Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung
Seminar

Jörn Sellentin

Zeitgesteuerte Kommunikationssysteme
für verteilte
Hard-Real-Time Anwendungen

Betreuender Prüfer: Prof. Dr.-Ing. Bernd Schwarz

Fakultät Technik und Informatik
Studiendepartment Informatik

Jörn Sellentin

Thema der Seminararbeit

Zeitgesteuerte Kommunikationssysteme für verteilte Hard-Real-Time Anwendungen

Stichworte

Verteilte Systeme, Echtzeit, Verlässlichkeit, Fehlertoleranz, Fail-Silence, Fault-Tolerant-Unit Kraftfahrzeug, FlexRay, Kommunikationszyklus, TDMA, Time-Triggered, TTCAN, TTP/C, CAN, Softwareentwurf

Kurzzusammenfassung

In dieser Seminararbeit werden die Erfordernisse für ein zeitgesteuertes Kommunikationsmedium für Hard-Real-Time Anwendungen beschrieben, wie es in den Entwicklungen in der Automobilindustrie verwendet wird.
Die Thematik der verlässlichen und fehlertoleranten Kommunikation wird erläutert und der Ansatz der Fault-Tolerant-Unit und der Fail-Silence Modus für fehlertolerante Systeme dargestellt.
Des Weiteren wird die zeitgesteuerte Kommunikationstechnologie FlexRay vorgestellt und deren Konzepte zur Verbesserung der Fehlertoleranz wie zum Beispiel der redundanten Datenübertragung dargestellt. Der Focus liegt hierbei auf der Sicherungs- und der Vermittlungsschicht des OSI Schichtenmodells und der zusätzlichen Integration einer ereignisgesteuerten Kommunikation in das zeitgesteuerte FlexRay Protokoll. In diesem Zusammenhang werden die einzelnen Komponenten eines FlexRay Knotens vorgestellt, das zeitgesteuerte Busprotokoll beschrieben und der dafür benötigte Uhrensynchronisationsalgorithmus erläutert. Zudem beinhaltet die Seminararbeit einen Vergleich zwischen FlexRay und anderen Kommunikationssystemen der Automobilindustrie.
Anschließend wird der Softwareentwurf eines auf FlexRay basierenden zeitgesteuerten Kommunikationssystems dargestellt. Der wichtigste Aspekt dieses Abschnitts ist der gemeinsame Zeitplan aller beteiligten Komponenten und seine Rolle bei der Systemerstellung.
Zum Abschluss wird ein Ausblick auf eine Masterarbeit in dem Bereich der zeitgesteuerten Kommunikationssysteme gegeben. Diese umfasst den Umbau eines fahrerlosen Transportsystems, auf dem Mini-PC's eingesetzt wurden, auf FlexRay Hardware. Des Weiteren behandelt die Masterarbeit den Softwareentwurf für ein zeitgesteuertes Softwarekonzept in den Komponenten, sowie den Entwurf eines zeitgesteuertes Kommunikatssystems zwischen den Komponenten. Zusätzlich wird ein Masterarbeitsthema der Firma Decomsys erarbeitet, welches einen tieferen Einblick in die FlexRay Technologie ermöglicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Diese Ausarbeitung beschreibt die Erfordernisse für ein zeitgesteuertes Kommunikationssystem in verteilten Hard-Real-Time Anwendungen. Es stellt als verfügbare Technologie FlexRay vor und vergleicht diese mit anderen am Markt erhältlichen Produkten.

Verteilte Systeme werden aus den unterschiedlichsten Gründen immer häufiger eingesetzt.

- Sicherheit
  Bei dem Ausfall einer Komponente kann das verbleibende System fehlerfrei weiterarbeiten.
  Der Ausfall von einer Komponente kann durch die Übernahme der Funktion durch eine andere funktionierende Komponente kompensiert werden.
  Mehrere Komponenten führen die gleichen Berechnungen durch, um zufällige Fehler zu entdecken.
- Zusammensetzbarkeit
  Die Steuerungshardware und Softwareentwicklung kann an Fremdfirmen mit dem Know-how eines bestimmten Anwendungszwecks ausgelagert werden.
  Der Testaufwand kann gesenkt werden, da nur die neuen Komponenten getestet werden müssen, sofern die Schnittstellen erhalten bleiben.
- Skalierbarkeit
  Reicht die Verarbeitungskapazität nicht aus, können einem verteilten System neue Komponenten hinzugefügt werden.
- Kosten
  Mit mehreren günstigen Rechnern die gleiche oder mehr Rechenleistung erreichen als mit einem einzelnen Rechner erzielt werden kann.
  Bindung der Komponente an ihren Einsatzort und Verbindung der Komponenten untereinander über einen Bus, um Verdrahtungsaufwand einzusparen.

