf das System ohne Hauptrechner. Die erhöhte Ausfallsicherheit sowie Flexibilität, überzeugen in diesem Fall. Zudem besteht die Möglichkeit, alle Module nun mit dem gleichen Mikrocontroller auszustatten, was evtl. eine einheitliche Softwarearchitektur zulässt“ (Schuckert, 2007, S. 10).

Die nachstehende Abbildung 1.3 veranschaulicht die logische Verbindung der Komponenten. Die AVR-Module sind jeweils mit den Sensoren direkt und untereinander über den CAN-Bus verbunden.

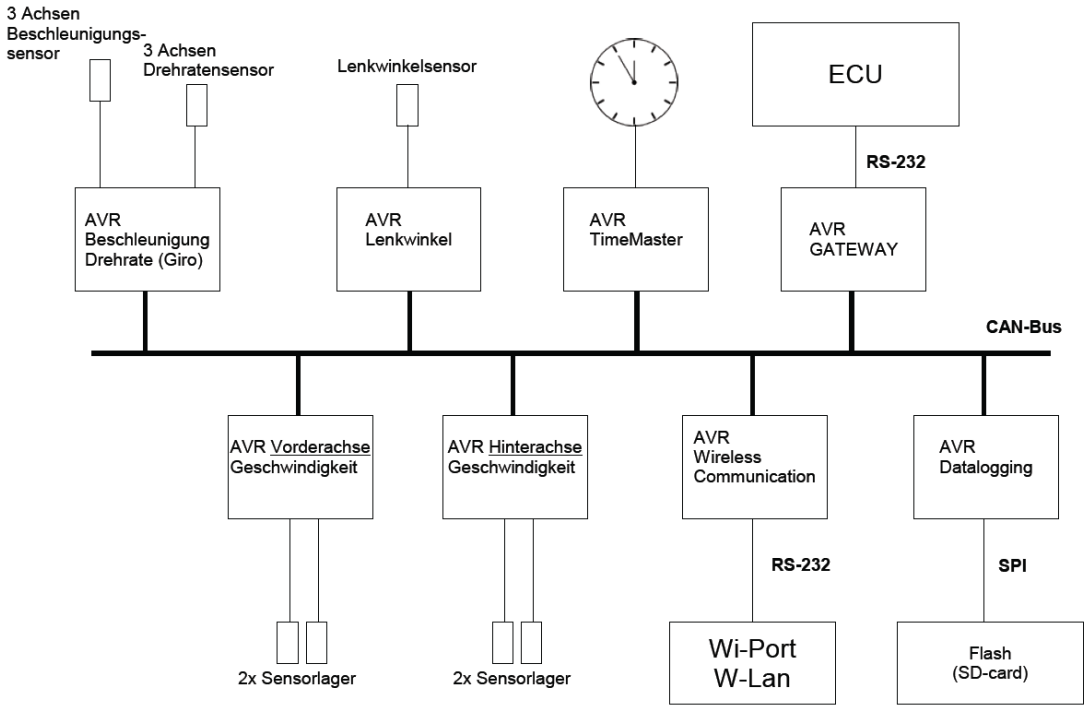

Abbildung 1.3: Systementwurf von Schuckert (2007) für den Hawk07

1.2.2 Die Funktionen und Möglichkeiten

Im Hawk07 wurden folgende Werte über eingebaute Sensoren ausgelesen:

- Geschwindigkeit des Wagens über 4 Radlagersensoren
- Drosselklappenstellung, Motordrehzahl, -temperatur u. v. m. durch Auslesen der Motorsteuerung
- Lenkradstellung durch Winkelsensor

Für den Hawk08 sind außerdem diese Sensoren in der Entwicklung:

- Beschleunigungswerte in 3 Achsen über Beschleunigungssensor

- Öldruck, Öltemperatur und Position des mit der Fahrzeugbewegung drehenden Ölsaugrohres
- Schaltautomatik anhand der Motordrehzahl zur besseren Beschleunigung

Außerdem ist das System mit einem Datenlogger und einem WLAN-Modul ausgerüstet. Der Datenlogger speichert alle Nachrichten des CAN-Busses auf einer Flash-Speicherkarte, sodass diese auch hinterher analysiert werden können, bspw. zur Fehlerdiagnose oder zum Fahrertraining. Über das WLAN-Modul werden alle Daten an die Kontrollstation gesendet, von der u. a. mit einer LabVIEW[6]-Oberfläche der Zustand des Wagens kontrolliert und Einstellungen des Telemetriesystems vorgenommen werden können.

1.2.3 Der Zukunftsgedanke des bisherigen Systems

Von dem bisherigen System wurden lediglich passive Aufgaben übernommen. Es werden Sensoren ausgelesen, verarbeitet und dargestellt, die der Diagnose sowie dem Piloten während der Fahrt nützen. Es wird nicht ins Fahrzeug eingegriffen. Auch bei einem anteiligen oder vollständigem Ausfall des Telemetriesystems kann der Pilot ohne größere Beeinträchtigung oder gar Sicherheitsbedenken die Fahrt fortsetzen. Dies ist besonders wichtig für die Sicherheit des Piloten, des Fahrzeugs und der Umgebung.

Der nächste Schritt der Weiterentwicklung ist die integration von Steuerungsaufgaben in das System. Das bedeutet, dass Systeme, die bislang separat funktionierten oder noch nicht integriert werden konnten, über das Bussystem gesteuert werden. Ein Beispiel sind die Schaltwippen am Lenkrad zum Gangwechsel: Von diesen geht jeweils eine Leitung durch das Fahrzeug zur Getriebesteuerung, die in der Nähe des Motors sitzt.

Stattdessen könnte ein Mikrocontroller, der in der Nähe des Lenkrads oder direkt im Lenkrad sitzt, die Tastimpulse der Schaltwippen auslesen und den Zustand in einer Nachricht über den Bus versenden. Die Getriebesteuerung, die für die Schaltautomatik wegen der Motordrehzahl bereits mit dem Bus verbunden ist, könnte diese Nachricht empfangen und entsprechend reagieren. Dabei würden die beiden Leitungen entfallen, wodurch Gewicht und Aufwand der Verkabelung reduziert wird.

1.3 Gliederung dieser Arbeit

Nachdem diese Einführung einen Einblick in das Umfeld dieser Arbeit gegeben hat, wird das zweite Kapitel die für den Zusammenhang wichtigen Grundlagen erläutern.

[6] LabVIEW ist ein graphisches Programmiersystem von National Instruments für Mess- und Automatisierungstechnik, http://www.ni.com/labview/

Im folgenden dritten Kapitel werden die Anforderungen beschrieben, denen die Entwicklung dieser Arbeit genügen muss. Daraufhin wird in der Analyse untersucht, welche Möglichkeiten es in der Umsetzung typischerweise gibt, und welche davon unter den gegebenen Umständen durchführbar sind.

Das fünfte Kapitel der Konzeption konkretisiert die gewählte Strategie für die Umsetzung. In dem letzten Kapitel der Realisierung werden die Details der Implementierung aufgezeigt.

Die beiden Themen Antriebsschlupfregelung und TimeMaster-Redundanz werden abwechselnd behandelt: In jedem der folgenden 5 Kapitel wird zuerst die Antriebsschlupfregelung und dann die TimeMaster-Redundanz bearbeitet.

Abschließend wird im Kapitel Fazit das Ergebnis besprochen.

In blauer Schrift dargestellte Begriffe sind im Abkürzungsverzeichnis bzw. Glossar wiederzufinden.

2 Grundlagen

In diesem Kapitel werden die Komponenten und Technologien erläutert, die für diese Arbeit relevant sind.

2.1 Fahrassistenzsysteme

Fahrassistenzsysteme sind elektronische und mechanische Zusatzeinrichtungen in Kraftfahrzeugen, welche die Sicherheit des Fahrzeugs, der Insassen und der Umgebung erhöhen sollen. Es gibt Situationen, in denen der durchschnittliche Fahrzeugführer die Kontrolle über das Fahrzeug verliert, da er abgelenkt wurde oder diese außergewöhnliche Situation ihn überfordert.

In diesen kritischen Situationen unterstützen die Fahrassistenzsysteme den Fahrer durch autonome Eingriffe in die Funktionen des Fahrzeugs. Dabei werden z. B. die Bremsen betätigt oder die Antriebskraft reguliert.

Bei der Wahl der Fahrassistenzsysteme muss der Nutzen den Kosten gegenübergestellt werden: So sichert ein Antiblockiersystem auch in einem Rennwagen die Manövrierfähigkeit beim Bremsen. Jedoch ist der Eingriff ins Bremssystem kompliziert, und die Apparatur bringt unerwünschtes Gewicht mit sich.

Man unterscheidet zwischen aktiven und passiven Fahrassistenzsystemen (vgl. Bosch Fahrsicherheitssysteme, 1998, S.4):

2.1.1 Aktive Fahrassistenzsysteme

Aktive Systeme „helfen, Unfälle zu vermeiden und tragen damit vorbeugend zur Sicherheit im Straßenverkehr bei“ (Bosch Fahrsicherheitssysteme, 1998, S.4). Beispiele hierfür sind das Antiblockiersystem, die Antriebsschlupfregelung und das Elektronische Stabilitätssystem. „Diese Sicherheitssysteme stabilisieren das Fahrzeug in kritischen Situationen und erhalten dabei deren Lenkbarkeit“ (Bosch Fahrsicherheitssysteme, 1998, S.4).

2.1.2 Passive Fahrassistenzsysteme

Passive Systeme kommen erst bei einem Unfall zum Einsatz. Sie helfen, die Folgen eines Unfalles zu mildern und die Verletzungsgefahr für Insassen und Außenstehende zu senken. Ein Beispiel ist der Airbag, der die Insassen beim Aufprall schützt (vgl. Bosch Fahrsicherheitssysteme, 1998, S.4).

2.2 Antriebsschlupfregelung

Die Antriebsschlupfregelung ist ein aktives Fahrassistenzsystem, welches das Durchdrehen der Antriebsräder verhindert, solange die physikalischen Grenzen nicht überschritten werden. Die Stabilität und Lenkfähigkeit beim Beschleunigen bleiben gesichert, auch wenn der Fahrer in einer unkontrollierbaren Schlupf-Situation falsch reagiert (vgl. Bosch Fahrsicherheitssysteme, 1998, S. 72).

2.2.1 Problemstellung Bodenhaftung

Die Räder des Fahrzeuges bilden die alleinige Verbindung des Fahrzeuges zum Untergrund. Sie übertragen sämtliche Kräfte, mit denen das Fahrzeug fortbewegt wird, also beim Beschleunigen, Bremsen und Lenken. Geht diese Verbindung teils oder ganz verloren, ist das Fahrzeug nicht mehr sicher steuerbar.

Die Qualität dieser Verbindung wird mit der über die Reifen auf den Untergrund übertragbaren Kraft bewertet. Diese hängt von vielen Faktoren ab:

Unebene Fahrbahn
: Die Räder stets am Boden zu halten ist Aufgabe des Fahrwerks. Wenn dieses jedoch nicht richtig oder für einen anderen Fahrbahntyp eingestellt wurde, können die Räder den Kontakt zur Fahrbahn verlieren.

Bodenbeschaffenheit
: Auf dem gesäuberten Asphalt einer Rennstrecke ist die Haftung der Reifen weitaus besser als auf einer mit Rollsplit bestreuten Straße. Noch schlechter wird die Verbindung auf nasser bis überschwemmter Fahrbahn.

Reifenprofil
: Es gibt verschiedene Reifenprofile für unterschiedliche Situationen. Beispielsweise ist ein profilloser Reifen optimal auf einer trockenen Rennstrecke, aber gänzlich unbrauchbar bei Regen.

Temperaturen

Die Temperaturen von Fahrbahn und Reifen spielen im Rennsport eine große Rolle. Je wärmer diese sind, desto größer ist die Haftung und damit die übertragbare Kraft.

2.2.2 Entstehung von Schlupf

Der übertragbaren Kraft steht die Kraft gegenüber, die der Fahrer mit dem Motor auf die Strasse bringen möchte oder muss, falls er in eine gefährliche Situation gerät und zum Ausweichen gezwungen wird.

Steigt die Antriebskraft an den Rädern durch einen zu starken Tritt auf das Gaspedal über die Kraft, die in dem Moment von den Reifen auf den Untergrund übertragen werden kann, geht die Haftung verloren und die Räder drehen durch. Dies wird als hundertprozentiger Schlupf bezeichnet.

Umgekehrt blockiert das Rad bei einer Vollbremsung, wenn die zu übertragende Kraft in umgekehrter Richtung größer als die maximal übertragbare Kraft ist. Dies wird ebenfalls als Schlupf bezeichnet (vgl. Bosch Fahrsicherheitssysteme, 1998, S. 16). Da dieser jedoch durch ein Antiblockiersystem und nicht durch eine Antriebsschlupfregelung verhindert wird, ist im Folgenden stets der Antriebsschlupf gemeint.

2.2.3 Auswirkung von übermäßgem Schlupf

Ein durchdrehendes oder blockierendes Rad hat nahezu jede Bodenhaftung verloren. Insbesondere können keine Seitenführungskräfte mehr übertragen werden. Während dies bei Geradeausfahrt noch einigermaßen durch sofortiges Gaswegnehmen kontrollierbar ist, bricht bei einem Hinterradantrieb in der Kurve schnell das Heck aus. Der Wagen übersteuert, es muss neben dem Gaswegnehmen auch gegengelenkt werden.

2.2.4 Optimaler Schlupf

Schlupf an sich hat nicht nur eine negative Wirkung. Ohne Schlupf kann ein Reifen keine Kraft übertragen. Man stelle sich ein stehendes, startendes Zweirad vor. Das angetriebene Rad dreht sich, der Reifen verformt sich, 'zieht' dann an dem Untergrund, und das Rad bewegt sich. Ohne Schlupf müsste sich mit dem ersten Moment der Raddrehung auch das nichtangetriebene Rad drehen. Durch die Flexibilität der Reifen ist dies aber nicht möglich. Die verformung des Reifens wird Walkarbeit genannt, dabei entsteht der Energieverbrauch durch die Erwärmung des Reifens.

Tatsächlich ist sogar ein Schlupf von 10 % üblich für Autos, auch bei gemäßigter Fahrweise und unter guten Bedingungen, wie aus der Grafik 2.1 abzulesen ist.

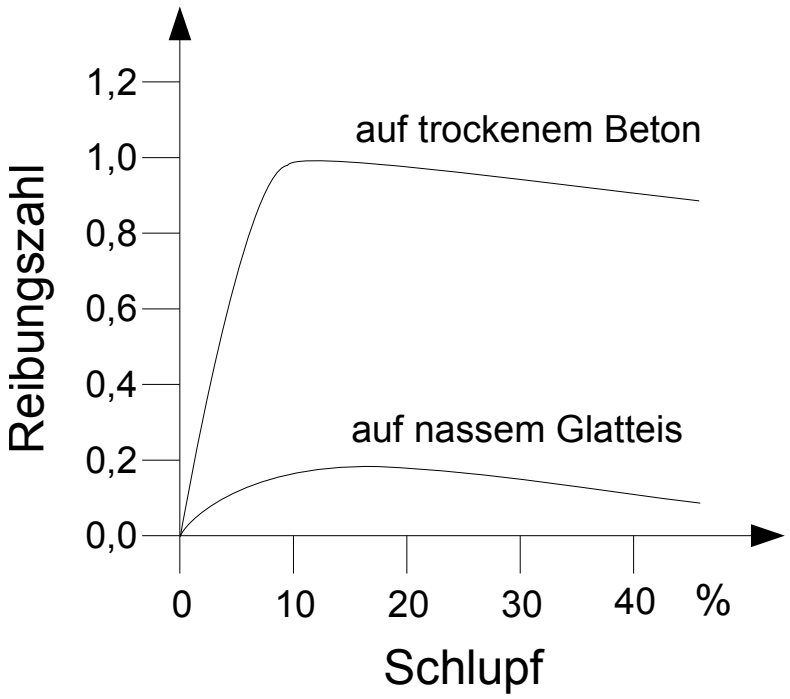

Abbildung 2.1: Reibungszahl in Abhängigkeit vom Schlupf
(vgl. Bosch Fahrsicherheitssysteme, 1998, S.16)

Die Reibungszahl entspricht hierbei indirekt der Kraft, die zur Fortbewegung umgesetzt werden kann. Eine Reibungszahl von 1 ist das Maximum an Kraft, welches mit einem Reifen in Fahrtrichtung übertragen werden kann.

2.3 Redundanz

Redundanz ist definiert als das „Vorhandensein von mehr als für die Ausführung der vorgesehenen Aufgaben an sich notwendigen Mitteln“ (DIN 40041, 1967, 4.2.1).

Für die volle Funktion des zeitgesteuerten Bussystems ist ein TimeMaster ausreichend. Wenn dieser jedoch ausfällt, ist das System nicht mehr funktionsfähig. Um diesem Problem zu entgehen, werden die TimeMaster redundant ausgelegt. Dabei werden mehrere TimeMaster integriert, die bis auf einen aktiven als Reserve-TimeMaster fungieren.

2.4 Bussysteme

Ein Bussystem, kurz Bus, bezeichnet eine kompakte Verbindung mehrerer Komponenten. Dabei wird nicht jede Komponente über eine getrennte Leitung mit jeder anderen Verbunden, sondern alle Komponenten sind an eine Leitung geschlossen. Was genau mit Leitung gemeint ist, hängt von dem Bussystem ab. Es gibt Eindrahtsysteme, Zweidrahtsysteme (Takt und Daten, oder 2x Daten in beide Richtungen) und Dreidrahtsysteme (Takt und 2x Daten in beide Richtungen) in verschiedensten Varianten, bis hin zu parallelen Systemen mit weitaus mehr Datenleitungen pro Richtung.

Teils wird auch die Zufuhr der Stromversorgung an mehrere Komponenten über ähnliche Konstrukte als Bus bezeichnet, in dieser Arbeit ist aber nur die Datenverbindung gemeint. Die an einen Bus angeschlossenen Komponenten werden auch Busteilnehmer oder Knoten genannt.

2.4.1 Vorteile

Durch die beträchtlich geringere Anzahl von Leitungen sinken der Verkabelungsaufwand und das Kabelgewicht. Die Komponenten müssen nur noch eine Schnittstelle zur Kommunikation bereithalten statt einer Schnittstelle für jede andere Komponente. Dadurch ist ein bestehendes System einfach erweiterbar (vgl. Lawrenz, 1999, S. 12).

Nachrichten, die an alle Teilnehmer geschickt werden sollen, müssen bei einem Bussystem nicht vielfach an jeden Empfänger, sondern nur einmal verschickt werden, da alle Teilnehmer des Busses diese Nachrichten gleichzeitig empfangen.