Ein höherer Grad an Verteilung in einem System steigert den Kommunikationsaufwand zwischen den einzelnen Komponenten. Dies stellt strenge Anforderungen an das Kommunikationsmedium. In heutigen Kraftfahrzeugen werden zum Beispiel mehr als 50 Steuerungen eingesetzt. Durch die Bandbreiteneinschränkungen des heute weit verbreiteten CAN-Busses müssen in Oberklassewagen bis zu 4 unabhängige Busse eingesetzt werden, um dem Kommunikationsaufwand gerecht zu werden. Die Entwickler von Fahrzeugen stehen an den Grenzen der heute üblichen Kommunikationstechnologien, dadurch ist es notwendig, dass neue Technologien entwickelt werden, mit den man auch in der nahen Zukunft die gewünschten Ergebnisse erzielen kann. Eine neue Technologie ist FlexRay, welche in dieser Ausarbeitung vorgestellt wird.

Anforderungen an das Kommunikationsmedium in Fahrzeugen

Durch Technologien wie zum Beispiel ESP, ABS und ASR wird in das Fahrverhalten eingegriffen, um gefährliche Fahrsituationen zu verhindern [5]. Um diese Funktionen zu realisieren, werden sicherheitsrelevante Daten zwischen den beteiligten Komponenten übertragen. Gehen diese Daten verloren oder kommen sie nicht rechtzeitig an kann es zu unerwünschten möglicherweise sogar gefährlichen Fahrsituationen kommen. Diese Funktionen erfordern ein immer höheres Maß an Sicherheit in der Kommunikationsschicht. Zum Beispiel sollen Ausfallwahrscheinlichkeiten so gering wie möglich gehalten werden und der Ausfall einer Komponente darf die Buskommunikation der anderen Komponenten nicht beeinträchtigen. Zu diesem Zweck muss das gesamte Hard- und Softwaresystem deterministisches Verhalten aufwei-

sen, das bedeutet, dass das Verhalten, zu jedem Zeitpunkt der Laufzeit des Systems, bestimmt ist. In Zukunft sollen Technologien wie X-by-wire, welche die elektrische Lenkung und die elektrische Bremse ohne mechanische Verbindungen beinhaltet, vom TÜV für den Straßenverkehr zugelassen. Dies ist aber nur möglich, wenn verlässliche Komponente, sowie eine verlässliche Kommunikationsschicht verwendet werden.
Die Übertragung von regelungstechnischen Daten erfordert eine garantierte Kommunikationslatenz mit minimalem Jitter, um die Ist- und Sollwerte zwischen den an der Regelung beteiligten Komponenten auszutauschen. Dies setzt ein zeitgesteuertes Bussystem voraus. Zum Beispiel die Hybridtechnologie in modernen Kraftfahrzeugen erfordert einen hohen Abstimmungsaufwand zwischen den Geschwindigkeitsreglern der einzelnen Motoren. Das Schließen von Regelkreisen über den Bus fordert zudem aufgrund der periodischen Übertragungszyklen eine sehr hohe Bandbreite.
Zusätzlich sollen auch eventgesteuerte Nachrichten über das Bussystem gesendet werden, um nichtzyklische Diagnosefunktionen, wie zum Beispiel das Auslesen des Fehlerspeichers oder das Beeinflussen von Systemparametern, zu gewährleisten. Ebenso ist es auch erforderlich, die alten Kommunikationsstrukturen zu unterstützen, damit nicht alle Steuergeräte in den Fahrzeugen komplett ersetzt werden müssen, sondern auch eine Anpassung oder ein Austausch der Kommunikationsebene ausreicht.
Die Zusammensetzbarkeit des Busses muss gewährleistet sein. Dadurch können die Komponenten unabhängig voneinander entwickelt und getestet werden und das Zusammenfügen zum Gesamtsystem hat keinen Einfluss auf das Verhalten der einzelnen Komponenten. Dies spart Entwicklungs- und Testaufwand, da die Komponenten nicht wie sonst üblich im Verbund getestet werden müssen, um die durchschnittliche und die sehr schwierig zu ermittelnde maximale Buslast zu analysieren. In heutigen Kraftfahrzeugen werden ca. 50 und mehr Steuerungen eingesetzt, dies führt bei einer geringen Zusammensetzbarkeit zu einem nicht mehr kostengünstig zu bewerkstelligenden Testaufwand beim Integrieren des Gesamtsystems oder bei der Änderung an einer Komponente. In dem hier vorgestellten FlexRay Protokoll ist die Zusammensetzbarkeit bei gleich bleibenden Schnittstellen gegeben. Bei Änderungen an der Schnittstelle einer Komponente muss der statische Zeitplan neu berechnet und in alle Komponenten eingebracht werden (siehe Medienzugriff).
Die Skalierbarkeit von Kommunikationssystemen ist in die Konzeption der KFZ-Elektronik einzuplanen, da neue Komponenten durch Extrafunktionen oder Neuentwicklungen hinzukommen können. Zum Beispiel werden vom Gesetzgeber immer neue Assistenzsysteme wie das automatische Notrufsystem ‚eCall' gefordert, womit voraussichtlich ab 2009 alle in der EU produzierten Neufahrzeuge ausgestattet sein müssen. Dieses muss dann nachträglich in bestehende Fahrzeugstrukturen und somit in die KFZ-Kommunikationssysteme integriert werden [8].