2.4.2 Nachteile

Als Nachteil zählt der größere Verwaltungsaufwand, der beim Senden einer Nachricht über ein Bussystem getätigt werden muss. Der Empfänger muss außerdem bei jeder Nachricht, die ihn erreicht, prüfen, ob diese für ihn bestimmt ist bzw. für ihn relevante Daten enthält.

2.4.3 Koordination

Es gibt verschiedene Verfahren, um die Kommunikation vieler Teilnehmer über eine Leitung zu koordinieren. Bei manchen erteilt eine Master-Komponente den anderen Slave-Komponenten die zeitweise Erlaubnis zur aktiven Nutzung des Busses. Dies geschieht über separate Leitungen, genannt Slave Select oder auch Chip Select.

Da diese separaten Leitungen jedoch dem Konzept der kompakten Verkabelung widersprechen und nur über die Distanz einer Leiterplatine praktikabel sind, benutzen die meisten anderen Bussysteme auf höherer Ebene ein nachrichtenorientiertes Protokoll. Diese Nachrichten dienen zur Koordination von Buszugriffen und können dabei Identifizierer, Adressen und Daten enthalten.

2.4.4 Ein ereignisgesteuerter Bus

Bei einem ereignisgesteuerten Bus versenden die Busteilnehmer ihre Nachrichten zu einem beliebigen Zeitpunkt; beispielsweise wird direkt nach Auslesen des Sensors der gelesene Wert über den Bus weitergeleitet. Dabei muss, falls gerade eine andere Nachricht übertragen wird, auf das Ende dieser Übertragung gewartet werden.

Eine Nachricht mit niedriger Priorität könnte dabei dauerhaft durch höherpriorisierte Nachrichten blockiert werden. Vom frühestmöglichen und im Bereich der Echtzeitdatenauswertung auch gewünschten Sendezeitpunkt der Sensordaten bis zum tatsächlichen Sendezeitpunkt liegt also eine Zeitspanne, über die zum Entwicklungszeitpunkt keine verlässlichen Aussagen getroffen werden können. Entsprechende Zeitmessungen am Endprodukt wären nur für die jeweilige Konstellation gültig, Erweiterungen des Systems würden die Wiederholung der Tests oder ein Herabsetzen der Echtzeitanforderung bedeuten.

2.4.5 Ein zeitgesteuerter Bus

Bei einem zeitgesteuerten Bus werden Zeitfenster (engl. Slots) für das Schreiben auf den Bus festgelegt. Diese Zeitfenster bilden einen Zyklus, der sich fortlaufend wiederholt.

Die Länge eines solchen Zyklus hängt von der gewünschten Nachrichtenfrequenz ab und variiert von einer Millisekunde bis zu einer Sekunde. Bei dem im Rennwagen integrierten System dauert ein Zyklus 40 ms, sodass auf einfache Weise Sendefrequenzen von 25, 50 und 100 Hertz möglich sind, indem die Nachricht ein-, zwei- oder viermal pro Zyklus gesendet wird.

Das bedeutet, ein Busteilnehmer hat pro zugewiesenem Zeitfenster alle 40 ms die Gewährleistung, eine Nachricht versenden zu können, unabhängig von der Priorität seiner Nachricht. Ein solches System eignet sich *„besonders gut für Anwendungen, in denen der gesamte Nachrichtenverkehr periodischer Natur ist.“* (Müller, 2001, S. 44)

2.4.5.1 Synchronisation

Die unbedingt zu erfüllende Herausforderung hierbei besteht darin, eine globale Zeit, welche das aktuelle Zeitfenster angibt, bei allen aktiven Busteilnehmern synchron zu halten. Es kann nicht davon ausgegangen werden, dass die Hardware-Timer der Teilnehmer über viele Sekunden hinweg ausreichend synchron laufen. Die Abweichungen der Zeitzähler der verschiedenen Knoten können durch Oszillatorungenauigkeiten, unterschiedliche Systemfrequenzen, Produktionstoleranzen, Hardwareeigenschaften und Umgebungseinflüsse entstehen.

2.4.5.2 Die Funktion eines TimeMasters

Um diese Drift zu korrigieren, werden die Knoten periodisch synchronisiert. Dazu verschickt ein spezieller Teilnehmer, der TimeMaster (dt.: Zeitgeber), zu Beginn des Zyklus eine Referenznachricht an alle Teilnehmer, an die sich diese synchronisieren.

Da den Knoten die Soll-Länge eines Zyklus bekannt ist, gibt die Differenz der Empfangszeitstempel zweier Referenznachrichten darüber Aufschluss, ob die eigene Uhr korrekt läuft. Ist dies nicht der Fall, muss die Uhr korrekt gestellt und wenn möglich ihre Geschwindigkeit angeglichen werden. Die Synchronisation geschieht fehlerfrei, sofern die Abweichungen in einem Zeitzyklus eine jeweilige Toleranzgrenze nicht überschreiten.

2.4.6 Der CAN-Bus

Der Controller Area Network-Bus ist ein ereignisgesteuerter (siehe 2.4.4), flexibler Bus für die asynchrone Kommunikation verteilter Knoten. Er wurde von Bosch[1] 1983 entwickelt, in dem ISO-Standard 11898 international standardisiert und ist „das derzeit am häufigsten eingesetzte Kfz-Bussystem sowohl für Low-Speed- als auch für High-Speed-Anwendungen“ (Zimmermann und Schmidgall, 2006, S. 32).

CAN wurde zur „Kommunikation zwischen mehreren Kfz-Steuergeräten zum Austausch von Mess-, Steuer- und Regelsignalen im Echtzeitbetrieb mit hoher Fehlersicherheit“ (Zimmermann und Schmidgall, 2006, S. 32) entwickelt.

Er basiert auf einem Bitstrom-orientierten Übertragungsprotokoll mit bidirektionaler Zwei-Draht-Leitung als Linien-Bus, über den Botschaften mit 11 bzw. 29 (CAN 2.0A bzw. B) Bit Identifizierer und bis zu 8 Datenbyte versendet werden (vgl. Zimmermann und Schmidgall,

[1] Robert Bosch GmbH, http://www.bosch.de

2006, S. 32). Der Identifizierer beschreibt den Absender oder den Inhalt der Nachricht, ermöglicht Empfangsfilter und verhindert Nachrichtenkollisionen: Bei gleichzeitigem Sendeversuch wirkt die Nachricht mit niedrigerem Identifizierer dominant und wird fortgesetzt, die andere Nachricht wird ausgesetzt und später wiederholt. Der Zugriffsmechanismus wird als CSMA/CA – Carrier Sense, Multiple Access, Collision Avoidance – bezeichnet.

Es werden zur Anbindung eines Mikroprozessors ein Controller und ein Transceiver benötigt. Der Controller behandelt das Protokoll des Data Link Layers, welches die Kommunikation zwischen mehreren Controller regelt. Der Transceiver ist für die Umsetzung der Nachrichtensignale auf das Differenzsignal des Physical Layers zuständig, d. h. die Daten auf einem Medium zu übertragen (siehe dazu auch den folgenden Abschnitt 2.5.1) (vgl. Zimmermann und Schmidgall, 2006, S. 32).

Vorteilhaft ist die „netzweite Datenkonsistenz, da alle CAN-Controller die empfangenen Daten ignorieren, wenn ein oder mehrere Geräte einen Übertragungsfehler erkennen“ (Zimmermann und Schmidgall, 2006, S. 32). Zudem ist eine automatische Übertragungswiederholung im Fehlerfall und eine automatische Abschaltung von defekten Controllern, realisiert durch Fehlerzähler, spezifiziert.

Durch die Nutzung eines CAN-Busses ist diese Verbindung der Knoten stets funktionsfähig, unabhängig davon, wieviele der Knoten mit dem Bus verbunden sind und ob diese aktiv sind (vgl. Schulze, 1996, S. 169).

2.4.7 TTCAN

TTCAN steht für Time-Triggered CAN und ist somit die zeitgesteuerte Variante des CAN-Busses (siehe 2.4.5 Ein zeitgesteuerter Bus).

Beim TTCAN wird der Zyklus Matrixzyklus genannt und in mehrere gleich lange Basiszyklen unterteilt. Zur Synchronisation wird zu Beginn jedes Basiszyklus die Referenznachricht versendet.

Die Abbildung 2.2 auf der nächsten Seite veranschaulicht einen Matrixzyklus mit 4 Basiszyklen in der üblichen Darstellungsweise mit untereinander angeordneten Basiszyklen. Durch Platzierung einer Nachricht in zwei oder vier Basiszyklen lassen sich doppelte bzw. vierfache Sendefrequenzen erreichen, sodass die Busnutzung einfach an die nötigen Anforderungen angepasst werden können.

In der Abbildung 2.2 auf der nächsten Seite werden die Nachrichten D, E, F und G mit 25 Hertz, die Nachrichten A und B mit 50 Hertz und die Nachricht C mit 100 Hertz versendet. Als Besonderheit des TTCAN muss die CAN-typische Sendewiederholung deaktiviert

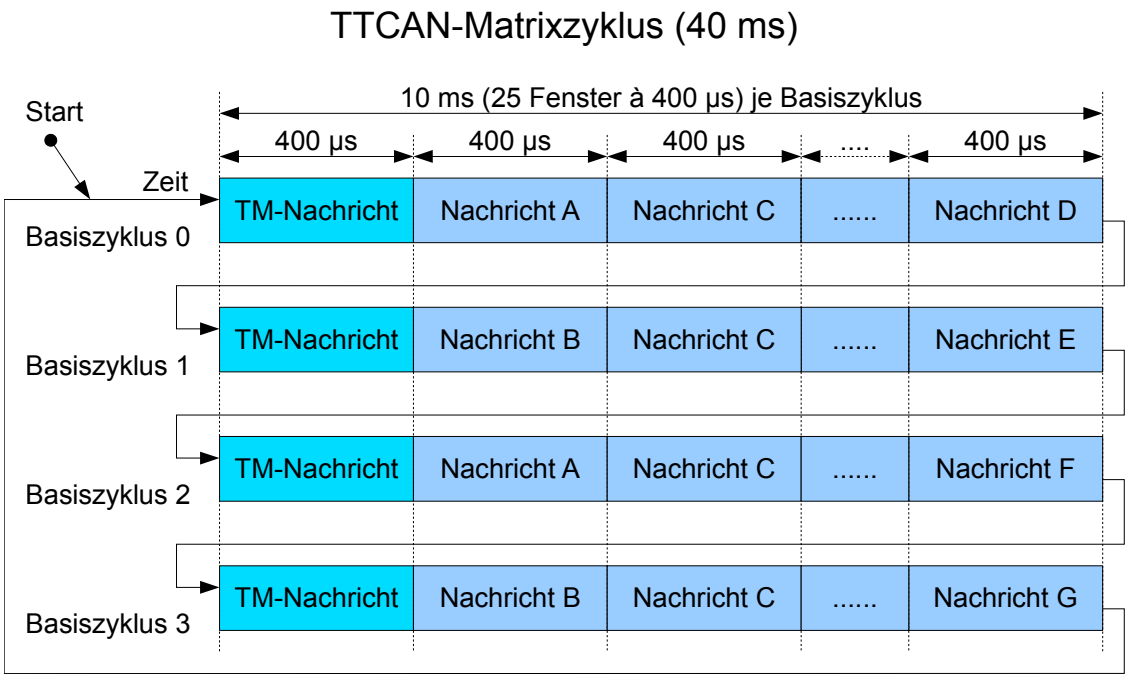

Abbildung 2.2: Zeiteinteilung des TTCAN in Matrixdarstellung

werden. Sonst würde bei einem Fehler oder bei einem fälschlicherweise blockiertem Zeitfenster das erneute Senden der Nachricht in das nächste Zeitfenster hineinragen, sodass die Matrixstruktur nicht eingehalten werden könnte.

2.4.8 Die Idee mehrerer TimeMaster

Wie die Analyse später zeigt (siehe 4.3 Mögliche Ausfallursachen des Bussystems), gibt es mehrere Ursachen, welche die Kommunikation des TTCAN beeinträchtigen oder sogar zerstören können. Die Ursache, die in dieser Arbeit behandelt wird, ist die des Synchronisationsausfalls.

Um diese Gefahr zu umgehen, soll die Idee der TimeMaster-Redundanz (vgl. TTCAN, 2000, 3.) verfolgt werden. Es gibt dann mehrere TimeMaster, die sich arrangieren und im Fehlerfalle eines TimeMasters füreinander einspringen.

Im bisherigen Protokoll (vgl. Haase, 2007, 4.2.2) sind die CAN-IDs 1 bis 7 für TimeMaster vorgesehen. Der bislang einzige integrierte TimeMaster sendete mit der ID 1. Für die Entwicklung in dieser Arbeit stehen also für 7 TimeMaster IDs zur Verfügung. Die Anzahl könnte

man durch Modifikation des Protokolls erhöhen, jedoch ist dies unerwünscht (siehe die Anforderung 3.2.6 Protokollkompatibilität) und bei einem System dieser Größenordnung nicht erforderlich (siehe 1.2 Das bestehende Telemetriesystem).

2.5 Der Mikrocontroller AT90CAN128

Alle bisher im System verwendeten Mikrocontroller sind dasselbe Modell AT90CAN128 der Atmel Corporation[2]. Er basiert auf einer 8-bit RISC[3] Architektur, unterstützt bis zu 16 MHz Systemfrequenz und kann auf 128 KB Flash, 4 KB EEPROM[4] und 4 KB SRAM[5] als Speicher zurückgreifen (vgl. AT90 Datenblatt, 2008).

Weiterhin integriert sind diverse Peripherien, darunter 4 Zähler mit Vergleichsregistern und PWM[6]-Generatoren, 8-bit Analog-Digital-Wandler, 2 USARTs[7] sowie jeweils einen Bus-Controller für CAN[8], SPI[9] und TWI[10]. Der AT90CAN128 ist das komplexeste Modell der 8-bit-RISC-Reihe der Atmel Corporation.

2.5.1 Mikrocontroller und CAN-Transceiver

Der integrierte CAN-Controller des AT90CAN128 kann nicht direkt an die CAN-Bus-Leitung angeschlossen werden, da unterschiedliche Signalpegel vorliegen. Die Aus- und Eingänge des Mikrocontrollers werden gegen Masse gesetzt bzw. gelesen, die beiden Adern der CAN-Leitung werden als Differenzsignal angesehen. Für diese Umsetzung ist der Transceiver zuständig.

2.5.1.1 Differenzsignal

Bei einem Differenzsignal werden die Adern verglichen: Liegt an diesen eine Spannung ungleich Null an, wird dies als dominanter Pegel gewertet, andernfalls als rezessiver. Der Vorteil des Differenzsignals ist, dass Gleichtaktstörungen, also elekromagnetische Störungen, die

[2] Atmel Corporation: http://www.atmel.com
[3] Reduced Instruction Set Computing, das Rechnen mit reduziertem aber flexiblem Befehlssatz
[4] Persistenter Speicher, vom Mikrocontroller-Programm aus beschreib- und löschbar
[5] Static Random Access Memory: schnelle flüchtige Speicherzellen als Arbeitsspeicher
[6] Pulsweitenmodulation: genutzt für z. B. primitive Digital-Analog-Wandlungen oder Servo-Ansteuerungen
[7] Universial Synchronous Asynchronous Receiver Transmitter: Controller für serielle Schnittstellen
[8] Controller Area Network, siehe 2.4.6 Der CAN-Bus
[9] Serial Peripheral Interface: serielle Dreidrahtleitung mit Master-Slave-Prinzip
[10] Two Wire Inteface, identisch mit I^2C, Inter-Integrated Circuit

sich auf die CAN-Leitung auswirken, keine Fehler verursachen, da sich durch eine Spannungsverschiebung beider Adern die Differenz nicht verändert.

2.5.1.2 Dominanter & rezessiver Pegel

Der Buszugriffsmodus CSMA/CA (siehe 2.4.6) des CANs erfordert außerdem eine spezielle Betrachtung der Leitungspegel. Diese werden als dominant und rezessiv bezeichnet. Die Busleitung wird über Widerstände auf dem rezessiven Pegel gehalten. Alle passiven Busteilnehmer schalten auf den neutralen Zustand Tri-State, beeinflussen den Buspegel also nicht.

Wenn ein Busteilnehmer aktiv werden möchte, zieht er die Leitung auf den dominanten Pegel. Dies können alle verbundenen Teilnehmer registrieren. Um ein rezessives Bit zu senden, lässt er die Leitung von den Widerständen auf den rezessiven Pegel ziehen. Wenn dabei ein anderer Teilnehmer ein dominantes Bit sendet, also die Leitung aktiv auf den dominanten Pegel zieht, überschreibt er damit die Information des ersten Teilnehmers. Dieser bemerkt dies, stellt das Senden ein und versucht es gegebenenfalls später erneut. Mit dieser Logik arbeitet die Busarbitration, da dominante Bits gegenüber rezessiven Vorrang haben.

2.5.1.3 Anschluss des Transceivers

Der CAN-Transceiver besitzt neben der Spannungsversorgung die Ein- und Ausgänge CAN-High und CAN-Low, den Eingang Tx und den Ausgang Rx. Die Tx-Leitung ist die Sendeleitung des CAN-Controllers im Mikrocontroller. Ist diese auf High-Pegel (5 Volt) beeinflusst der Transceiver die CAN-Leitungen nicht, liegt sie auf Low-Pegel (0 Volt) zieht er die CAN-Leitung auf den dominanten Pegel.

Die Rx-Leitung gibt den Zustand der CAN-Leitung wieder, also High-Pegel (5 Volt) bei rezessivem Pegel und Low-Pegel (0 Volt) bei dominantem Pegel. Durch eine Differenz zwischen Tx- und Rx-Leitung erkennt der CAN-Controller, dass sein rezessives Bit durch ein dominantes eines anderen Controllers überschrieben wurde.

2.5.2 Der TTCAN-Fehler des CAN-Controllers

Um einen TTCAN zu betreiben, reicht ein einfacher CAN-Controller nicht aus. Denn wenn beim Senden auf einem TTCAN ein Fehler auftritt, muss diese Nachrichtenübertragung abgebrochen werden. Einfache CAN-Hardware würde in einem solchen Fall versuchen die Nachricht bei nächster Gelegenheit erneut zu versenden. Beim TTCAN würde die Nachricht dann nicht mehr in dem dafür vorgesehenen Slot liegen, andere Nachrichten würden

verzögert und der gewünschte Determinismus zerstört werden. Aus diesem Grunde ist die automatische Sendewiederholung von CAN-Nachrichten nicht erlaubt (vgl. TTCAN, 2000, 3.2).

In dem Mikrocontroller AT90CAN128 gibt es zur Initialisierung die Möglichkeit, über ein Flag den Controller in einem TTC-Modus zu betreiben. Dann werden bei Fehlern keine Sendewiederholungen gestartet.