Verlässliche Kommunikation

Dieses Kapitel gibt einen Überblick über die Terminologie zu verlässlichen Computersystemen und speziell über verlässliche Kommunikation [2]. Zusätzlich werden die Definitionen von Verlässlichkeit, Faults, Errors und Failures genannt. Das für die verlässliche Kommunikation sehr entscheidende Thema Fehlertoleranz wird in einem eigenen Abschnitt behandelt.

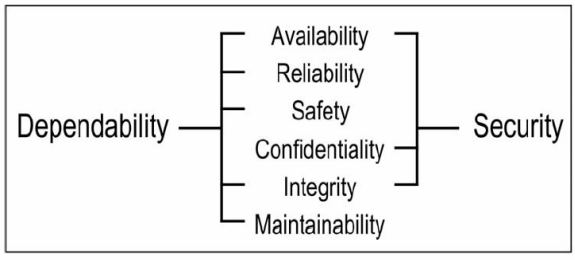

Abbildung 1 Attribute der Verlässlichkeit (Dependability) und Sicherheit (Security) [2]

Die Definition von Verlässlichkeit beschreibt die Fähigkeit einen Dienst anzubieten, dem vertraut werden kann. Verlässlichkeit beruht auf die Verfügbarkeit, Vertrauenswürdigkeit,

Sicherheit, Integrität und Wartbarkeit einer Komponente (siehe Abbildung 1).

Ein Failure, im folgenden Fehlverhalten genannt, ist ein Ereignis, durch welches die Abweichung eines Dienstes von seiner Spezifikation sichtbar wird. Ein Fehlverhalten zeigt also einen Verlust der Verlässlichkeit eines Systems an. Fehlverhalten lassen sich kategorisieren anhand ihrer Domäne (inhaltliche-, zeitliche-, Stopp-Ausfälle), ihrer Sichtbarkeit, ihrer Konsistenz und ihrer Konsequenzen (siehe Abbildung 2).

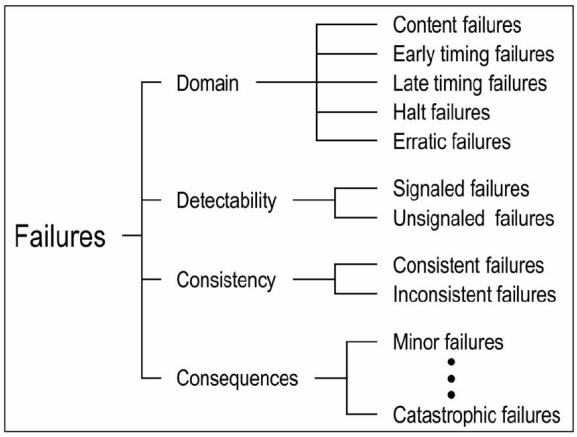

Abbildung 2 Kategorisierung von Fehlverhalten (Failures) [2]

Ein Fehlverhalten wird durch einen Error ausgelöst. Ein Error, im folgenden Fehler genannt, ist ein fehlerhafter interner Zustand eines Gerätes. Ein Fault, im folgenden Fehlerursache genannt, ist der Grund für einen Fehler, und damit der Grund für ein Fehlverhalten. Fehlerursachen können aus den verschiedensten Gründen entstehen, wodurch sie sich kategorisieren lassen. (siehe Abbildung 3).