Im TTCAN-Standard ist aber noch eine ähnliche Situation spezifiziert: Neben fest vergebenen Zeitfenstern sind auch arbitrierende möglich, in denen weniger wichtige, sporadisch auftretende Nachrichten versendet werden können. Wenn ein Knoten eine Nachricht in einem solchen Busslot versenden möchte, aber die Arbitrierung fehlschlägt, weil der Bus durch eine höher priorisierte Nachricht belegt ist, wird wie bisher auf die CAN-interne Sendewiederholung verzichtet, der Knoten muss es im nächsten Matrixzyklus erneut versuchen.

Das Problem für den Atmel Mikrocontroller liegt darin, dass der integrierte CAN-Controller laut Datenblatt die folgenden Fehlertypen kennt und bei diesen etwa die Sendewiederholung deaktiviert (vgl. AT90 Datenblatt, 2008, 19.7.2):

- Bit-Fehler (Leitungsfehler)
- Stuff-Bit-Fehler
- CRC-Fehler (beim Empfänger)
- Form-Fehler (Aufbau der Nachricht)
- Acknowledgement-Fehler (beim Sender)

Arbitrierungsfehlschläge gelten laut Datenblatt also nicht als Fehler. Ein Versuchsaufbau meinerseits hat ergeben, dass bei Busbelegung auf einen freien Bus gewartet und die Nachricht anschließend versendet wird. Demnach ist der Mikrocontroller-interne CAN-Controller trotz TTC-Flag nicht für arbitrierende Zeitfenster geeignet. Bestätigung brachte eine Nachfrage bei Atmel Corporation[11].

Diese Einschränkung ist relevant für die Analyse und die Konzeption, da hierdurch ein auf der Arbitrierung basierendes Konzept nicht möglich ist.

[11] Atmel Corporation: http://www.atmel.com

2.6 Scheduler

Die Programme der Mikrocontroller basieren auf einem getaktetem Scheduler (dt. Planer), der von Schuckert (2007) nach Pont (2001) entwickelt wurde. Ein Scheduler kann als ein einfaches Betriebssystem angesehen werden, welches die periodische Durchführung mehrerer Aufgaben, Tasks genannt, ermöglicht (vgl. Pont, 2001, S. 245).

Dazu basiert ein getakteter Scheduler auf einem Hardware-Timer, der bei jedem Überlauf einen Interrupt auslöst, welcher eine erneute Bearbeitung des Schedulings bewirkt. Dieser Interrupt wird Tick genannt und ist die kleinste Einheit des Schedulers. In diesem System beträgt ein Tick 200 μs.

Der hier integrierte Scheduler verhält sich kooperativ, d. h. dass der laufende Task nicht unterbrochen werden kann (vgl. Schuckert, 2007, S. 23). Ist der Task am Ende seiner vorgegebenen Zeit nicht abgeschlossen, registriert der Scheduler dies als Fehler (vgl. Pont, 2001, S. 246).

Zudem bietet der Scheduler die Möglichkeit, eine verhältnismäßig kleine Task zu definieren, die jeden Tick ausgeführt wird. Diese besitzt präemptives Verhalten, unterbricht einen laufenden Task also nur kurzzeitig anstatt ihn abzubrechen. Die Kombination aus kooperativen und präemptiven Verhalten wird Hybridscheduler genannt.

2.7 Watchdog

Ein Watchdog (dt. Wachhund) ist eine Sicherheitsfunktion von Mikrocontrollern, die eine Gruppe von Software- und Hardwarefehlern entschärfen soll. Diese Fehler führen zu Endlosschleifen oder zu einer Situation, in der der Prozessor zusammenhanglose Maschinenbefehle ausführt. Dies geschieht, wenn das Befehlszählerregister einen ungewollten Wert annimmt.

Problem

Dies kann durch Fehler in der Software, z. B. bei unzuverlässiger Berechnung des Befehlszählerregisterwertes oder Störungen in der Hardware entstehen, etwa wenn die Spannungsversorgung kurzfristig einbricht ohne einen vollständigen Reset auszulösen. So wird ein Befehl ohne Bezug zur aktuellen Situation, ein Datenwort oder eine ungenutzte Speicherzelle als Befehl interpretiert.

Funktion

In beiden Fällen ist das Programm nicht in der Lage einen korrekten Zustand wiederherzustellen, ein Reset ist dazu die einzige Möglichkeit. Diese Aufgabe übernimmt der Watchdog. Dieser ist im Prinzip ein Hardware-Zähler, der bei Programmstart aktiviert wird, zu zählen beginnt und bei Überlauf einen Reset auslöst. Diese Laufzeit hängt von der Systemfrequenz und der Watchdogkonfiguration ab, bei dem verwendeten Mikrocontroller (siehe Abschnitt 2.5) und 16 MHz Systemtakt ist eine Laufzeit von 15 ms bis 2 s wählbar (vgl. AT90 Datenblatt, 2008, S. 58).

Zurücksetzen

Um dem Watchdog-Zähler die korrekte Funktion der Software zu beweisen, muss diese regelmäßig vor Ablauf der Laufzeit den Zähler zurücksetzen. Als zusätzliche Schutzvorrichtung ist hierfür das aufeinanderfolgende Schreiben von zwei Registern innerhalb von 4 Takten erforderlich (vgl. AT90 Datenblatt, 2008, S. 58).

Bei Programmstart kann aus der Hardware ausgelesen werden, ob der letzte Reset durch den Watchdog verursacht wurde. So kann, wenn sich der Fehler wiederholt, eine Alternativprozedur ausgeführt werden kann, beispielsweise zur Meldung des Fehlers.

3 Anforderungen

Dieses Kapitel stellt die Anforderungen zusammen, denen das System genügen muss, und schildert die Randbedingungen, unter denen das System entwickelt wird.

3.1 Anforderungen ASR

Die Anforderungen an eine ASR beziehen sich auf die Wirkung, das Informieren des Fahrers, die Schadensfreiheit und die zum Wettbewerb einzuhaltenden Regelanforderungen.

3.1.1 Wirkung

Identifikator
Wirkung

Beschreibung
Die Antriebsschlupfregelung soll dem Fahrer einen Nutzen bringen. Dabei soll die Antriebsschlupfregelung den Schlupf feinfühliger regeln, als es der Fahrer ohne Hilfsysteme könnte. Das Fahrzeug darf durch die Antriebsschlupfregelung nicht anders reagieren als es der Fahrer nach kurzer Gewöhnungsphase erwartet.

Quelle
Siehe Abschnitte 2.2 Antriebsschlupfregelung und 2.2.4 Optimaler Schlupf.

Abnahmekriterium
Die Wirkung der Antriebsschlupfregelung ist schwer zu bewerten. Man könnte die benötigte Zeit für einen Beschleunigungsabschnitt oder einen Rundkurs messen, je einmal mit und ohne aktivierter Antriebsschlupfregelung. Ist die Zeit des Laufes mit aktivierter Antriebsschlupfregelung kürzer, wurde diese erfolgreich entwickelt. Zudem sollten viele geübte Testfahrer nach einem Urteil befragt werden.

3.1.2 Fahrerinformation

Identifikator

Fahrerinformation

Beschreibung

Der Fahrer soll in jedem Moment informiert sein, in welchem Zustand sich die Antriebsschlupfregelung befindet.

Problembeschreibung

Wenn der Fahrer nicht nachvollziehen kann, ob das momentane Verhalten des Autos von ihm, von der Antriebsschlupfregelung oder von übrigen Einflüssen, wie z. B. Fahrbahnhaftung, Fahrwerkseinstellung, ausgeht, kann er im weiteren Verlauf der Fahrt nicht angemessen reagieren.

Wenn der Fahrer sich auf die Antriebsschlupfregelung verlässt, diese aber in der nächsten Kurve wegen einer überschrittenen Temperaturschwelle nicht regelt und nicht für Stabilität sorgt, die Fehleranzeige dafür aber fehlt, ist das Fahrzeug nicht mehr Steuerbar und kommt von der Fahrspur ab.

Quelle

Siehe dazu 2.2.1 Problemstellung Bodenhaftung, 5.1.4 Zustände der ASR und 4.2.3.3 Motorsteuerung: Zündwinkel verstellen.

Abnahmekriterium

Der Fahrer muss über den Zustand der Antriebsschlupfregelung informiert werden und darf nicht das Gefühl der Unkontrollierbarkeit haben.

3.1.3 Schadensfreiheit

Identifikator

Schadensfreiheit

Beschreibung

Die Antriebsschlupfregelung soll so entwickelt werden, dass sie entweder prinzipbedingt keinem Bauteil des Fahrzeugs schädigen kann, oder so, dass dies durch entsprechende Kontrollsensoren oder eine Regelzeitbegrenzung verhindert wird. Solange diese Grenzen überschritten sind, darf die Antriebsschlupfregelung nicht regeln, um nicht das Risiko einzugehen, die Fahrtüchtigkeit des Rennwagens zu beeinträchtigen.

Eine (kurzfristig) inaktive Antriebsschlupfregelung ist zwar unvorteilhaft, aber einem womöglich irreparablen Totalausfalls des Rennwagens vorzuziehen.

Problembeschreibung

Durch den Eingriff in verschiedene kritische Systeme des Fahrzeugs, z. B. Bremssystem und Antriebsstrang, besteht die Gefahr diese Systeme zu zerstören, wenn der Eingriff nicht gegen alle Eventualitäten abgesichert ist. Die Eventualitäten sind unterschiedlich, je nach gewählter Umsetzung.

Quelle

Siehe dazu 4.2.1 Bremseneingriff und 4.2.3.3 Motorsteuerung: Zündwinkel verstellen.

Abnahmekriterium

Es sollen die gängigen Sicherheitsvorkehrungen getroffen werden um die Elektronik abzusichern, wie etwa einen Watchdog (siehe Abschnitt 2.7) zu integrieren und geschirmte Leitungen zu verwenden.

Außerdem soll mit der zuständigen Baugruppe des Teams die Umsetzung detailiert besprochen werden.

3.2 Anforderungen TTCAN

Im folgenden sind die Anforderungen gelistet, die einen zuverlässigen TTCAN Bus kennzeichnen, sofern diese Teil des Konzepts der TimeMaster-Redundanz sind. Dieses ist im Abschnitt 2.4.8 Die Idee mehrerer TimeMaster beschrieben.

3.2.1 Verfügbarkeit

Identifikator

Verfügbarkeit

Beschreibung

Die Referenznachricht ist für den korrekten Betrieb des zeitgesteuerten Busses zwingend erforderlich.

Problembeschreibung

Wie bereits unter 2.4.7 TTCAN beschrieben ist die Verfügbarkeit des Busses von der Referenznachricht eines TimeMasters abhängig. Die Ausfallsicherheit der Referenznachricht ist die Hauptanforderung dieser Arbeit. Das zugrunde liegende Prinzip ist das der TimeMaster-Redundanz, siehe dazu auch den Abschnitt 2.4.8 Die Idee mehrerer TimeMaster. Von welchem Modul diese Nachricht versendet wird, soll nicht relevant sein.

Ein Ausbleiben der Referenznachricht muss schnellstmöglich erkannt und das Senden von einem Ersatz-TimeMaster übernommen werden. Um auch mehrere Ausfälle von TimeMastern überstehen zu können, soll die Möglichkeit bestehen mehrere Reservemodule zu integrieren.

Quelle

Siehe Abschnitte 2.4.7 TTCAN und 5.2.2 Knotenzeitabweichung.

Abnahmekriterium

Wenn die Module so progammiert werden, dass eine Referenznachricht ausbleiben darf, muss bei Ausbleiben einer Nachricht diese direkt nachgeholt werden. Das bedeutet, innerhalb des 400 μs langen Zeitfensters der Referenznachricht soll das Ausbleiben erkannt, die Ersatz-Nachricht gesendet und von den Modulen empfangen worden sein.

Ist dies nicht möglich, muss die Referenznachricht des folgenden Basiszyklus vom Ersatz-TimeMaster gesendet werden, sodass nur eine Referenznachricht ausbleibt. Die anderen Busteilnehmer müssen dies entsprechend akzeptieren. Die erste Methode ist der zweiten vorzuziehen, da sich durch die fehlende Referenznachricht im zweiten Basiszyklus die Ungenauigkeit der Synchronisation verdoppelt, siehe dazu den Abschnitt 5.2.2 Knotenzeitabweichung.

3.2.2 Verzögerungsfreiheit

Identifikator

Verzögerungsfreiheit

Beschreibung

Das System ist ein aktives Echtzeitsystem. Bei Ausfall eines TimeMasters darf der Bus nicht stillliegen, etwa weil erst entschieden werden muss, welcher Reserve-TimeMaster den ausgefallenen ersetzt.

Problembeschreibung

Das System kann während der Fahrt nicht in einen sicheren Zustand überführt werden, in dem bis zur Lösung des Problems verblieben werden kann. Würde der Bus stillliegen, bekämen der Fahrer und aktive Komponenten des Systems keine Sensorinformationen mehr. Der Fahrer allein kann auch ohne Geschwindigkeitsanzeige fahren, ein für den Fahrbetrieb erforderliches oder zumindest erwünschtes System ist ohne Daten nicht funktionsfähig.

Der Wechsel der TimeMaster muss also in kürzester Zeit stattfinden, um den Basiszyklus nicht aussetzen zu lassen und ihn im besten Fall sogar ohne Zeitverzug fortsetzen zu können.

Würde ein Basiszyklus ausgesetzt werden, hätte das für ein sicherheitsrelevantes Fahrassistenzsystem u. U. verheerende Folgen, wie folgendes Beispiel zeigt: Das Fahrzeug fährt mit 200 km/h durch eine größere Kurve und das Elektronische Stabilitätsprogramm steuert zur Stabilisierung des Fahrzeugs das Bremssystem über den TTCAN-Bus an. Fällt ein Basiszyklus aus, bewegt sich das Fahrzeug einen halben Meter vorwärts, wie folgende Rechnung zeigt:

$$200\frac{km}{h} \times 10ms = 200\frac{m}{h} \times 10s = 2000\frac{m \times s}{3600s} = \frac{20}{36}m = 0.\bar{5}m$$

Bei nur einem TimeMaster, der bei einem Ausfall im 2. Basiszyklus statt den 3. Zyklus fortzusetzen beim 0. beginnt, würden Nachrichten des 3. Zyklus um 3 Zyklen verzögert werden. Hierbei muss noch die Startup-Zeit des TimeMasters von bis zu 70 ms beachtet werden, sodass sich mit 3 Basiszyklen je 10 ms eine Totzeit von 100 ms ergibt, was einem Fahrweg von 5 Metern entspricht:

$$200\frac{km}{h} \times 100ms = 200\frac{m}{h} \times 100s = 20000\frac{m \times s}{3600s} = \frac{200}{36}m = 5.\bar{5}m$$

Reagiert das System in diesen 5,5 Metern nicht, bricht das Fahrzeug mit hoher Wahrscheinlichkeit aus.

Quelle

Siehe Abschnitte 2.4.7 TTCAN und 5.2.2 Knotenzeitabweichung.

Abnahmekriterium

Der Zeitverzug wird gemessen zwischen dem vorgeschriebenen Sendebeginn der ursprünglichen Nachricht und dem Sendebeginn der einsetzenden Nachricht. Liegt dieser unter der Toleranzgrenze für die Knotenzeitabweichung (siehe 5.2.2), ist die Verzögerung akzeptabel.

3.2.3 Zykluskonsistenz

Identifikator

Zykluskonsistenz

Beschreibung

Der zeitgesteuerte Bus befindet sich immer in einem von 4 Basiszyklen. Der aktuelle

Basiszyklus wird über die Referenznachricht mitgeteilt. Die korrekte Reihenfolge der Basiszyklen ist wichtig für eine störungsfreie Kommunikation, und, da die Scheduler der Module an den Zyklus gekoppelt sind, auch für einen störungsfreien Ablauf der Programme.

Problembeschreibung

Wenn der Scheduler eine Referenznachricht mit der Zyklusnummer 3 erwartet, aber eine mit der Zyklusnummer 1 empfängt, nimmt er an, dass ein nicht behebbarer Fehler aufgetreten ist, und führt eine Notfall-Funktion (meist einen Reset) aus (vgl. 4.5 Bisher implementiertes Verhalten).

Quelle

Siehe Abschnitte 2.4.7 TTCAN und 2.6 Scheduler.

Abnahmekriterium

Ein Reserve-TimeMaster, der das Senden der Referenznachricht übernimmt, muss also die Zyklusnummer der zuvor versendeten Nachricht beachten und fortführen. Dies ist lediglich dann nicht möglich, wenn zu einem Zeitpunkt alle noch aktiven TimeMaster neu gestartet werden, da dann die Zeitdifferenz zur letzten Referenznachricht nicht nachvollzogen werden kann.

Ein Reset aller TimeMaster geschieht jedoch nur bei einem Einbruch der Versorgungsspannung, und da das Telemetriesystem über eine einzige Versorgung gespeist wird, gäbe es bei einem Stromausfall keine aktiven Module für deren Betrieb die Referenznachricht erforderlich wäre.

3.2.4 TimeMaster-Transparenz

Identifikator

TimeMaster-Transparenz

Beschreibung

Der Wechsel des aktiven TimeMasters ist transparent zu halten, um die Software der Knoten nicht komplexer werden zu lassen.

Problembeschreibung

Falls der aktive TimeMaster ausfällt, wird die Referenznachricht von einem anderen TimeMaster versendet. Sie wird dadurch von einem anderen Modul versendet und trägt eine andere Identifikationsnummer. Die unterschiedlichen Nummern der TimeMaster sind für einen störungsfreien Betrieb des Busses und der TimeMaster notwendig.

Quelle

Siehe Abschnitt 2.4.8 Die Idee mehrerer TimeMaster.

Abnahmekriterium

Die TimeMaster-Identifikationsnummer der Referenznachricht soll für die Synchronisation der Knoten nicht relevant sein.

3.2.5 Flexibilität

Identifikator

Flexibilität

Beschreibung

Die verschiedenen Eigenschaften des verteilten Systems, wie etwa Komplexität, Hardware-Aufwand, Stromverbrauch und Sicherheitsgüte sollen variiert werden können. Dazu muss das Konzept so aufgebaut sein, dass es unabhängig von der Anzahl der integrierten TimeMaster ist.

Problembeschreibung

Unterschiedliche Anwendungen erfordern unterschiedlich hohe Sicherheitsanforderungen und ermöglichen unterschiedlich aufwendige Systeme. So soll das verwendete Bussystem auch in die folgenden Rennwagen des Hawks Racing Teams integriert werden, oder das bestehende System des Hawk07 soll um die Fortschritte dieser Arbeit erweitert werden. Da sich die integration von Fahrassistenzsystemen von Fahrzeug zu Fahrzeug ändert, ist es vorteilhaft die Anzahl der TimeMaster anzupassen, um oben genannte Faktoren zu optimieren.

Quelle

Siehe Abschnitt 2.4.8 Die Idee mehrerer TimeMaster.