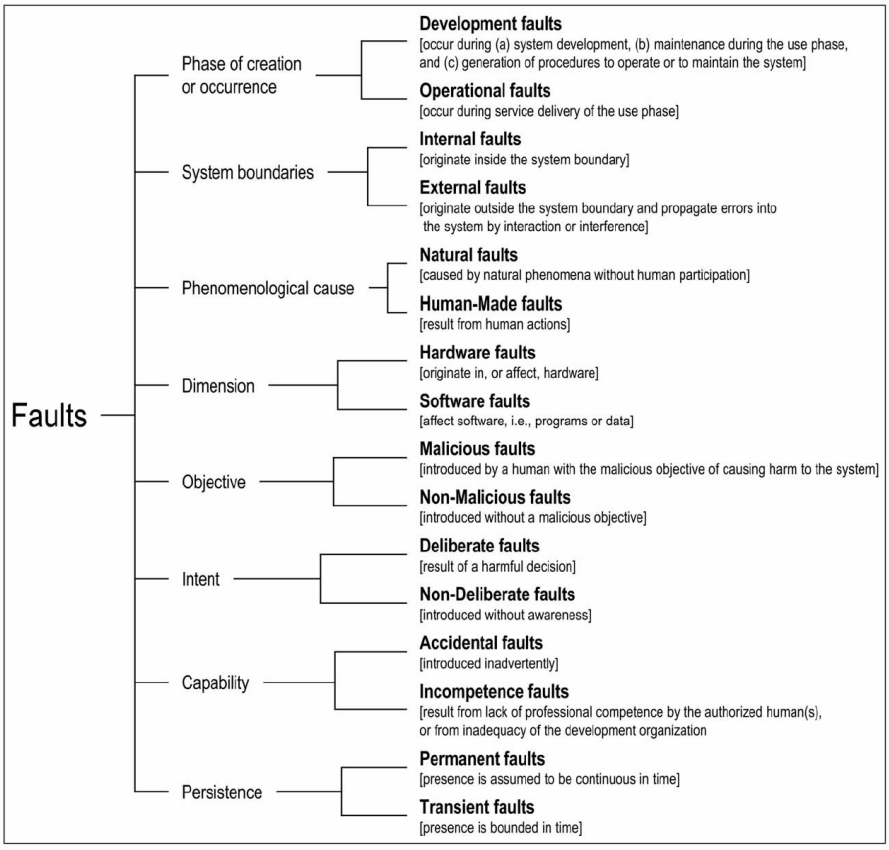

Abbildung 3 Übersicht verschiedener Fehlerursachen (Faults) [2]

Fehlertoleranz

Unter Fehlertoleranz versteht man die Vermeidung des Fehlverhaltens eines Dienstes aufgrund einer Fehlerursache. Fehlerursachen bewirken Fehler in einem System, welche wiederum das Fehlverhalten eines Dienstes verursachen (siehe Abbildung 4). Um Fehlertoleranz zu

gewähren, müssen also Fehler so gekapselt werden, dass das Fehlverhalten eines Dienstes verhindert werden kann.
Betrachtet man die Kommunikation zwischen Komponenten als Dienst, bedeutet das, dass

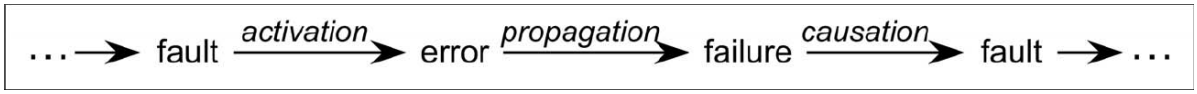

Abbildung 4 Fehlerwirkungskette [2]

eine fehlerhafte Komponente die Kommunikation der anderen Komponenten nicht beeinflussen darf. Ein fehlertolerantes Kommunikationsprotokoll muss also dafür sorgen, dass eine korrekte Nachrichtenübertragung nicht gestört werden kann. Dies setzt voraus, dass die Nachrichten keine Priorität besitzen, denn sonst könnte eine hochpriore Nachricht die Übertragung niederpriorer Nachrichten verhindern. Um eine Überlastung des Busses, und damit einen Nachrichtenverlust zu verhindert, muss dafür gesorgt werden, dass zu keiner Zeit mehrere Nachrichten gesendet werden können. Diese beiden Kriterien lassen sich mit dem TDMA Verfahren, bei dem jeder Komponente Zeitschlitze für den Buszugriff zugeteilt werden, lösen. Eine weitere Notwendigkeit besteht darin, zwei redundante Kommunikationskanäle zu benutzen, um einen Fehler auf einem Kanal kompensieren zu können. Außerdem müssen die Komponenten den fail-silence Modus unterstützen. Dadurch wird sichergestellt, dass eine Komponente nach der Entdeckung eines permanenten Fehlers alle damit verbundenen Dienste einstellt. Die fehlerhafte Komponente wird somit von allen Teilnehmern erkannt, und es können passende Reaktionen gestartet werden.
Die Gruppierung mehrerer Komponenten zu einer Fault-Tolerant-Unit kann die Fehlertoleranz zusätzlich steigern[1]. Eine FTU garantiert Fehlertoleranz nach Außen. Um dies zu gewährleisten, können die Komponenten der FTU die Aufgaben untereinander aufteilen. Das bedeutet, dass die Aufgaben einer ausgefallenen Komponente von einer anderen Komponente in der FTU übernommen werden. Hierfür ist es nötig, dass Ausfälle erkannt werden, was durch fail-silence oder redundante Berechnungen realisiert werden kann.

Das Time-Triggered Protokoll FlexRay

Das FlexRay Protokoll wird von dem FlexRay Konsortium als frei zugängliche und frei nutzbare Spezifikation entwickelt, um sich möglichst schnell zu einem weit verbreiteten Standard zu etablieren. FlexRay ist ein offenes, skalierbares, und deterministisches Protokoll [3] [4] [6].
Das seit 1987 von Kopetz für verlässliche Systeme entwickelte, zeitgesteuerte Protokoll TTP/C war die Grundlage für die Arbeit des Konsortiums [1]. Das ausschließlich zeitgesteuerte TTP/C genügte den Anforderungen der Automobilindustrie nicht, da es keine eventgesteuerte Kommunikation zulässt. Das FlexRay Protokoll ist vom TTP/C abgeleitet, der größte Unterschied liegt in der von FlexRay unterstützten dynamischen Kommunikation. Da das TTP/C vor FlexRay kommerziell erhältlich war, wird es seit dem Jahr 1998 in sicherheitskritischen Einsatzgebieten wie zum Beispiel Flugzeugen eingesetzt.
Das FlexRay Konsortium wurde im Jahr 2000 von den Firmen BMW, Daimler-Chrysler, Phillips und Freescale, mit dem Ziel gegründet, eine neue Kommunikationstechnologie zu entwickeln, um den steigenden technischen Anforderungen an die Buskommunikation in Fahrzeugen gerecht zu werden. Noch im Jahr 2000 schlossen sich Bosch und General Motors dem Konsortium an. Ford, Mazda, Elmos und Siemens-VDO entschieden sich 2002 dem Konsortium beizutreten. Mit der Entscheidung von VW/Audi im Jahr 2003 sich dem Konsortium anzuschließen waren nun fast alle großen Automobilhersteller vertreten, so dass im Jahr 2004 auch Toyota, Honda, Nissan, Renault und PSA-Peugeot-Citroën einstiegen.

Medienzugriff

Die Kommunikation in einem FlexRay-System erfolgt nach dem TDMA bzw. FTDMA Verfahren [4]. Der Buszugriff erfolg dabei anhand eines vorab bestimmten Buszyklus, der sich zyklisch wiederholt. Dabei kann jeder Busteilnehmer zu einem bestimmten Zeitpunkt den Bus exklusiv nutzen. Um einen exklusiven Zugriff zu gewährleisten, müssen die Zeiten der Teilnehmer abgeglichen werden. Dies wird vom Protokoll bereitgestellt, indem die lokalen Uhren am Ende jedes Kommunikationszyklus abgeglichen werden.
Ein Kommunikationszyklus besteht aus einem statischen- und einem dynamischen Segment (vgl. Abbildung 5). Im statischen Segment werden Nachrichten nach dem TDMA Verfahren zu den festgelegten Zeitpunkten in den static-slots übertragen. Das Format der Nachricht ist wie auch die Dauer der static-slots vorher festgelegt.
In dem dynamischen Segment können Nachrichten übertragen werden die nicht vorher festgelegt wurden, dies geschieht nach dem FTDMA Verfahren. Dies unterscheidet sich vom TDMA Verfahren dadurch, dass nur die Reihenfolge des Buszugriffs vorweg geregelt ist, die