Abnahmekriterium

Vollständige Flexibilität ist erreicht, wenn das Bussystem mit jeder beliebigen Zusammensetzung von TimeMastern lauffähig ist, und wenn dabei die Software der TimeMaster nicht verändert werden muss. Das bedeutet, TimeMaster #X wird mit der für #X konfigurierten Software betrieben, unabhängig davon, wie viele TimeMaster in das System integriert werden und welche IDs diese tragen.

3.2.6 Protokollkompatibilität

Identifikator

Protokollkompatibilität

Beschreibung

Nach dem Protokoll von Haase (2007) werden alle CAN-Nachrichten mit 8 von 8 möglichen Daten-Bytes verschickt, unabhängig davon wieviel Bytes Nutzdaten tatsächlich in der Nachricht enthalten sind. Das Byte 0 wird bereits durch das Protokoll belegt. In der Referenznachricht werden bisher die Bytes 1, 2 und 7 verwendet.

Problembeschreibung

Wie später in der Analyse deutlich wird, würde es Vorteile bringen, die Referenznachricht zu kürzen. Dafür müssten die genutzten Bytes zusammengerückt werden und die Referenznachricht könnte mit 4 Daten-Bytes versendet werden.

Jedoch soll versucht werden dies zu vermeiden, um die Kompatibilität mit alten Komponenten zu gewährleisten. Zudem könnten die momentan ungenutzten 4 Bytes in einer zukünftigen Entwicklung praktisch oder erforderlich sein.

Quelle

Das bisherige Protokoll ist beschrieben in Haase (2007, 4.2.2). In Anhang A auf Seite 72 ist die seitdem durch andere Entwicklungen erweiterte Version angehängt. In dieser sind alle vergebenen Identifizierer, Adressen und Datenbytes aufgelistet.

Abnahmekriterium

Protokollkompatibilität wird erhalten, wenn für die Integration der TimeMaster-Redundanz keine bereits festgelegten Identifizierer, Adressen oder Datenbytes verändert werden.

Erforderlichkeit

Nicht vollständig erforderlich, da die Integration der TimeMaster-Redundanz über der vollständigen Lauffähigkeit von alten Modulen steht.

3.2.7 Regelanforderungen

Identifikator

Regelanforderungen

Beschreibung

Der Rennwagen muss den unten aufgeführten Regelwerken entsprechen, sonst ist die Teilnahme am Event nicht möglich.

Quelle

Siehe die internationalen Formula Student Regeln SAE Rules 2008 (2007) und die deutsche Ergänzung für den Formula Student Germany Wettbewerb in Hockenheim (FSG Rules 2008, 2007).

Abnahmekriterium
Sämtliche Regeln müssen beachtet werden.

3.3 Zusammenfassung der Anforderungen

In Tabelle 3.1 sind die Anforderungen zusammengefasst.

Anforderung	Erforderlichkeit
Verfügbarkeit	erforderlich
Verzögerungsfreiheit	erforderlich
Zykluskonsistenz	erforderlich
TimeMaster-Transparenz	erforderlich
Flexibilität	erforderlich
Protokollkompatibilität	nicht vollständig erforderlich
Regelanforderungen	erforderlich

Tabelle 3.1: Übersicht der Anforderungen und ihrer Erforderlichkeit

3.4 Randbedingungen

Im Folgenden werden die Umstände aufgezeigt, unter denen diese Arbeit entstanden ist.

3.4.1 Termine

Im Jahr 2008 nimmt das Hawks Racing Team an den Wettbewerben in Silverstone (England) und Hockenheim (Deutschland) teil. Diese dauern knapp eine Woche und beginnen am 09.07.2008 bzw. am 06.08.2008. Zu diesen Terminen soll das System lauffähig sein.

Begonnen wurde mit dieser Arbeit im März 2008.

3.4.2 Zusammenarbeit im Team

Bei einem Projekt wie dem Rennwagen des Hawks Racing Teams, bei dem viele Baugruppen unterschiedlicher Fachrichtung an einem Produkt arbeiten, ist es immer wieder erforderlich, sich mit den anderen Baugruppen abzusprechen. Die Entwicklung dieser Arbeit erforderte die Zusammenarbeit mit den folgenden zwei Baugruppen:

3.4.2.1 Baugruppe Motor

Mit der Baugruppe für Motor und Antrieb wurde diskutiert, welche Möglichkeiten es zur Schlupferkennung und -regelung gibt, und welche sich davon in dem Hawk08 umsetzen lassen.

3.4.2.2 Baugruppe Elektrik

In diesem Jahr hat die Baugruppe Elektrik die Integration der Mikrocontroller-Module übernommen. Mit dieser Baugruppe wurde die Integration eines zweiten TimeMaster-Moduls sowie des ASR-Moduls besprochen.

3.4.3 Testumgebung

Da sich der Rennwagen selbst zur Entwicklungszeit dieser Arbeit noch in der Fertigungsphase befand, war ein Testen am endgültigen Objekt erst sehr spät möglich. Die Redundanz der TimeMaster kann unabhängig vom Fahrzeug im Labor entwickelt und getestet werden.

Auch die Motorsteuerung, siehe dazu den Abschnitt 4.2.3 Motorsteuerung, steht als Duplikat zur Verfügung, sodass die Fähigkeiten und Eigenschaften dieser bezüglich Regel- und Kommunikationsmöglichkeiten getestet werden können.

3.4.4 Testfahrten

Um den Rennwagen in Bewegung testen zu können, muss dieser zu einem geeigneten und viele Kilometer entfernten Gelände gebracht werden, da auf dem Gelände der Hochschule für Angewandte Wissenschaften Hamburg keine Möglichkeit für größere Testfahrten besteht. Eine solche Testfahrt ist zudem aufwendig, da mehrere Teammitglieder für den Transport und die Beaufsichtigung organisiert werden müssen.

4 Analyse

In diesem Kapitel werden die Lösungswege diskutiert, die zu einem System führen könnten, das den Anforderungen des vorigen Kapitels genügt. Einer dieser Wege wird im anschließenden Kapitel der Konzeption ausgearbeitet und im Kapitel der Realisierung umgesetzt.

4.1 Erkennen von Schlupf

Es gibt mehrere Techniken der Schlupferkennung, die sich in der Komplexität und Effizienz unterscheiden, und somit für unterschiedliche Einsätze interessant sind.

4.1.1 Raddrehzahlvergleich angetriebener und nichtangetriebener Räder

Der Raddrehzahlvergleich mehrerer Achsen ist die simpelste Methode zur Schlupferkennung. Sie beruht darauf, dass ein nichtangetriebenes Rad ohne Schlupf auf dem Untergrund mitläuft und die tatsächliche Geschwindigkeit des Fahrzeuges angibt. Hierbei wird nur die Raddrehzahl der angetriebenen Räder und die von mindestens einem nichtangetriebenen Rad benötigt. Daraus lässt sich mit der Drehzahl ω_0 eines mitlaufenden Rades für das Antriebsrad mit der Drehzahl ω der prozentuale Antriebsschlupf S berechnen:

$$S = \frac{\omega - \omega_0}{\omega}$$

Es ist jedoch zweckmäßig die Drehzahlen von beiden mitlaufenden Rädern auszuwerten, da sich diese bei der Kurvenfahrt unterscheiden.

4.1.2 Differential der Raddrehzahl

Die Methode des Raddrehzahl-Differentials benötigt nur die Raddrehzahl des Antriebsrades, erfordert aber weitere Kenntnisse über Räder und Fahrzeug. Hierbei wird das Differential

über die Raddrehzahl gebildet, also die Veränderung der Raddrehzahl über die Zeit betrachtet. In der einfachsten Variante vergleicht man den aktuellen Wert ω_t mit dem vorigen ω_{t-d}, wobei d die zeitliche Differenz der Messungen in Sekunden ist.

Lässt man den Schlupf außen vor, sind die Raddrehzahl und die Fahrzeuggeschwindigkeit linear voneinander abhängig. Es gibt also eine Funktion

$$f(\omega) = v$$

mit der die Raddrehzahl in die Geschwindigkeit v umgerechnet werden kann. Differenziert man die Geschwindigkeit über die Zeit, erhält man die Beschleunigung, die dem Fahrzeug widerfährt:

$$a_t = \frac{f(\omega_t) - f(\omega_{t-d})}{d}$$

Es gibt in Abhängigkeit von Fahrzeugtyp und Bodenhaftung ein a_{max}, welches die maximale erreichbare Beschleunigung angibt. Diese liegt etwa bei $1g$. Erreicht das berechnete a_t einen Wert über a_{max}, kann die Geschwindigkeit des angetriebenen Rades nicht der tatsächlichen Fahrzeuggeschwindigkeit entsprechen, dies weist auf außerordentlichen Schlupf hin. Da bei einem durchdrehendem Rad der Werte schlagartig auf Vielfache von a_{max} steigt, muss diese Schwelle zur Regelung nicht besonders genau bestimmmt werden, sodass $2 * a_{max}$ als Schwelle gewählt werden kann.

4.1.3 Vergleich der Schlupferkennungsmethoden

Die beiden oben genannten Methoden unterscheiden sich nach außen hin nur geringfügig. Die erste Methode, der Vergleich angetriebener und nichtangetriebener Räder, benötigt im Gegensatz zur anderen Methode, dem Differential der Raddrehzahl, auch an den vorderen, nichtangetriebenen Rädern Drehzahlsensoren. Da diese aber ohnehin eingebaut werden, um dem Fahrer eine verlässlichere Geschwindigkeitsangabe geben zu können, wirkt sich dies nicht als Nachteil aus.

4.2 Regeln von Schlupf

Die Antriebsschlupfregelung ist dauerhaft bereit, optional ist sie vom Fahrer abschaltbar. Die Regelung ist im normalen Fahrzustand passiv und greift bei Schlupferkennung autonom ein. Hierzu gibt es zwei Grundprinzipien mit mehreren Möglichkeiten, die sich weitestgehend

kombinieren lassen: die Regelung mithilfe des Bremssystems, einer elektronischen Drosselklappe oder der Motorsteuerung. Jedoch schränken die tatsächlichen Gegebenheiten die Möglichkeiten ein, wie die folgende Auflistung zeigt.

4.2.1 Bremseneingriff

Der direkteste Weg, ein durchdrehendes Rad zu verlangsamen, ist die vorsichtige Betätigung der entsprechenden Bremse. Bei dieser Methode ist das Anfahren mit schweren Fahrzeugen auf Eis oder im Sand besser als bei Regelung durch Motoreingriff, da nur das durchdrehende Rad gebremst wird, ohne die Leistung des Motors für alle Räder unter ein benötigtes Maß sinken zu lassen.

Die Nachteile sind der komplizierte und das Gewicht des Rennwagens erhöhende Eingriff ins Bremssystem, da ein elektronisch betätigter Druckzylinder in die Bremskreisläufe integriert werden muss. Außerdem besteht die Gefahr der Bremsenüberhitzung, wenn die Regelung über lange Zeit aktiv ist, etwa bei Fahrt mit zuviel Gas durch tiefen Sand.

Da der Rennwagen des Hawks Racing Teams jedoch nie auf Eis oder durch tiefen Sand bewegt werden wird, sind diese Punkte keine Kriterien. Da außerdem jedes zusätzliche Kilo unerwünscht ist, ist der Bremseneingriff keine akzeptable Lösung.

4.2.2 Elektronische Drosselklappe

Um die Kraft des Motors zu drosseln, wird die Gasgemisch-Zufuhr reduziert, so als würde der Fahrer den Fuß vom Gaspedal nehmen. Damit die Antriebsschlupfregelung in die Drosselklappe eingreifen kann, ist ein elektronisches Gaspedal erforderlich. Dies bedeutet, dass es keine direkte mechanische Verbindung mehr zwischen Gaspedal und Drosselklappe gibt, sondern stattdessen ein Sensor am Gaspedal die gewünschte Stellung auf elektronischem Wege an einen Aktor an der Drosselklappe übermittelt. In diese Übertragung kann nun das ASR-Modul zwischengeschaltet werden, sodass es den Wunsch des Fahrers unterdrücken und die Drosselklappe verengen kann.

Nach den internationalen Formula Student Regeln SAE Rules 2008 (2007) und der deutschen Anpassung für den Formula Student Germany Wettbewerb in Hockenheim (FSG Rules 2008, 2007) muss es zwischen Gaspedal und Drosselklappe eine zuverlässige mechanische Verbindung geben.

> „3.5.4.2 Throttle Actuation
>
> The throttle must be actuated mechanically, i.e. via a cable or a rod system. The use of electronic throttle control (ETC) or “drive-by-wire” is prohibited.

> The throttle cable or rod must have smooth operation, and must not have the possibility of binding or sticking. [...]“ (SAE Rules 2008, 2007, S. 50)
>
> „3.5.4.2 Drosselklappenantrieb
>
> Die Drosselklappe muss mechanisch betätigt werden, z. B. über ein Seil- oder Stangensystem. Der Einsatz von elektronischem Gaspedal oder “Drive-by-Wire” ist verboten.
>
> Das Seil oder die Stange muss leichtgängig arbeiten, ohne dass die Gefahr des Klemmens oder Feststeckens besteht. [...]“ (vgl. SAE Rules 2008, 2007, S. 50)

Eine elektronische Übertragung ist nicht erlaubt, vermutlich aus Sicherheitsbedenken. Damit könnte diese Art der Schlupfregelung nur zu privaten Versuchen in den Rennwagen integriert werden. Da aber die erfolgreiche Teilnahme an den Events im Vordergrund steht, sind andere Methoden dieser vorzuziehen.

4.2.3 Motorsteuerung

Die eingesetzte Motorsteuerung ist von Walbro[1] und ist wie der Motor einem Motorrad entnommen. Sie wird wie der Motor zum zweiten Mal in einem Rennwagen des Hawks Racing Teams eingesetzt.

Abbildung 4.1: Motorsteuerung ECUA von Walbro

Mit der Motorsteuerung (siehe Abbildung 4.1) lassen sich verschiedene Kennfelder konfigurieren sowie Statuswerte wie Motordrehzahl und -temperatur auslesen. Als Schnittstelle bietet die Motorsteuerung RS232 mit 38400 bit/s sowie K-Line[2]. Da K-Line mit 9600 bit/s

[1] Walbro: http://www.walbro.com
[2] K-Line oder K-Leitung: bidirektionaler Eindrahtbus für die Datenübertragung in der Automobiltechnik

jedoch zu langsam ist um die Daten in der geforderten Geschwindigkeit transportieren zu können, wird ausschließlich RS232 benutzt.

Eine Motorsteuerung bietet somit verschiedene Möglichkeiten, in den Antrieb einzugreifen und damit den Schlupf zu regeln. Jedoch ist mit der verwendeten Motorsteuerung nicht alles möglich, wie die folgenden Unterabschnitte zeigen.

4.2.3.1 Motorsteuerung: Einspritzung aussetzen

Der verwendete Motor von Kawasaki hat eine Saugrohreinspritzung mit elektronisch durch die Motorsteuerung geschalteten Einspritzventilen. Diese spritzen das Benzin in die angesaugte Luft, bevor dieses Gemisch in die Brennkammer geleitet wird. Durch Deaktivierung dieser Einspritzventile würde der Motor mangels Treibstoff keine Kraft mehr entwickeln.

Da die verwendete Motorsteuerung jedoch keine Möglichkeit dazu bietet, auf Befehl von außen die Einspritzung zu übergehen, kann diese Methode nicht eingesetzt werden.

4.2.3.2 Motorsteuerung: Zündung aussetzen

Durch das Aussetzen der Zündimpulse wird das explosive Benzin-Luft-Gemisch nicht gezündet und keine Energie umgesetzt. Die Motorsteuerung bietet diese Möglichkeit, da Motor und Motorsteuerung von einem Motorrad stammen. Dort wird die Zündunterbrechung (engl. ignition cut) verwendet, um beim Hochschalten die Kupplung nicht betätigen zu müssen, da ohne Zündung keine große Kraft auf das Getriebe wirkt.

Dass das Gemisch den Motor unverbrannt verlässt, hat aber Nachteile: Das Benzin verstopft den Katalysator und wird in die freie Luft ausgestoßen. Da bei der Antriebsschlupfregelung der Regelzeitraum mit mehreren Sekunden länger ist als ein Schaltvorgang von etwa einer Zehntelsekunde, ist dies keine von der Motorbaugruppe des Teams akzeptierte Lösung.

4.2.3.3 Motorsteuerung: Zündwinkel verstellen

Desweiteren gibt es die Möglichkeit auch während des Betriebes in der Motorsteuerung die Zündwinkel zu verändern. Das Gemisch im Brennraum wird nicht genau dann gezündet wenn der Kolben den oberen Totpunkt erreicht hat, sondern kurze Zeit vorher. Die Verbrennung des Gemisches besitzt eine nicht zu vernachlässigende Ausbreitungsgeschwindigkeit.

Im Idealfall ensteht die größte Verdichtung, wenn sich der Kolben kurz hinter dem oberen Totpunkt befindet, da so die größte Kraft über den Kolben auf die Kurbelwelle wirken kann.

Da sich während der Explosionsausbreitung der Kolben weiterbewegt, muss das Gemisch entsprechend früh gezündet werden. Der Zündzeitpunkt wird aus Sicht der Kurbelwelle in Grad vor dem oberen Totpunkt angegeben und hängt u. a. auch von der Motordrehzahl ab.

4.2.4 Vergleich der Schlupfregelungsmethoden

Den Vergleich nach den bereits in den vorigen Abschnitten angesprochenen Kriterien der Schlupfregelungsmethoden zeigt folgende Tabelle 4.1.

Regelungsmethoden	Kriterien		
	Integrierbar	Hawks-Interessen	Regelkonform
Bremseneingriff	ja	nein, Gewicht	ja
elektronische Drosselklappe	ja	ja	nein, s. o.
Einspritzung aussetzen	nein, ECU	ja	ja
Zündung aussetzen	nein, ECU	ja	ja
Zündwinkel verstellen	ja	ja	ja

Tabelle 4.1: Vergleich der Schlupfregelungsmethoden

Daraus ergibt sich, dass die Methode der Zündwinkelverstellung die einzige umsetzbare Methode ist.

4.3 Mögliche Ausfallursachen des Bussystems

Es gibt folgende Ursachen für einen Ausfall eines Bussystems, wobei die letzte nur bei einem zeitgesteuerten Bus vorkommen kann.

4.3.1 Zerstörung der physikalischen Verbindung

Ursachen für eine tote Leitung können sich lösende Steckverbindungen oder durchtrennte Kabel sein. Durch verlässliche Lötstellen, arretierbare und witterungsbeständige Kabelverbindungen sowie eine bedachte Verlegung der Leitungen ohne mechanische Beanspruchung ist ein solcher Ausfall sehr unwahrscheinlich.