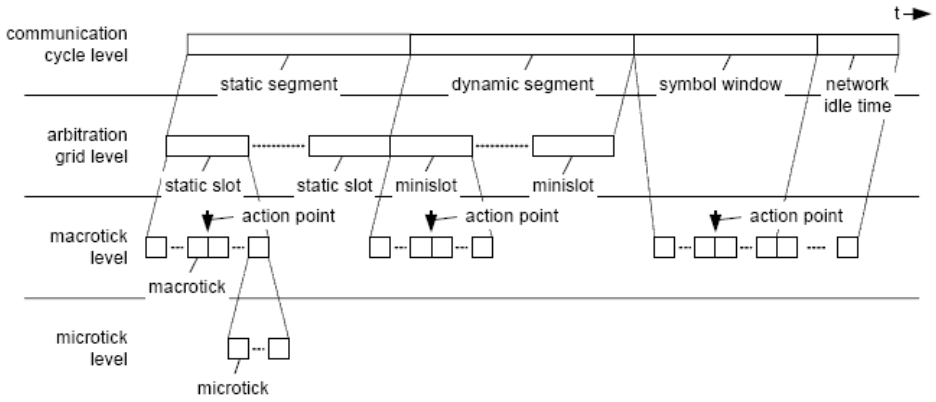

Abbildung 6 Timing im FlexRay Kommunikationszyklus [4]

Länge der Übertragung ist offen. Dies ist bei FlexRay so umgesetzt, dass das Segment in den Teilnehmern zugeteilten Minislots aufgeteilt, welche für die Übertragung genutzt werden können. Die Minislots werden, beginnend bei dem ersten, von allen Komponenten durchgezählt. Möchte eine Komponente eine Nachricht übertragen, tut sie das in dem ihr zugeteilten Slot. Die anderen Knoten bemerken die Übertragung (dies wird durch die minimale Länge des Minislots sichergestellt) und warten mit dem Weiterzählen, bis die Übertragung beendet ist. Da ein Kommunikationszyklus immer die gleiche Länge hat, werden bei der Verschiebugn der Minislots aufgrund des Versands einer Nachricht später folgende Minislots abgeschnitten. Somit bestimmt die Reihenfolge der Minislots die Priorität der zugewiesenen Komponenten. Die Übertragung einer Nachricht im dynamischen Segment ist somit nicht garantiert.
Jeder Static-Slot und jeder Minislot besteht aus einer festen Anzahl von Macroticks, je nach ihrer Zeitdauer. Ein Macrotick ist die kleinste globale Zeiteinheit und besteht aus mehreren Microticks, der kleinsten lokalen Zeiteinheit (vgl.

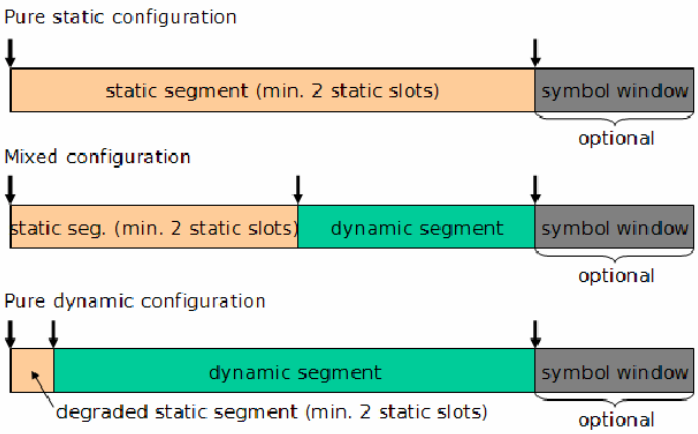

Abbildung 5 Verschiedene Konfigurationsarten des FlexRay Buszyklus [4]

Uhrensynchronisation). Das auf das statische und dynamische Segment folgende Symbol-Window ist für Netzwerkmanagementfunktionen reserviert. In der Network-Idle-Time findet keine Datenübertragung statt. Dadurch haben die Komponenten ein freies Zeitintervall zur Korrektur ihre lokalen Uhren.