Um dennoch einen solchen Ausfall zu überstehen, müsste man ein redundantes Bussystem integrieren. Der Aufwand ist hierbei jedoch enorm, da dies entsprechende Hardware erfordert. Zudem würde das Fahrzeuggewicht durch die doppelte Verkabelung steigen.

Diese Ursache wird in dieser Arbeit nicht näher behandelt.

4.3.2 Dauerhafte Signalstörung durch externe Einflüsse

Hohe Ströme, hohe Frequenzen (beispielsweise von Sprach- oder Datenfunk) sowie hohe Spannungen (z. B. von der Zündanlage) können in ungeschützten elektronischen Geräten Störungen verursachen. Das Themengebiet der elektromagnetischen Verträglichkeit (EMV) befasst sich mit diesen unerwünschten Effekten zwischen Geräten.

Simon Schuckert (2007) hat in Zusammenhang mit der Entwicklung des Systems eine EMV-Analyse durchgeführt und festgestellt, dass sich Einflüsse durch die Motorelektrik vermeiden lassen, wenn das System elektromagnetisch abgeschirmt wird. Das bedeutet im Wesentlichen, Gehäuse aus Aluminium oder Kohlenstoff statt Plastik sowie geschirmte Signalleitungen zu verwenden. Die Fehlerquelle der elektromagnetische Störungen ist bei Bedacht der Fahrzeugkomponenten und der Position der Geräte also sehr unwahrscheinlich (vgl. Schuckert, 2007, S. 25).

4.3.3 Dauerhafte Signalstörung durch defekte Bus-Controller

Bei einem Bussystem besteht wegen der festen Verkabelung das Problem, dass die Funktion des Busses durch einen fehlerhaften Teilnehmer fortwährend gestört wird (vgl. Etschberger, 2000, S. 76). „Insbesondere das bei CAN angewandte Prinzip der Fehlersignalisierung beinhaltet diese Gefahr, da ein defekter Teilnehmer ständig Fehlerflags generieren könnte“ (Etschberger, 2000, S. 76).

Daher ist im CAN-Protokoll für jeden CAN-Controller ein komplexer Fehlerzähler vorgeschrieben. Dieser erhöht sich mit jedem erkannten Fehler, wobei vom CAN-Controller selbst entdeckte Fehler mehrfach gewertert werden. Dies beruht auf dem Verdacht, dass selbst entdeckte Fehler auch selbst verschuldete sind. Sendefehler und Empfangsfehler werden getrennt gezählt und unterschiedlich stark gewichtet (vgl. Bagschik, 1999, S. 96).

Der Fehlerzähler besitzt zwei Schwellen: Ist die erste überschritten, sendet der Controller keine dominanten, sondern lediglich rezessive (genannt passive) Error-Frames. Bei Erreichen der zweiten Schwelle schaltet sich der Controller ab, um das Risiko, den Bus weiterhin außer Betrieb zu setzen, auszuschließen. Eine Reaktivierung erfolgt nicht autonom: hierfür muss der Mikrocontroller den CAN-Controller zurücksetzen (vgl. Bagschik, 1999, S. 94).

Wenn eine Nachricht erfolgreich versendet wurde, wird der Fehlerzähler wieder heruntergezählt. Dadurch wird der Controller erst bei sehr starker Häufung der Fehler deaktiviert, und nicht bei seltenen Fehlern im monatelangen Einsatz. Defekte Bus-Controller sind daher auf

lange Sicht keine Ursache für einen Ausfall des gesamten Bussystems. Lediglich die zum betroffenen Controller gehörende Komponente ist dann funktionsunfähig.

4.3.4 Ausfall der Synchronisation eines zeitgesteuerten Busses

Wie im Abschnitt 2.4.5.1 Synchronisation beschrieben ist die Synchronisation mithilfe des TimeMasters zwingend erforderlich für den geregelten Ablauf der Buskommunikation. Der TimeMaster im bestehenden System könnte durch Material- oder Fertigungsfehler ausfallen, was einen Totalausfall der Kommunikation bewirken würde.

4.3.5 Software-Fehler

Theoretisch sind auch Software-Fehler möglich, bei denen ein TimeMaster oder ein beliebiger anderer Busteilnehmer ein unkontrolliertes Verhalten aufweist. Dabei könnten Nachrichten mit falschen, andere Busteilnehmer verwirrende Identifizierern versendet werden, oder die Busauslastung könnte durch dauerhaftes Senden auf ein Maximum steigen und große Verzögerungen verursachen.

Da die intensive Überprüfung der Software als Teil des Software Engineerings außerordentlich komplex ist, kann dies nicht in dieser Arbeit behandelt werden.

4.4 Ausnahmereaktionen des Bussystems

Die Ausfallursache, die in dieser Arbeit behandelt wird, ist die unterbrochene Synchronisation der Knoten. Es sind folgende Möglichkeiten denkbar, um auf das Ausbleiben der Referenznachricht zu reagieren:

4.4.1 Vorsichtig

Das Senden von Nachrichten wird eingestellt, bis die nächste Referenznachricht empfangen wird. Der Nachteil ist, dass so lange keine Daten übertragen werden können, bis wieder eine Referenznachricht empfangen wurde, etwa weil der TimeMaster durch den Watchdog (vgl. 2.7) reaktiviert wurde.

Der theoretische Vorteil ist, dass eine Referenznachricht, die von dem durch einen Watchdog reaktivierten TimeMaster gesendet wird, nicht durch eine gerade versendete Nachricht verzögert werden kann. Da aber eine CAN-Nachricht mit weniger als 300 μs (bei 500 kbit/s

Bustakt, ohne Extended Frame) weitaus kürzer ist als ein Watchdog-Timeout (~16 ms bis 2 s, inkl. Start-Up), ist dieser Vorteil ohne große Bedeutung.

Für das Telemetriesystem ist diese Methode nicht geeignet, da sie der Anforderung eines unterbrechungsfreien Systems (siehe 3.2.2 Verzögerungsfreiheit) nicht gerecht wird.

4.4.2 Fallback

Es wird auf ereignisgesteuertes Verhalten zurückgeschaltet (engl. fall back). Ob dies überhaupt möglich ist, hängt von der Implementierung des TTCAN ab.

Ist die CAN-Hardware TTCAN fähig, übergibt die Software ihre Nachrichten zu unbestimmten Zeitpunkten an die CAN-Hardware, da diese den zeitlich korrekten Versand der Nachrichten regelt. Diese TT-Funktionalität könnte abgeschaltet werden, sodass die Nachrichten direkt nachdem die Software eine Nachricht übergeben hat, gesendet werden, sobald der Bus frei ist.

In dem bestehenden Telemetriesystem ist dies kaum möglich, da die Zeitsteuerung des Versendens von einem Software-Scheduler geregelt wird. Da dieser nicht, ohne ein Alternativprogramm parat zu haben, abgeschaltet werden kann, würden die Nachrichten nach wie vor zum (aus der Sicht des Mikrocontrollers) richtigen Zeitpunkt versendet werden.

Der Vorteil dieses Verfahrens gegenüber dem folgenden Toleranten ist, dass versucht wird Nachrichten eher als geplant über den Bus zu senden, und so trotz Verzögerungen durch Busbelegung noch im Rahmen liegen können.

Dieses Verhalten könnte man in zwei Varianten betreiben:

- Optimistisch:
  Sämtliche Teilnehmer versenden die reguläre Anzahl Nachrichten. Je mehr Nachrichten versendet werden, desto größer ist das Risiko der Nachrichtenverzögerung.
- Priorisiert:
  Die meisten Teilnehmer stellen das Senden ein, nur die notwendigsten versenden ihre Nachrichten. Dadurch können Aufgaben mit hoher Priorität, beispielsweise das Informieren des Fahrers über den Motorzustand (Temperaturen, Öldruck), weiterhin ausgeführt werden.

Wie groß die zu erwartenden Verzögerungen sind, hängt von dem Anteil vergebener Zeitfenster und von der Reservezeit ab. Mit Reservezeit ist die Differenz zwischen Zeitfensterdauer und Nachrichtendauer gemeint, also die durch die Auslegung der Slots bedingte ungenutzte Buszeit.

4.4.3 Tolerant

Die Busteilnehmer behalten ihr zeitgesteuertes Verhalten bei, zählen mit ihrer lokalen Uhr weiter und versenden ihre Nachrichten zu dem ihrer Zeitzählung nach korrekten Zeitpunkt. Da die verteilten Uhren früher oder später auseinanderlaufen werden, werden periodisch Überschneidungen der Zeitfenster entstehen und wieder verschwinden, sodass auf lange Sicht ungewünschte Nachrichtenverzögerungen entstehen. Auf kurze Sicht ist es aber möglicherweise eine passable Lösung, um einen kurzzeitigen Ausfall des TimeMasters zu überbrücken.

Das Problem hierbei ist, dass der TimeMaster, wenn er wieder funktionsbereit ist, entweder einen sprunghaften Neubeginn des Zyklus bewirkt, oder sich durch wages Einschätzen der Busnachrichten zum laufenden Zyklus synchronisieren muss.

- Auch hier gibt es die Varianten optimistisch und priorisiert, mit denselben Eigenschaften wie beim vorherigen Verhalten.

4.4.4 Reserve-TimeMaster

Falls die Referenznachricht des TimeMasters ausbleibt, springt sofort ein Reservemodul ein und übernimmt die Synchronisation des Busses. Durch mindestens einen Reserve-TimeMaster entfällt das Problem aus dem vorigen Verhalten, wonach der ausgefallene TimeMaster keine sichere Möglichkeit hat sich zu resynchronisieren und den Zyklus an der richtigen Stelle fortzusetzen.

Der große Vorteil dieses Verfahrens ist, dass der Buszyklus nicht unterbrochen wird, und die Busteilnehmer nicht wissen (müssen), welcher TimeMaster aktiv ist. Ein Nachteil ist lediglich, dass entweder einer oder mehrere vorhandene Knoten um die TimeMaster-Funktionalität erweitert oder hierfür neue Knoten hinzugefügt werden müssen.

Diese Strategie wird in dieser Arbeit umgesetzt.

4.4.5 Vergleich der Ausnahmereaktionen

Die folgende Tabelle 4.2 auf der nächsten Seite zeigt die genannten Ausnahmereaktionen in der Übersicht.

Aus der Tabelle ist ersichtlich, dass die Strategie der Reserve-TimeMaster die einzige ist, bei der der Bus geordnet und deterministisch bleibt, und damit in dieser Arbeit umgesetzt wird. Bei den anderen Methoden bleibt die Kommunikation stehen oder gerät ohne TimeMaster zwangsläufig ins Stocken.

TTCAN Anforderungen	Reaktionen					
	Vorsichtig	Fallback		Tolerant		Reserve-
		Optim.	Prior.	Optim.	Prior.	TimeMaster
Verfügbarkeit	nein	ja	ja	ja	ja	ja
Verzögerungsfreiheit	nein	nein	nein	nein	nein	ja*
Zykluskonsistenz	nein	kein Zyklus, dann Sprung				ja
TimeMaster-Transparenz	1 TM	kein TM		kein TM		ja*
Flexibilität	1 TM	kein TM		kein TM		ja
Protokollkompatibilität	ist aktuell	ja	ja	ja	ja	ja*

Tabelle 4.2: Übersicht der Ausnahmereaktionen und ihre Erfüllung der Anforderungen (bei maximaler Buslast, d. h. die Varianten Optimistisch und Priorisiert sind equivalent) (* hängt von der Implementierung ab)

4.5 Bisher implementiertes Verhalten

Das bislang implementierte Verhalten entspricht dem vorsichtigen, es ist aber möglich dieses toleranter zu gestalten:

> „Bleibt eine TM [eine Referenznachricht, F.K.] aus, so wird das vom TTC-System erkannt und es kann dann eine entsprechende Notfall-Funktion ausführen. Im einfachsten Fall ist dies ein Reset des Moduls, es sind aber auch andere Funktionen möglich. Denkbar ist auch eine Variante, bei der das Modul noch eine gewisse Zeit weiterarbeitet und seine Nachrichten weiterhin versendet, bis z. B. ein neuer TimeMaster die Kontrolle übernimmt. Dabei sollte die Anzahl der Zyklen so gering wie möglich gehalten werden.“ (Schuckert, 2007, 3.2.4)

Man könnte also ein Ausbleiben tolerieren und erst beim zweiten Fehlen den Reset des Moduls ausführen. Dies ist erforderlich, falls es nicht möglich ist, dass ein Ausfall der Referenznachricht noch im selben Zyklus bzw. im selben Busslot erkannt und ersetzt werden kann.

5 Konzeption

In diesem Kapitel wird ein Konzept aus der Analyse entworfen, welches die Anforderungen und Randbedingungen erfüllt.

5.1 Antriebsschlupfregelung

Wie in der Analyse unter 4.2.4 Vergleich der Schlupfregelungsmethoden beschrieben wurde, ist unter den vorgegebenen Umständen die Zündwinkelverstellung der einzige integrierbare Aktor der Antriebsschlupfregelung. Siehe dazu den Abschnitt 4.2.3 Motorsteuerung.

5.1.1 Konfiguration über RS232

Um die Zündwinkel zu verstellen, können entsprechende Kommandos über eine RS232-Schnittstelle an die Motorsteuerung gesendet werden. Hierfür wird das bestehende ECU[1]-Gateway modifiziert. Dieses kommuniziert bereits mit der Motorsteuerung um deren Sensordaten auszulesen. Dabei wird zu Beginn die Verbindung mit 4 Befehlen initialisiert, woraufhin die Motorsteuerung ununterbrochen Daten sendet. Die Tx-Leitung des ECU-Gateways in Richtung Motorsteuerung ist nach der Initialisierung also frei für Zündwinkel-Kommandos.

5.1.2 Zündwinkel-Kommando

Die Konfigurations-Kommandos für die Motorsteuerung bestehen aus 12 Zeichen und haben folgendes Format: Nach dem Befehltrennzeichen „:“ kommt ein für diese Fälle fixer Teil „C00FFEC0“, der sich aus Formatbeschreibung, Zieladresse, Ursprungsadresse und Nachrichtenlänge zusammensetzt. Darauf folgt die einstellige hexadezimale Einstellungskennung, der zweistellige hexadezimale zu setzende Wert und eine auf ein Byte gekürzte Prüfsumme der zuvor aufgezählten Zeichen (vgl. Zimmermann und Schmidgall, 2006, S. 28).

[1] Engine Control Unit, zu dt. Motorsteuerung

Da der Zündwinkel für jeden der 4 Zylinder separat konfiguriert werden muss, müssen für eine synchrone Veränderung der Werte 4 Nachrichten versendet werden. Die entsprechenden Kennungen der Zündwinkeleinstellungen sind 4 für den 1. Zylinder bis 7 für den 4. Zylinder.

Der in dem Kommando angegebene Wert wird als Versatz zum Normalwinkel des Kennfeldes mit 0,25° pro Bit interpretiert. Wenn die Antriebsschlupfregelung die Zündwinkel nicht beeinflussen soll, beispielsweise weil kein weiterer Regelbedarf besteht, wird 0 als Wert in die Nachricht geschrieben.

5.1.3 Zeitverhalten

Ein Aktor besitzt die Eigenschaft der maximalen Regelfrequenz. Diese besagt, wie oft pro Zeitintervall der Aktor einen neuen Befehl umsetzen kann. Bei einem System wie der Antriebsschlupfregelung ist dies von Bedeutung, da mehrere Male pro Sekunde die Kraft des Motors angepasst werden muss, um den optimalen Schlupf (siehe dazu 2.2.4) am Rad zu bewirken.

5.1.3.1 Schnittstelle der Motorsteuerung

Die serielle Schnittstelle der Motorsteuerung wird mit 34800 bit/s betrieben. Bei einem Übertragungsmodus von 8-N-1 (d. h. 9 Bit pro Zeichen) ergibt sich bei einem Kommando von 12 Zeichen eine Dauer von knapp über 3.103 Millisekunden pro Kommando:

$$\frac{1}{34800\frac{bit}{s}} \times 9\frac{bit}{Zeichen} \times 12\frac{Zeichen}{Kommando} = \frac{108}{34800}\frac{s}{Kommando} = 3.103\,\frac{ms}{Kommando}$$

So kommt es zu einer maximalen Kommandofrequenz von 322 Hertz:

$$\frac{1}{\frac{108}{34800}\frac{s}{Kommando}} = \frac{34800}{108}\frac{Kommando}{s} = 322.\bar{2}\,\frac{Kommando}{s}$$

Da für eine komplette Zündwinkelanpassung alle 4 Zündwinkel über eine eigene Nachricht gesetzt werden müssen, sinkt die theoretische Regelfrequenz auf $80.\bar{5}$ Hertz.

5.1.4 Zustände der ASR

Die Antriebsschlupfregelung soll sich stets in einem der folgenden Zustände befinden:

inaktiv
Die Regelung ist deaktiviert, z. B. durch einen Schalter im Cockpit, den der Pilot betätigen kann, wenn sich die Antriebsschlupfregelung nicht vorteilhaft verhält. Gegebenenfalls werden die Sensoren zur Diagnose ausgewertet ohne in das Fahrzeug einzugreifen.

passiv
Die Regelung ist regelbereit, wertet die Sensoren aus, aber greift momentan nicht in das Fahrzeug ein, da kein unerwünschter Schlupf vorliegt.

aktiv
Die Regelung ist regelbereit, wertet die Sensoren aus und greift in das Fahrzeug ein, um den Schlupf zu regeln.

regelbegrenzt
Die Regelung ist regelbereit, wertet die Sensoren aus aber greift nicht in das Fahrzeug ein, da äußere Umstände dies zur Zeit nicht zulassen. Ein solcher Umstand könnte z. B. die Temperatur der Bremsbeläge (siehe dazu 4.2.1 Bremseneingriff) sein, die erst wieder unter eine Schwelle sinken muss, damit die Antriebsschlupfregelung ohne Schäden für das Fahrzeug regeln kann (siehe dazu 3.1.3 Schadensfreiheit).

fehlerhaft
Die Regelung hat einen Fehler festgestellt und ist inaktiv.

5.1.5 ASR-Steuernachricht

Um das ECU-Gateway als Aktor der Antriebsschlupfregelung von der Logik trennen zu können, wird eine Steuernachricht definiert. Diese gibt an, ob die Regelung aktiv ist und in welchem Maße die Regelung eingreift. Steuernachrichten sind im CAN-Protokoll von Haase (2007) unter Control-Nachrichten definiert.

Neben dem Aktor kann diese Steuernachricht auch vom Live Monitoring, der Datenaufzeichnung und dem Lenkraddisplay ausgewertet werden, etwa um den Fahrer zu informieren.

5.1.6 Modifikation des ECU-Gateways

Das ECU-Gateway empfängt diese Steuernachricht und sendet, wenn die Antriebsschlupfregelung aktiv sein soll, den gewünschten Wert an die Motorsteuerung. Bleibt diese Nachricht aus oder die Antriebsschlupfregelung soll nicht aktiv sein, wird der neutrale Wert 0 gesendet, siehe dazu 5.1.2 Zündwinkel-Kommando.

5.1.7 ASR-Modul

Das ASR-Modul übernimmt die Rechenlogik der Antriebsschlupfregelung. Durch den CAN-Bus besteht die Möglichkeit, sämtliche Sensoren in der Berechnung zu beachten. So können die verschiedenen angeführten Schlupferkennungsmethoden aus 4.1.3 integriert werden, ohne dass an der Verkabelung etwas geändert werden muss. In dem Programm des ECU-Gateways sind zudem nur noch begrenzte Rechenkapazitäten verfügbar. Ein separates ASR-Modul vereinfacht das Testen des Prototypen.

5.2 Zeitliche Bedingungen

Hier werden kurz die zeitliche Bedingungen der Nachrichtendauer und der Knotenzeitabweichung erläutert, da diese im Folgenden zu beachten sind. Das Zeitfenster für die Referenznachricht ist wie alle für eine Nachricht ausgelegten Zeitfenster 400 μs lang.

5.2.1 Nachrichtendauer

Nach dem Protokoll von Haase (2007) wurde festgelegt, dass jede versendete Nachricht 8 Daten-Bytes umfasst. Da nach der Anforderung 3.2.6 Protokollkompatibilität an diesem Protokoll festgehalten werden soll, müssen kürzere Nachrichten nicht beachtet werden. Durch die CAN-Spezifikation wird eine 8-Daten-Byte-Nachricht mit 111 bis 130 Bit übertragen (vgl. Zimmermann und Schmidgall, 2006, S. 36). Die Variation begründet sich in den Stuffbits, die gegebenenfalls während der Übertragung von der CAN-Hardware eingefügt werden, um spätestens nach 5 Bits einen Flankenwechsel zu bewirken.

Dies ist erforderlich, damit die empfangenden CAN-Controller sich auch nach dem Startbit ohne weitere Taktsignale an der Nachricht selbst resynchronisieren können. Bei der im Telemetriesystem festgesetzten Busgeschwindigkeit von 500 kbit pro Sekunde ergibt sich eine

Nachrichten-Übertragungszeit von 222 bis 260 μs:

$$\frac{1}{500kbit/s} \times 130bit = 260\mu s$$

5.2.2 Knotenzeitabweichung

Die maximal zu erwartende Knotenzeitabweichung ergibt sich aus der Basiszykluszeit und der Ungenauigkeit der Knoten-Taktquelle. Ein Basiszyklus dauert 10 ms, die als Taktquellen verwendeten Quartzoszillatoren haben eine Ungenauigkeit von maximal 100 ppm[2]. Die maximale Abweichung beträgt demnach:

$$10ms \times \pm 100 \div 100000 = \pm 10\mu s$$

5.3 Das Einspringen von TimeMastern

Wie im Abschnitt 2.4.8 Die Idee mehrerer TimeMaster beschrieben wurde, soll bei Ausbleiben der Referenznachricht ein beliebiger anderer TimeMaster einspringen und das Senden der Referenznachricht übernehmen. Das Problem hierbei ist, dass eine gewisse Zeit benötigt wird, um das Ausbleiben zu erkennen. Diese Zeit setzt sich hauptsächlich aus der Übertragungszeit der Nachricht und der möglichen Differenz der Knotenzeiten zusammen, und wird im Folgenden als Timeout bezeichnet.

Bei Ablauf des Timeouts, gemessen ab Beginn des Referenznachrichten-Busslots, sollte der Ersatz-TimeMaster die Versendung der Referenznachricht übernehmen. Diese Nachricht würde dann jedoch in den folgenden Busslot hineinragen, da der Slot mit 400 μs kleiner ist als zwei Nachrichten plus die eventuelle Knotenzeitabweichung:

$$2 \times 260\mu s + 10\mu s = 530\mu s > 400\mu s$$

5.3.1 Mögliche Strategien

Es stehen folgende Möglichkeiten zur Diskussion, um dieses Problem zu umgehen:

[2] ppm: parts per million, eine Notation für kleine Anteile

5.3.1.1 Bewusstes Übertreten der Slotgrenze

Es wird über den Slot hinaus gesendet. Die Ersatznachricht würde

$$530\mu s - 400\mu s = \underline{130\mu s}$$

des folgenden Slots belegen, in diesem wären dann noch

$$400\mu s - 130\mu s = \underline{270\mu s}$$

frei. Die Nachricht des folgenden Slots könnte also problemlos in diesem versendet werden. Effektiv würde lediglich die Referenznachricht verspätet versendet werden (dazu im Folgenden mehr), keine andere Nachricht wäre betroffen.

5.3.1.2 Frühzeitigeres Erkennen

Es wird ab Slotbeginn der Bus (bzw. die Rx-Leitung des CAN-Tranceivers) abgehört, um eine Aktivität zu detektieren. Falls die Leitung in den ersten 100 μs stets auf rezessivem Pegel liegt, findet keine Nachrichtenübertragung statt, und der Ersatz-TimeMaster kann seine Referenznachricht noch innerhalb des Slots versenden.

5.3.1.3 Stures Senden aller TimeMaster zur selben Zeit

Alle (Reserve-)TimeMaster verschicken synchron zum selben Zeitpunkt ihre Nachrichten. Dies scheint übertrieben, ist aber durchaus praktikabel. Den TimeMastern würde man unterschiedliche CAN-IDs geben, sodass die Arbitrierung des Busses (siehe dazu 2.5.1.2 Dominanter & rezessiver Pegel) die Filterung auf eine Nachricht übernimmt. Die TimeMaster, deren Nachrichten die Arbitrierung verloren haben, müssen dabei die Sendungswiederholung deaktivieren, sodass die anderen Module nur eine Referenznachricht empfangen.

Der große Vorteil dieser Methode ist, dass die Referenznachricht in jedem Falle – sofern es noch wenigstens einen aktiven TimeMaster gibt – pünktlich versendet und empfangen wird. Der Zeitpunkt des Empfangs ist wichtig für die Synchronisation der Knoten-Software.

Diese Strategie kann in dieser Arbeit jedoch nicht umgesetzt werden, da sich der verwendete Mikrocontroller AT90128CAN betreffend der Arbitrierung nicht vollständig an die TTCAN-Spezifikation (vgl. TTCAN, 2000) hält, siehe dazu 2.5.2 Der TTCAN-Fehler des CAN-Controllers.

5.3.2 Bewertung der Strategien

Die ersten beiden der drei im vorigen Abschnitt 5.3.1 genannten Möglichkeiten bleiben vorerst als mögliche Lösung bestehen, da die dritte, das frühzeitigere Erkennen (siehe 5.3.1.2), wie beschrieben nicht umgesetzt werden kann.

Die erste, das bewusste Übertreten der Slotgrenze (siehe 5.3.1.1), ist jedoch nur bei zwei TimeMastern möglich. Bei mehreren Reserve-TimeMastern müssten diese sequentiell aufeinander warten:

1. TM#2 wartet die Nachricht von TM#1 ab und springt ggf. für diesen ein.
2. TM#3 muss nicht nur die Nachricht von TM#1 abwarten, sondern auch die von TM#2, denn wenn TM#3 und TM#2 gleichzeitig ihre Ersatznachricht versenden (wollen), wäre eine Arbitrierung ohne Sendewiederholung der Überstimmten erforderlich. Dies ist jedoch wie in Abschnitt 2.5.2 beschrieben nicht möglich.
3. TM#4 muss entsprechend auf TM#1, TM#2 und TM#3 warten, bevor er seine Nachricht verschicken darf.
4. Der letzte mögliche TimeMaster #7 müsste entsprechend auf 6 Nachrichten mit 6 Knotenzeittoleranzen warten, also

   $$6 \times (260\mu s + 10\mu s) = \underline{1620\mu s},$$

   was wiederum knapp über 8 Busslots (1600 μs) liegt, die damit im Falle von 6 ausgefallenen TimeMastern belegt wären. Dies kollidiert mit der Anforderung der Protokollkompatibilität (siehe 3.2.6).

Damit genügt diese Methode der Anforderung 3.2.6 der Protokollkompatibilität nicht.

Die zweite Methode des frühzeitigeren Erkennens (siehe 5.3.1.2) dagegen lässt sich auf 7 TimeMaster erweitern. Diese müssen dann auch wie im vorigen Abschnitt beschrieben auf die Nachricht des jeweils vorhergehenden TimeMasters warten. Um dies in einem Busslot zu erreichen, müssen jeweils verschobene Abtastintervalle gesetzt werden:

Der TimeMaster #1 sendet seine Nachricht direkt ab Beginn des Busslots. Der TimeMaster #7 muss das Senden seiner Nachricht bis zum Ende des Busslots beendet haben, das bedeutet er muss spätestens

$$400\mu s - 260\mu s = \underline{140\mu s}$$

nach Slotbeginn mit dem Senden beginnen – falls er einspringen muss.

5.3.3 Konzeption der Strategie „frühzeitigeres Erkennen“

Für die TimeMaster #2 bis #7 bleiben also 140 μs für die Staffelung der Prüfintervalle. Das ergibt $23.\bar{3}\mu s$ für jeden TimeMaster. Bei der gegebenen Busgeschwindigkeit ergibt dies

$$500kBit/s \times 23.\bar{3}\mu s = \underline{11.\bar{6}Bit}$$

in einem solchen Intervall. Durch die CAN-Spezifikation (siehe 2.4.6 Der CAN-Bus) wird durch Stuffbits in dieser Zeit mindestens ein Flankenwechsel und damit mindestens ein messbares dominantes Bit vorkommen. Eine Anordnung der Prüfintervalle wie in Tabelle 5.1 ist denkbar.

TimeMaster	Prüfintervallbeginn	Prüfintervallende & Sendebeginn	Sendeabschluss
TM #1	-	0	260
TM #2	0	20	280
TM #3	20	40	300
TM #4	40	60	320
TM #5	60	80	340
TM #6	80	100	360
TM #7	100	120	380

Tabelle 5.1: Prüfintervalle der Reserve-TimeMaster
(in μs ab Busslot-Beginn)

Wenn in dem Prüfintervall mindestens ein dominantes Bit erscheint, wird der Bus momentan aktiv benutzt. Theoretisch gibt es dafür drei Ursachen:

- die Übertragung einer Referenznachricht eines anderen TimeMasters
- die Übertragung irgend einer anderen Nachricht
- die Übertragung eines Error-Frames

Da es sich um einen zeitgesteuerten Bus handelt, dürfen in diesem Busslot lediglich die Referenznachrichten versendet werden. Für den Fall, dass unglücklicherweise zu diesem Zeitpunkt ein Error-Frame den Bus belegt, sollten sämtliche Knoten toleranter reagieren und eine ausgefallene Referenznachricht ignorieren und erst bei der zweiten die jeweilige Notfallfunktion (im einfachsten Fall einen Reset) ausführen.

5.3.4 Diskussion in der Literatur

Die Firma Bosch, maßgeblich an der Entwicklung von CAN und TTCAN beteiligt, beschreibt in einem Dokument (TTCAN, 2000, S. 6), wie sie sich den TimeMaster-Wechsel vorstellt. Es werden die entscheidenden Fälle für Aus- und Eintritt eines höher und niedriger priorisierten TimeMasters aus dem bzw. in das System beschrieben:

Wenn eine Referenznachricht ausbleibt, sollen die Reserve-TimeMaster dies nach einer kurzen Wartezeit bemerken und ihre eigene Referenznachricht verschicken. Dabei sollen alle Reserve-TimeMaster gleichzeitig handeln, und durch die Bus-Arbitration wird der am höchsten priorisierte als nun aktiver TimeMaster gewählt.

> „During operation a missing reference message is recognized by all potential time masters within short latency. The latency is realized by a timeout. After this timeout is reached a potential time master starts sending the reference message [...]. The functionality of the time master is reestablished and the reference message is sent. Again, the **bitwise arbitration** of the standard CAN protocol decides among competing potential time masters.“ (TTCAN, 2000, S. 6)

Wenn ein TimeMaster mit einer niedrigeren ID als der des momentan aktiven TimeMasters mit dem Bus verbunden bzw. durch einen Watchdog resettet wird, soll er zum nächsten Basiszyklus seine Referenznachricht versenden. Die Arbitration würde den bisher aktiven TimeMaster dann degradieren.

> „Whenever a reference message with a lower priority is received, the potential time master [...] tries to become time master by sending its own reference message at the start of the next basic cycle. Due to higher priority it will win the **arbitration**.“ (TTCAN, 2000, S. 6)

Die Methode von Bosch setzt also an zwei Stellen auf die Bus-Arbitration. Wie in Kapitel 2.5.2 beschrieben ist dies jedoch mit dem benutzten Mikrocontroller nicht möglich.

5.3.5 Machbarkeitsuntersuchung zu dieser Lösung

Diese Methode bringt in der Umsetzung folgende Schwierigkeiten mit sich. Die kurzen Prüfintervalle erfordern eine zeitlich fein definierbare Software. Das Prüfen der Busleitung muss selbst realisiert werden, da es in der benutzten Hardware nicht bereits integriert ist.

5.3.5.1 Anforderungen dieser Lösung an die Software

Die Programme der Mikrocontroller basieren auf einem Scheduler (siehe 2.6 Scheduler), dessen Tick 200 μs beträgt, also weitaus größer ist als die geforderte Einheit von maximal 20 μs. Mit viel Aufwand ließe sich dieser Tick zwar verringern, jedoch ist dies keine passable Lösung: Die Interrupt Service Routine des Schedulers würde entsprechend zehnmal so oft aufgerufen werden und entsprechend Rechenzeit verbrauchen. Zudem würde diese feine Zeiteinteilung nur für die Referenznachricht gebraucht werden, für die restlichen Nachrichten und Funktionen der Module dagegen sind 200 μs optimal für die Slots des Bussystems.

Stattdessen sollte versucht werden, die zeitlichen Anforderungen unabhängig vom Scheduler zu koordinieren. Für den Scheduler gibt es so nach wie vor nur einen Task, der für die Versendung der Referenznachricht zuständig ist. Dieser Task ist aktiv, bis eine andere Referenznachricht registriert oder die eigene versendet wird. Die zeitliche Verzögerung kann dann über abgezählte NOPs[3] oder über eine bedingte Warteschleife mit Unterbrechung durch einen Timer-Interrupt geschehen. Dazu mehr in der Realisierung unter 6.2.1.1 Software-Delay.

5.3.5.2 Anforderungen dieser Lösung an die Hardware

In dem verwendeten Mikrocontroller gibt es keine direkte Möglichkeit, sich in quasi-Echtzeit (also in Zeitspannen weit unter einer Nachrichtenlänge) über den Zustand des Busses zu informieren. Es gibt einen Interrupt für erfolgreich empfangene Nachrichten sowie für aufgetretene Fehler, jedoch keinen für den Beginn einer Nachrichtenübertragung.

Wie in Kapitel 2.5.1 beschrieben entspricht die Rx-Leitung des CAN-Transceivers dem aktuellen Pegel des Busses. Diese Leitung kann nun, zusätzlich zum Anschluss an den integrierten CAN-Controller, ohne weiteres mit einem zweiten Eingang des Mikrocontrollers verbunden werden. Nun können der Pegel und die Flanken des Busses vom Mikrocontroller registriert werden. Den Pegel kann die Software direkt auslesen, die Erkennung der Flanken kann, bei ausreichender Taktung, über Software oder über extern triggerbare Zähler erfolgen.

Falls für die zeitliche Verzögerung ein Timer-Interrupt benutzt werden soll, muss für den betreffenden Zeitraum ein Hardware-Timer zur Verfügung stehen.

[3] NOP steht für *no operation*, einen Assemblerbefehl ohne Auswirkungen, der die CPU einen Takt lang belegt

6 Realisierung

In diesem Kapitel werden die Strategien aus der Konzeption, die den Anforderungen genügen, umgesetzt.

6.1 ASR-Steuernachricht

Die Steuernachricht der Antriebsschlupfregelung sieht wie folgt aus: Die Nachrichten-ID setzt sich nach dem Protokoll von Haase (2007) aus der Prioritätsklasse der Dienstnachrichten und der ID des ASR-Moduls, die auf 0x10 festgesetzt wurde, zusammen. Es ergibt sich folgende ID:

$$4 \ll 8 \mid 0x10 = 0x410$$

Das 1. Byte der 8 Datenbytes der CAN-Nachricht wird vom Protokoll für die Version verwendet. Byte Nummer zwei enthält die ID des Busteilnehmers, für den diese Nachricht bestimmt ist. Da die Nachricht an mindestens zwei Empfänger gerichtet ist, das ECU-Gateway und die Lenkradanzeige, wird dort 0xFF für Broadcast-Nachrichten eingesetzt.

Die verbleibenden 6 Datenbytes der Nachricht sollen für Folgendes genutzt werden:

Regelungsstatus
: Dieser Wert entspricht einem der Zustände der Antriebsschlupfregelung, siehe 5.1.4 Zustände der ASR. Hierfür reicht ein Byte aus.

Zündwinkelwert
: Dieser Wert soll vom ECU-Gateway, wenn der Status auf aktiv gesetzt ist, an die Motorsteuerung gesendet werden, um das Drehmoment zu beeinflussen. Da die Motorsteuerung nur die Variation über 256 Möglichkeiten anbietet, ist hier ein Byte ebenfalls ausreichend.

Siehe dazu auch im Anhang A.3 Steuernachrichten auf Seite 72.

6.2 Registrieren der TimeMaster-Nachricht

Wie in Abschnitt 5.3.3 Konzeption der Strategie „frühzeitigeres Erkennen“ beschrieben, horcht jeder Reserve-TimeMaster in einem bestimmten Zeitbereich innerhalb des Referenznachrichten-Busslots auf Busaktivität. Die Umsetzung lässt sich in die folgenden Teile gliedern.

6.2.1 Abpassen des Prüfbereiches

Wie in Tabelle 5.1 Prüfintervalle der Reserve-TimeMaster auf Seite 57 ersichtlich ist, hat der Prüfbereich für jeden Reserve-TimeMaster einen unterschiedlichen Startzeitpunkt. Dieser lässt sich aus der Nummer des TimeMasters ableiten. Die Dauer des Prüfbereiches ist für alle TimeMaster gleich und beträgt 20 μs.

Um diese Zeiten zu messen, gibt es zwei Möglichkeiten:

1. Software Delay:
   Das Programm zählt in einer vorher berechneten Schleife, um Zeit verstreichen zu lassen.
2. Hardware Delay:
   Bei Beginn des Intervalls wird ein Hardware-Timer so eingestellt, dass er nach der gewünschten Zeit einen Interrupt erzeugt. Das Hauptprogramm läuft anschließend in eine Endlosschleife, in der es eine boolesche Variable überprüft.
   Der Interrupt des Timers bewirkt lediglich eine kurze Interrupt Service Routine, die den Wert dieser Variablen negiert. Das Hauptprogramm verlässt daraufhin die Schleife und setzt die Programmlogik fort.