Das dynamische Segment ist optional, genau wie das Symbol-Window, das statische Segment muss aus mindestens 2 Static-Slots bestehen, um die Uhrensynchronisation zu gewährleisten. Die sich dadurch ergebenden drei verschiedene Konfigurationsarten eines Kommunikationszyklus zeigt Abbildung 6.

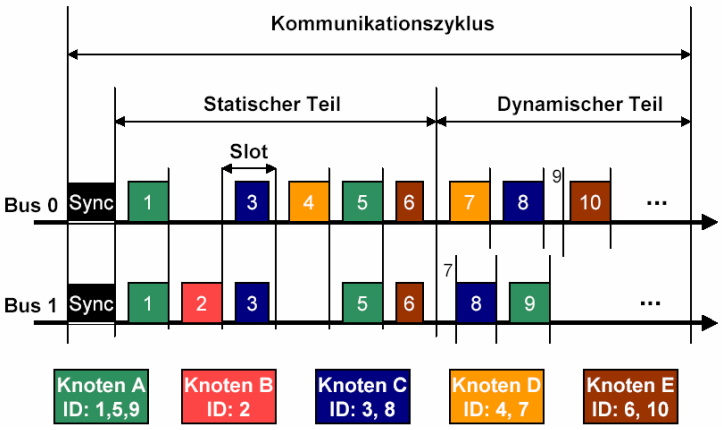

Abbildung 7 Beispiel eines FlexRay Buszyklus [4]

Der Zeitplan eines FlexRay Systems, der die Übertragungszeitpunkte der Nachrichten und damit den Buszyklus vorschreibt, muss auf alle Komponenten abgestimmt und für alle Komponenten identisch sein. Dadurch ist ein höherer Spezifikationsaufwand erforderlich, welcher aber durch die Zusammensetzbarkeit und Fehlertoleranz des Busses, die möglich gemacht wurde, kompensiert wird. In Abbildung 7 ist beispielhaft ein Kommunikationszyklus auf den zwei Kommunikationskanälen dargestellt. Hier ist die Übertragung von Nachrichten über einen oder beide Kanäle sowie die Verschiebung der Minislots im dynamischen Segment dargestellt.

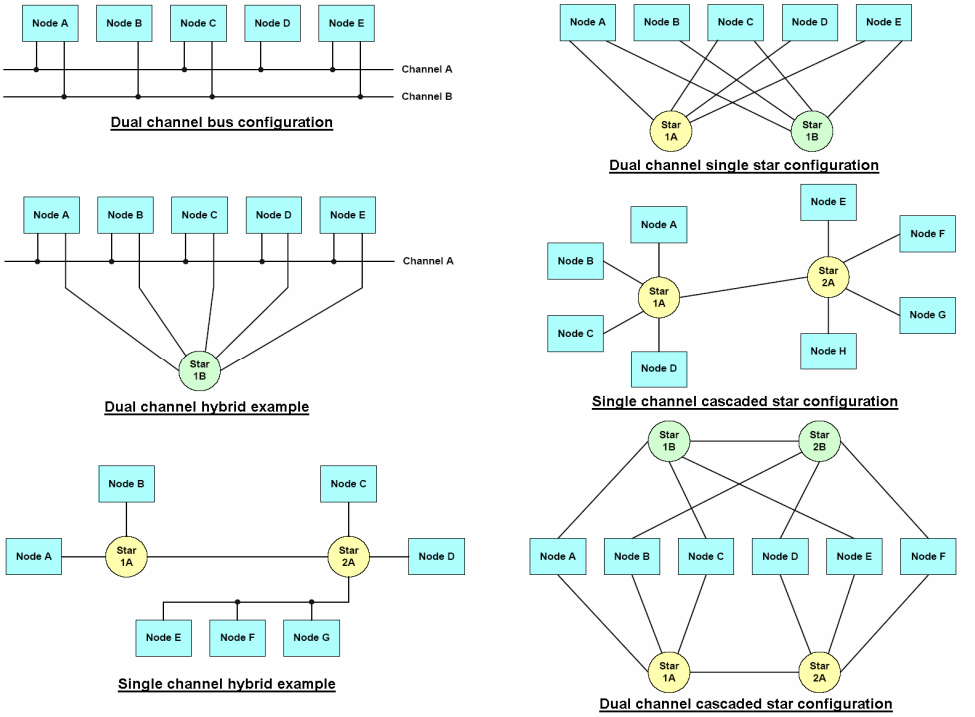

Abbildung 8 Übersicht zu FlexRay Bustopologien [4]