In diesem Fall sind beide Varianten möglich, da noch mehrere Hardware-Timer unbenutzt sind. Um diese Timer evtl. für eine spätere Erweiterung nutzen zu können, wird die erste Variante gewählt. Die zweite müsste gewählt werden, wenn das Hauptprogramm in der Wartezeit andere Aufgaben abarbeiten sollte. Dies ist hier aber nicht der Fall.

6.2.1.1 Software-Delay

Zur Integrierung des Software-Delays wurde eine im avr-gcc-Paket[1] enthaltene Delay-Methode genutzt. Dieser wird die zu wartende Zeit in Millisekunden als Parameter übergeben. Die Methode basiert auf einer exakt 3 Takte dauernden Assembler-Schleife, die entsprechend oft durchlaufen wird.

[1] avr-gcc ist der C-Compiler für AVR-Mikroprozessoren

Da diese Methode ebenso wie die von ihr aufgerufene Methode als inline deklariert ist, entsteht keine Verzögerung durch den Methodenaufruf oder die Berechnung der Anzahl der Schleifendurchläufe. Der Compiler fügt den Code der Delay-Methode direkt in die aufrufende Stelle ein und berechnet die Anzahl der nötigen Durchläufe im Voraus. Hierbei muss beachtet werden, dass der übergebene Parameter konstant ist und nicht von einer Variablen abhängt. Außerdem muss die Optimierung des Compilers aktiviert sein.

6.2.2 Bestimmen der Busaktivität

Um die Busaktivität zu bestimmen wird ein Hardware-Timer als Zähler benutzt. Der Mikrocontroller AT90CAN128 (siehe 2.5) besitzt 4 Timer, jeweils 2 sind 8 Bit bzw. 16 Bit breit. 8 Bit Zählerbreite ist hierbei mehr als ausreichend.

6.2.2.1 Timer-Wahl

Der erste, Timer0 mit 8 Bit, wird vom Scheduler verwendet, die übrigen drei sind bislang ungenutzt. Timer2 mit 8 Bit wird mit einem externen Takt im asynchronen Modus betrieben, ist daher z. B. für Uhrenquarze geeignet, lässt sich aber nicht ohne 4 externe Takte zurücksetzen. Da dies aber erforderlich ist, muss auf einen der größeren Timer ausgewichen werden.

Praktischerweise werden bei dem Mikrocontroller der Timer1-Takteingang und der CAN-Rx-Eingang an denselben Anschluss-Pin geführt, sodass keine zusätzlichen Lötarbeiten erforderlich sind, um den Takteingang mit der CAN-RX-Leitung zu verbinden. Da Timer1 und Timer2 funktional identisch sind, fällt die Wahl auf Timer1.

6.2.2.2 Timer-Konfiguration

Der Timer wird so konfiguriert, dass er den externen Takteingang benutzt, also die CAN-Rx-Leitung als Takt fungiert. Zu Beginn des Prüfbereiches wird der Timer auf 0 gesetzt. Nach Ablauf der Prüfzeit wird der Zählerstand ausgelesen: Wenn dieser ungleich Null ist, war der Bus aktiv.

6.3 Erweiterung der TimeMaster um das TTC-System

Die Software des bislang einzigen TimeMasters beruhte auf dem Scheduler von Schuckert (2007) und der CAN-Bibliothek von Haase (2007). Die TTC-Erweiterung war nicht erforderlich, denn dieser TimeMaster war in jeder Situation dominant und musste keine äußeren Umstände beachten. Nach diesem TimeMaster haben sich alle anderen Module gerichtet. Wäre der TimeMaster ausgefallen und hätte wieder eingesetzt, hätte er mit dem ersten Basiszyklus begonnen.

Nun gibt es mehrere TimeMaster, von denen aber zu jedem Zeitpunkt nur einer dominant sein darf. Ein zum Leben erwachter TimeMaster prüft, ob bereits ein anderer TimeMaster aktiv ist. Ist dies der Fall, synchronisiert er sich an dem bestehenden Buszyklus. Hierfür wird die TTCAN-Funktionalität benötigt. Falls er eine niedrigere ID besitzt, ist er dominanter und übernimmt das Senden der TM-Nachricht.

Die Module synchronisieren sich an den Referenznachrichten. Da diese nun vom Busslot aus betrachtet zu unterschiedlichen Zeiten versendet werden (siehe Abschnitt 5.3.3 Konzeption der Strategie „frühzeitigeres Erkennen“), muss auch die Synchronisation geändert werden.

6.3.1 Bisherige Synchronisation

Bislang haben die Module sich nur relativ synchronisiert. Das Verfahren wurde in den Abschnitten 2.4.5.1 Synchronisation und 2.4.5.2 Die Funktion eines TimeMasters beschrieben und besteht aus zwei Teilen: Als erstes wird die Referenznachricht empfangen und der Zeitstempel gesichert. Der zweite Schritt vergleicht die Zeitstempel und synchronisiert den Scheduler. Dieser Schritt muss als eigener Task implementiert werden, um einen Tick kontrolliert verlängern oder verkürzen zu können (vgl. Schuckert, 2007, S. 37). Dieser Task muss nicht direkt auf den Empfang folgen, jedoch innerhalb des Basiszyklus ausgeführt werden.

Relativ bedeutet, dass die Zyklusdauer der Module immer der des TimeMasters angeglichen wird, aber es einen Versatz der Zyklen unterschiedlicher Module geben kann, wie die Abbildung 6.1 auf der nächsten Seite zeigt.

Hierbei entsteht der Versatz durch zwei Gegebenheiten:

1. Durch die Übertragungszeit der Nachricht, die erst über den Bus gesendet (siehe dazu 5.2.1 Nachrichtendauer) und anschließend vom CAN-Controller des Empfängers angenommen werden muss, bevor dessen Software die Nachricht erhält.
2. Der bisherige TimeMaster hat seine Referenznachricht nicht zu Beginn der Basiszyklen versendet, sondern 7 Schedulerticks später. Dies war kein Problem, da sich die übrigen Module nur an der Zeitdifferenz der Referenznachrichten orientiert haben, der

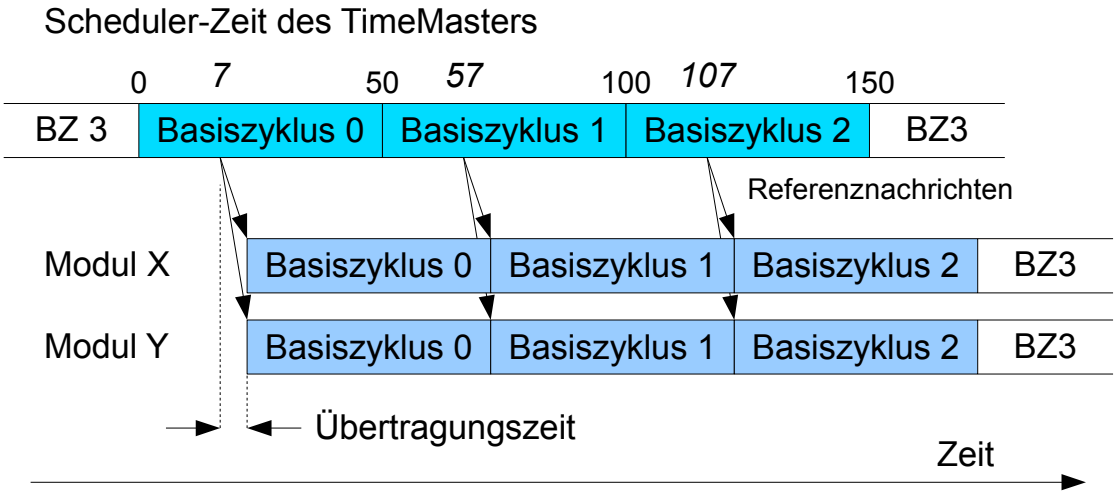

Abbildung 6.1: Frühere Synchronisation mit Versatz

TimeMaster außer der Referenz keine anderen Nachrichten versendet hat und somit nicht zeitgleich zum Buszyklus laufen musste.

6.3.2 Neue Synchronisation

Den Versatz durch den Scheduler könnte man durch Versetzen des Sende-Tasks einfach auf 0 schieben, jedoch bliebe der Versatz durch die Nachrichtenübertragung, deren Dauer mit 222 bis 260 μs zudem um 38 μs variiert. Bislang war dies kein Problem, da die Synchronisation entsprechend tolerant war. Nun muss jedoch eine Zeit von 20 μs eingehalten werden: die Prüfbereiche der TimeMaster, siehe 5.3.3 Konzeption der Strategie „frühzeitigeres Erkennen“. Es darf keinen außergewöhnlichen Versatz zwischen den Referenznachrichten verschiedener TimeMaster geben. Alle TimeMaster müssen stets exakt laufen, mit einem Versatz unter 10 μs.

Die Übertragungszeit der Nachricht im Voraus zu berechnen ist wegen der variierenden Nachrichtenlänge (siehe Abschnitt 5.2.1 Nachrichtendauer) nicht möglich. Stattdessen wird Folgendes umgesetzt:

Der Synchronisationszeitpunkt wird auf das Ende der Nachrichtenübertragung gesetzt. Gleichzeitig wird damit eine absolute Synchronisation bewirkt. Mit absolut ist gemeint, dass der Versatz der Zyklen unterschiedlicher Module kontrolliert und auf ein Minimum gebracht wird.

Hierbei wird ein Wartefenster eingeführt, dessen Länge nicht konstant ist, sondern mit der Übertragungsdauer der Referenznachricht variiert. Für die passiven TimeMaster und übrige Busteilnehmer ist das Wartefenster beendet, wenn die Referenznachricht empfangen wurde. Der aktive TimeMaster empfängt seine Nachricht nicht selbst, kann aber den erfolgreichen Abschluss der Übertragung als Anhaltspunkt nehmen. Diesen erfährt er über die Statusregister seines CAN-Controllers.

Die folgende Abbildung 6.2 veranschaulicht dieses Zusammenspiel. Mit TXOK-Flag ist das Registerbit des CAN-Controllers gemeint, welches beim Sender nach einer erfolgreichen Übertragung gesetzt wird.

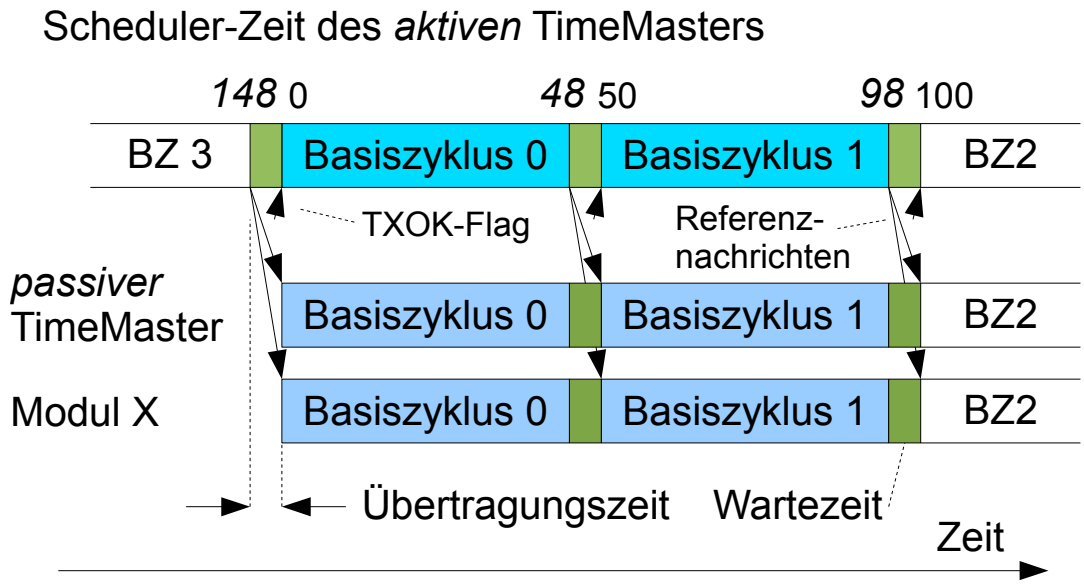

Abbildung 6.2: Neue Synchronisation nach Übertragungsende

6.3.3 Startsynchronisation

Die TimeMaster verhalten sich zu Beginn ihrer Lebenszeit wie in folgendem Zustandsdiagramm aus Abbildung 6.3 auf der nächsten Seite dargestellt.

Jeder TimeMaster startet im Zustand *passiv*. In diesem Zustand wartet er ab, ob ein anderer TimeMaster bereits aktiv ist. Falls er eine Referenznachricht von einem anderen TimeMaster empfängt, synchronisiert er sich an dieser Nachricht. Dies ist erforderlich für die Anforderung 3.2.3 Zykluskonsistenz. Anschließend geht er in den Zustand *aktiv* über. Empfängt er eine gewisse Zeit lang, *timeout* genannt, keine Referenznachricht, wechselt er von selbst in den

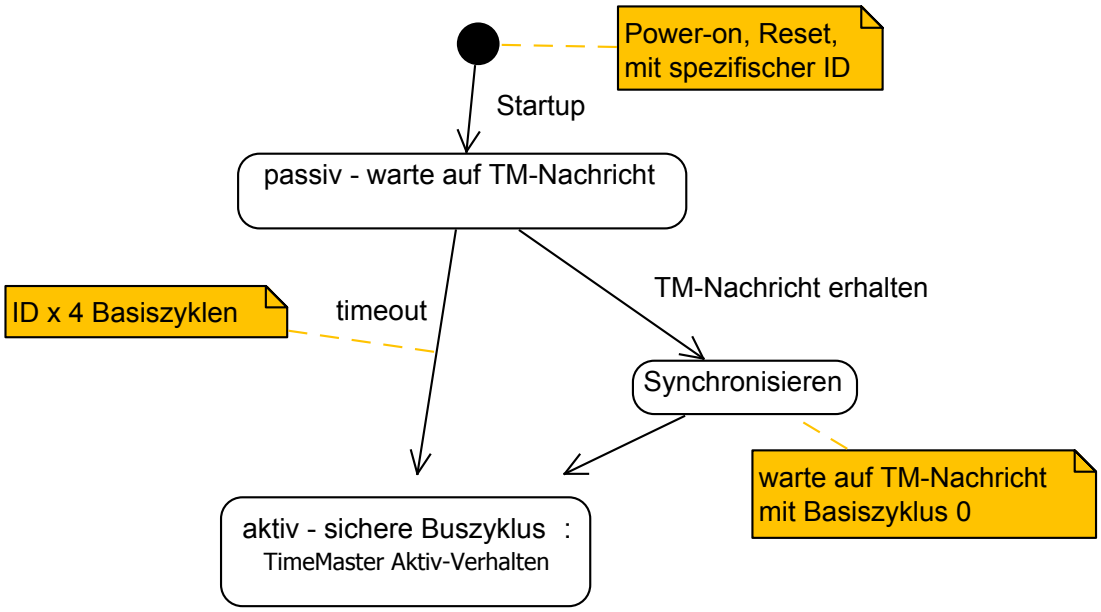

Abbildung 6.3: Startverhalten der redundanten TimeMaster

Zustand *aktiv*, und startet mit dem Senden seiner Referenznachricht, beginnend mit dem 0. Basiszyklus.

Dieser *timeout* hängt von der ID des TimeMasters ab und ist so für alle TimeMaster unterschiedlich. Dies ist notwendig, damit bei einem simultanen Start mehrerer TimeMaster, wie er ständig beim Einschalten der Fahrzeugelektrik vor kommt, nicht jeder dieser TimeMaster für sich entscheidet, der erste TimeMaster zu sein. Dies würde beieinanderliegende Referenznachrichten mit derselben Zyklusnummer bewirken.

6.3.4 Fortführende Synchronisation

Der *aktiv*-Zustand, in dem sich die TimeMaster fortan endlos befinden, ist für das Aufrechterhalten des Buszyklus zuständig. In dem Zeitfenster der Referenznachricht handelt er wie der nachstehende Zustandsautomat aus Abbildung 6.4 auf der nächsten Seite.

Mit *RX OK* ist das Status-Flag der Hardware gemeint, welches den erfolgreichen *Empfang* einer Nachricht anzeigt, mit *TX OK* ist das Status-Flag der Hardware gemeint, welches den erfolgreichen *Versand* einer Nachricht anzeigt.

Es wird wie im Konzept gefordert der Bus zur jeweiligen Prüfzeit auf Aktivität getestet. Wird Aktivität registriert, wird der Empfang der Nachricht als Synchronisationszeitpunkt gewählt.

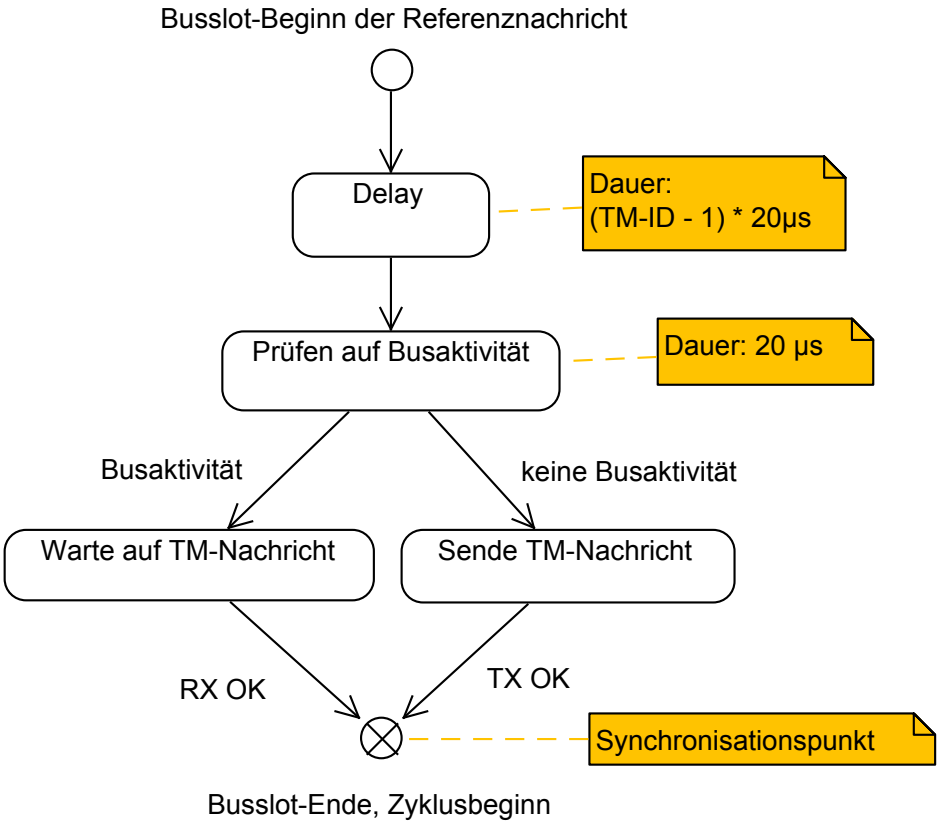

Abbildung 6.4: Aktiv-Verhalten der redundanten TimeMaster
im Busslot der Referenznachricht

Gibt es keine Aktivität, ist kein priorisierterer TimeMaster aktiv, wird die eigene Referenznachricht versendet und das Ende des Versands als Synchronisationszeitpunkt gewählt. Mit dem Synchronisationszeitpunkt endet das Zeitfenster und der Scheduler führ den Task des nächsten Zeitfensters aus.

7 Fazit

In diesem Kapitel wird das Ergebnis dieser Arbeit geschildert, besprochen und bewertet. Außerdem werden Vorschläge zur weiteren Verbesserung gemacht.

7.1 Bewertung der Einsatzmöglichkeiten

Die Entwicklungen konnten wie folgt im Rennwagen eingesetzt werden:

7.1.1 TimeMaster-Redundanz

Die TimeMaster-Redundanz wurde erfolgreich entwickelt. Im Test konnte das geforderte Verhalten kontrolliert werden. Auch im Fahrzeug wurden zwei TimeMaster eingesetzt. Seit der Integration des zweiten TimeMasters war das Fahrzeug etwa zwei Wochen auf dem Testgelände in Papenburg und eine Woche auf dem Formula Student Germany Event in Hockenheim gefordert. In dieser Zeit gab es kein Problem, welches auf die TimeMaster bzw. die Synchronisation zurückzuführen wäre.

Es wurden nicht mehr als zwei TimeMaster integriert, da der in den bereits gefertigten Elektronik-Gehäusen verfügbare Platz ausgereizt war. Zudem werden von dem Bussystem keine unbedingt notwendigen Steuerungsaufgaben übernommen, sodass eine Redundanz, die über zwei Module hinausgeht, noch nicht erforderlich ist.

7.1.2 Antriebsschlupfregelung

Die Antriebsschlupfregelung konnte nicht vollständig entwickelt werden. Im Labor wurde die Manipulation der Motorsteuerung erfolgreich getestet. Da der Rennwagen jedoch erst sehr spät, kurz vor dem Event in Silverstone, fertiggestellt wurde, ergab sich keine Möglichkeit die Antriebsschlupfregelung zu integrieren. Da für die Abstimmung der Antriebsschlupfregelung viel Zeit zum Testen nötig ist, musste auf eine Integration verzichtet werden.

7.2 Ausblick

Das gesamte Bussystem hat wie im Vorjahr ausnahmslos seinen Dienst verichtet. Die TimeMaster-Redundanz hat die Sicherheit des Systems erweitert und die Basis für Fahrassistenzsysteme gelegt. Das Telemetriesystem könnte gut in die zukünftigen Rennwagen des Teams integriert werden.

7.2.1 Verbesserungen

Eine mögliche Verbesserung sehe ich in der Überwachung der TimeMaster. Durch die Referenznachricht ist die Präsenz des aktiven TimeMasters allen Modulen ersichtlich. Auch im Live Monitoring wird die ID des momentan aktiven TimeMasters angezeigt, wie die Abbildung 7.1 zeigt:

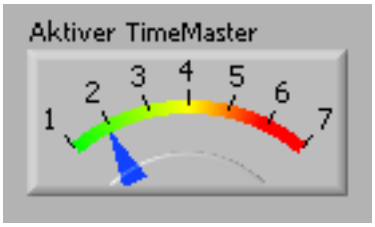

Abbildung 7.1: Anzeige der ID des aktiven TimeMasters im Live Monitoring

Es ist jedoch nicht bekannt, wieviele TimeMaster in das System integriert sind und welche von diesen noch einsatzbereit sind, da sich im Normalfall alle niedriger priorisierten TimeMaster stumm verhalten. Für eine genauere Statusübersicht sollten die bereits im Protokoll (siehe Anhang A) definierten Statusnachrichten genutzt werden. Diese werden periodisch von den Modulen verschickt und enthalten den Status des Schedulers sowie weitere komponentenspezifische Statuswerte.

Damit könnte der Zustand der TimeMaster überwacht werden und z. B. im Live Monitoring oder auf einer Diagnose-Seite des Lenkraddisplays angezeigt werden.

Literaturverzeichnis

[AT90 Datenblatt 2008] ATMEL CORPORATION (Hrsg.): *AT90CAN32/64/128 Datasheet*. 2008. – URL `http://www.atmel.com/dyn/resources/prod_documents/doc7679.pdf`. – Abruf: 2008-08-22

[Bagschik 1999] BAGSCHIK, Peter: *CAN-Hardware/Software-Grundlagen : Ausnahmebehandlung*. Kap. 4.1.4. In: LAWRENZ, Wolfhard (Hrsg.): *CAN Controller Area Network*. Heidelberg : Hüthig, 1999. – ISBN 3-7785-2734-7

[Bosch Fahrsicherheitssysteme 1998] BECKER, R. ; PFÄFFLE, J. ; SOWA, P. ; CZINCZEL, A. ; SCHMIDT, G. ; GERSTENMAIER, J. ; KNUST, A. ; KÜHNER, K. ; REINKE, K.-H. ; STEGMAIER, A. ; MEISSNER, M. ; SIGL, A.: Bremssysteme für Personenkraftwagen : Antriebsschlupfregelung. In: ROBERT BOSCH GMBH (Hrsg.): *Fahrsicherheitssysteme*. Braunschweig/Wiesbaden : Vieweg, 1998, S. 32–81. – ISBN 3-528-03875-6

[DIN 40041 1967] DEUTSCHES INSTITUT FÜR NORMUNG (Hrsg.): *DIN 40041: Zuverlässigkeit elektrischer Bauelemente : Begriffe*. Bd. 40041/Vornorm. Berlin/Köln : Beuth, 1967

[Etschberger 2000] ETSCHBERGER, Konrad: *CAN-Hardware/Software-Grundlagen : Ausnahmebehandlung*. Kap. 2. In: ETSCHBERGER, Konrad (Hrsg.): *Controller-Area-Network : Grundlagen, Protokolle, Bausteine, Anwendungen*. München, Wien : Carl Hanser, 2000. – ISBN 3-446-19431-2

[FSG Rules 2008 2007] FORMULA STUDENT GERMANY (Hrsg.): *Formula Student Germany Rules 2008*. 2007. – URL `http://www.formulastudent.de/uploads/media/Formula_Student_Germany_2008_Rules_01.pdf`. – Abruf: 2008-08-22

[Haase 2007] HAASE, Sebastian: *Telemetrie im Formula Student Rennwagen auf Basis von CAN Bus, Datenspeicherung und Wireless LAN Technologien*, Hamburg: Hochschule für Angewandte Wissenschaften, Dep. Informatik, Bachelorarbeit, 2007

[Lawrenz 1999] LAWRENZ, Wolfhard: *Eigenschaften und Anforderungen von Feldbusprotokollen*. Kap. 2.1.1. In: LAWRENZ, Wolfhard (Hrsg.): *CAN Controller Area Network*. Heidelberg : Hüthig, 1999. – ISBN 3-7785-2734-7

[Müller 2001] MÜLLER, Bernd: Die Zeit ist reif! : Zeitgesteuerte Kommunikation mit CAN. In: *Auto & Elektronik* 3 (2001), Nr. 2, S. 44–47

[Pont 2001] PONT, Michael J.: *Pattern for time-triggered embedded systems : Building reliable applications with the 8051 family of microcontrollers*. ACM Press, 2001. – ISBN 0-201-33138-1

[SAE Rules 2008 2007] SAE INTERNATIONAL (Hrsg.): *2008 Formula SAE® Rules*. 2007. – URL `http://students.sae.org/competitions/formulaseries/rules/rules.pdf`. – Abruf: 2008-08-22

[Schuckert 2007] SCHUCKERT, Simon: *Mikrocontrollerbasierte Telemetrie und Echtzeitauswertung von Sensordaten im Formula Student Rennwagen*, Hamburg: Hochschule für Angewandte Wissenschaften, Dep. Informatik, Bachelorarbeit, 2007

[Schulze 1996] SCHULZE, W.: Can für Echtzeitanwendungen in der Verpackungsindustrie. In: RZEHAK, Helmut (Hrsg.): *Echtzeitsysteme und objektorientierter Entwurf*. Braunschweig/Wiesbaden : Vieweg, 1996, S. 167–178

[TTCAN 2000] FÜHRER, Thomas ; MÜLLER, Bernd ; DIETERLE, Werner ; HARTWICH, Florian ; HUGEL, Robert ; WALTHER, Michael ; ROBERT BOSCH GMBH (Hrsg.): *Time Triggered Communication on CAN (Time Triggered CAN - TTCAN)*. 2000. – URL `http://www.semiconductors.bosch.de/pdf/CiA2000Paper_1.pdf`. – Abruf: 2008-08-22

[Zimmermann und Schmidgall 2006] ZIMMERMANN, Werner ; SCHMIDGALL, Ralf: *Bussysteme in der Fahrzeugtechnik : Protokolle und Standards*. Wiesbaden : Vieweg, 2006. – ISBN 3-8348-0166-6

A Bus-Protokoll

A.1 Matrixzyklus

Die Tabelle A.1 auf der nächsten Seite zeigt die Anordnung der Nachrichten im Matrixzyklus.

A.2 Datennachrichten

Die Tabelle A.2 auf Seite 74 zeigt die Datennachrichten und die Bedeutung ihrer Datenbytes.

A.3 Steuernachrichten

Die Tabelle A.3 auf Seite 74 zeigt die Steuernachrichten und die Bedeutung ihrer Datenbytes.

Abbildung A.1: Protokoll: Matrixzyklus

Basiszyklus 0 (10ms) Steuerung

Scheduler- Zeit	198	0	2	4	6	8	10	12	14	16	18
Zeitfenster	400µs	2,4ms						400µs	400µs	400µs	400µs
Identifizierung	0x01 – 0x07							0x09	0x09	0x41	
										T C	

Basiszyklus 1 (10ms) Daten 1

Scheduler- Zeit	48	50	52	54	56	58	60	62	64	66	68
Zeitfenster	400µs	2,4ms						2ms			
Identifizierung	0x01 – 0x07							0x35 – 0x3A (6 Nachrichte			

Basiszyklus 2 (10ms) Daten 2

Scheduler- Zeit	98	100	102	104	106	108	110	112	114	116	118
Zeitfenster	400µs	2,4ms						400µs	400µs	400µs	400µs
Identifizierung	0x01 – 0x07										

Basiszyklus 3 (10ms) Fehlerkontrolle

Scheduler- Zeit	148	150	152	154	156	158	160	162	164	166	168
Zeitfenster	400µs	2,4ms						400µs	400µs	400µs	400µs
Identifizierung	0x01 – 0x07								0x09		

TM-Referenz	Reserviert für 50Hz und 100Hz Daten

20	22	24	26	28	30	32	34	36	38	40	42	44	46
400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs

70	72	74	76	78	80	82	84	86	88	90	92	94	96
	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs
n)	0x3B	0x33	0x34	0x3C	0x3D	0x3E	0x3F	0x40					

120	122	124	126	128	130	132	134	136	138	140	142	144	146
400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs

170	172	174	176	178	180	182	184	186	188	190	192	195
400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	400µs	600µs	600µs
			0x0D	0x0E						0x0C	0x0A	

Abbildung A.2: Protokoll: Datennachrichten

Nachrichten vom Typ Data Single (Byte-Reihenfolge: Big-Endian)												
Komponenten / Werte	**Quell-Adr.**	**B 0**	**B 1**	**B 2**	**B 3**	**B 4**	**B 5**	**B 6**	**B 7**	**Einheit**	**Bereich**	
Time Master Nr. 1 bis 7	0x01 - 0x07	Wird für Protokolldaten verwendet	u char u char u char u char	u char u char u char u char	u char u char u char u char			x x x x	0 1 2 3	Datum: dd.mm.yy Uhrzeit: hh.mm.ss Datum: dd.mm.yy Uhrzeit: hh.mm.ss	x = 0 -> BusMode: off x = 1 -> BusMode: record	
Raddrehzahl vorne links Raddrehzahl vorne rechts Geschw. vorne gemittelt	0x33		u short		u short		u short			RPM km/h	0 bis ... 0 bis 400	
Raddrehzahl hinten links Raddrehzahl hinten rechts Geschw. hinten gemittelt	0x34		u short		u short		u short			RPM km/h	0 bis ... 0 bis 400	
EngineRoundCounter MAP Phase1 TAir	0x35		u short		u short		u short		u char	1/bit 1mBar/bit 0.1°/bit 0.6°/bit	0 bis 65535 0 bis 65535 0 bis 3553,5 -30 bis 129,375	siehe K-Line Protokoll
TPS RPM Klambda1 Inj1	0x36		u char	u short	u short		u short			0,5%bit 1rpm/bit 1e-4/bit 1µS/bit	0 bis 127,5 0 bis 65535 0 bis 99.9009 0 bis 65535	
Inj2 Inj3 Inj4 ValvPosition	0x37		u short		u short		u short		u char	1µS/bit 1µS/bit 1µS/bit 0,4%bit	0 bis 65535 0 bis 65535 0 bis 65535 0 bis 100	
Spark3 Spark4 Spark2 Sidestand	0x38		s short		s short		s short		u char	0,1°/bit 0,1°/bit 0,1°/bit 1/bit	-3276,8 bis 3276,7 -3276,8 bis 3276,7 -3276,8 bis 3276,7 0 bis 255	
Spark1 Lambda VBatt DFarf TEngine LambdaTarget	0x39		s short	u char	u char	s char	u char	u char	u char	0,1°/bit 0,01/bit 0,07V/bit 1/bit 0.6°/bit 0,01/bit	-3276,8 bis 3276,7 0 bis 2,55 0 bis 18,05 -127 bis 128 -30 bis 129,375 0 bis 2,55	
Speed Tipover Gear Active Block BAP	0x3A		u short	u char	u char	u char	u short			km/h 1/bit 1/bit 1/bit	0 bis 255 0 bis 255 0 bis 255 0 bis 255	
Lenkwinkel	0x3B		s short							Grad	-180 bis +180	
Beschleunigung x (hinten) Beschleunigung y (rechts) Beschleunigung z (oben)	0x3C		s short		s short		s short			mg	-6g bis 6g	
Drehrate Drehratensensortemperatur Drehratensensortemperatur Drehratensensortemperatur	0x3D		s short		u short		s short		s char	°/s, positiv = UZS bit K °C	12 bit 12 bit, 1 K / 6.88 b 0 bis 595 K geschnitten auf ±127°C	
Öldruck Öltemperatur Ölschnorchelposition	0x3E		u char	u char	s short					dezibar °C ° im UZS	0 bis 10 bar 0 bis ... 0° bis 359°	
Bremsdruck	0x3F		u char							bar	0 bis 200 bar	
Gang	0x40		s char							0 neutr, -1 unbek.	0 bis 5	

Abbildung A.3: Protokoll: Steuernachrichten

Nachrichten vom Typ Control (Byte-Reihenfolge: Big-Endian)								
Steuernachricht	**B 0**	**B 1**	**B 2**	**B 3**	**B 4**	**B 5**	**B 6**	**B 7**
Generel Command	Protokolldaten	Zieladresse	0x01 = Reset 0x05 = BusModeOff 0x06 = BusModeOn					
Datalogger Read File			0x11 = Read File		FileID (uint32)			
Datalogger Command			0x12 = Read File Folder Structure 0x13 = Record Data 0x15 = Cancel					
TimeMaster Time				0x22 = Set Time		hour (uint8)	minute (uint8)	second (uint8)
TimeMaster Date				0x33 = Set Date	dayOfTheWeek (uint8)	day (uint8)	month (uint8)	year (uint8)
Whellspeed Tire Size				0x55 = Tire Size		1 = km/h 2 = rpm	tire size in mm (uint16)	
Antriebsschlupfregelung			0x01 = inaktiv 0x02 = passiv 0x03 = aktiv 0x04 = regelbegrenzt 0x05 = fehlerhaft	Zündwinkel-Sollwert				

Tabellenverzeichnis

Abbildungsverzeichnis

Glossar

Bezeichnung	Beschreibung	Seiten
Arbitrierung	Die kontrollierte Zuweisung von Zugriffsrechten für Komponenten, etwa Bussystemen, an mehrere interessierte Module, siehe 2.5.1.2.	14, 26, 55, 56
Busauslastung	Das Verhältnis zwischen der von Nachrichten belegten und der ungenutzen Buszeit.	9, 46
Delay	Dt. Verzögerung, zur geplanten zeitlichen Kontrolle des Programmablaufes.	61
Gateway	An mehrere Netzwerke angeschlossene Komponente, die Nachrichten zwischen diesen Netzen vermittelt, wenn diese nicht direkt verbunden werden können.	50, 52, 53, 60
Live Monitoring	Live Monitoring ist die Echtzeitüberwachung des Fahrzeugzustandes im Kontrollstand.	52, 69
Scheduler	Dt. Planer, Verwaltungsarchitektur zur periodischen Durchführung mehrerer Aufgaben, siehe 2.6.	27, 47, 69
Software Engineering	Software Engineering beschäftigt sich mit der effizienten Softwareentwicklung.	46
Task	Ein Programm wird in logisch unabhängige Tasks (dt. Aufgaben) aufgeteilt.	27
Telemetrie	Die Übertragung von Sensorwerten vom Messort zu einem entfernt gelegenen Beobachtungsort.	11, 13, 34, 47, 53
Tick	Die kleinste zählbare Zeiteinheit eines getakteten Systems oder eine Schedulers.	27, 59

Bezeichnung	**Beschreibung**	**Seiten**
Totpunkt	Der Punkt in einem Hebelmechanismus, bei dem die verbindenden Gelenke und die einwirkenden Kraftvektoren auf einer Linie liegen.	43, 44
Transceiver	Eine kombinierte Sende- und Empfangseinheit, siehe 2.5.1.	22, 24, 25, 59

Abkürzungsverzeichnis

Bezeichnung	Beschreibung	Seiten
ABS	Antiblockiersystem	8, 15
ASR	Antriebsschlupfregelung	8, 9, 14, 15, 53, 68
CAN	Controller Area Network, siehe 2.4.6	11, 45, 53, 59
ECU	Engine Control Unit, dt. Motorsteuerung, siehe 4.2.3	50, 52, 53, 60
ESP	Elektronisches Stabilitätsprogramm	8, 15
FAS	Fahrassistenzsystem	8, 9
TM	TimeMaster	9, 14
TTCAN	Time Triggered Controller Area Network, siehe 2.4.7	9, 22

Index

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 22. August 2008	
Ort, Datum	Unterschrift