Kommunikationskontroller

Ein Flexray-Controller verfügt über eine oder zwei Busverbindungen. Dies ermöglicht den Aufbau von zwei redundanten Kommunikationskanälen, um den Ausfall eines Kanals zu kompensieren. Das Netzwerk kann im Bus, Stern oder in Mischformen aufgebaut werden. Falls keine Busredundanz benötigt wird, können die zwei Busse auch zur Verdopplung der Bandbreite genutzt werden. (vgl. Abbildung 8)

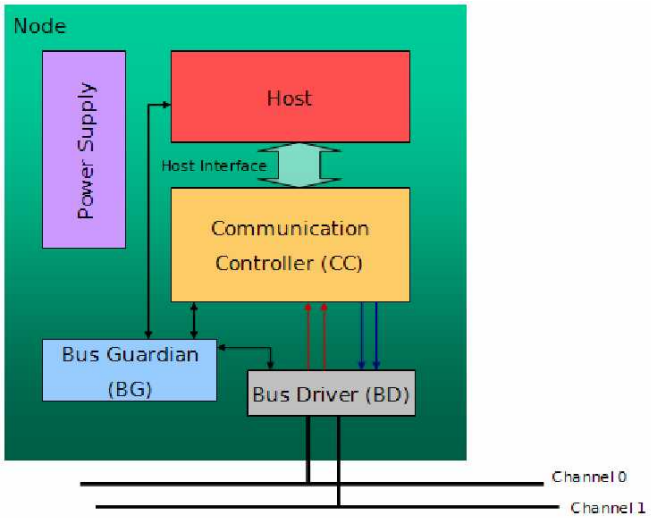

Abbildung 9 Architektur eines FlexRay Knotens [4]

Den Aufbau eines FlexRay Knotens zeigt Abbildung 9. In dem Host wird die Applikationssoftware ausgeführt. Der Kommunikationskontroller stellt dem Host eine Schnittstelle für den Zugriff auf die FlexRay-Signale zur Verfügung. Der Kommunikationskontroller greift über den Bustreiber auf die redundanten Kanäle zu. In Verbindung mit dem Kommunikationskontroller kann ein Buswächter eingesetzt werden. Dieser sorgt dafür, dass der TDMA Zugriff durch den Bustreiber eingehalten wird und kann diesem, wenn erforderlich den Zugriff auf den Bus sperren. Der Buswächter informiert den Host über Fehler im Kommunikationskontroller.

Das FlexRay Nachrichtenformat besteht aus drei Teilen, dem Nachrichtenkopf, den Daten und dem CRC Feld (vgl. Abbildung 10). Der Nachrichtenkopf besteht hauptsächlich aus dem Zykluszähler, welcher den aktuellen Zyklus kennzeichnet, der Länge der Daten in Words und der Nachrichten ID, Länge und ID werden durch einen Header-CRC abgesichert. Das Datenfeld besteht aus maximal 254 Datenbytes. Zur Absicherung der gesamten Nachricht wird ein 24 Bit CRC verwendet.

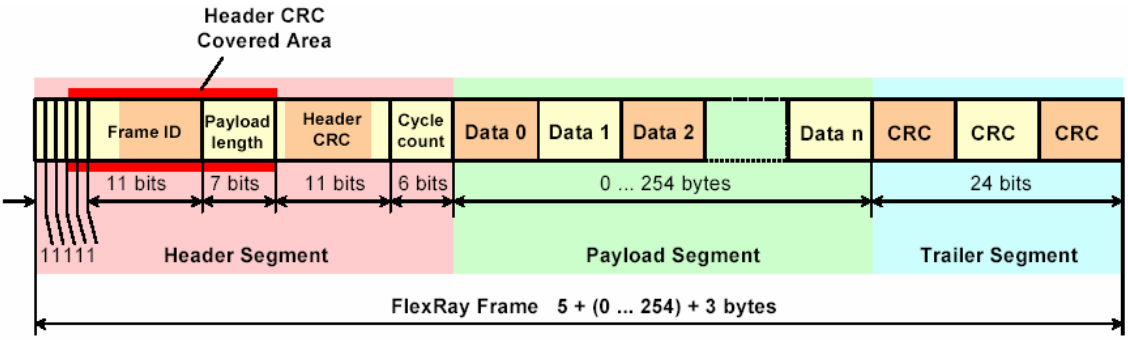

Abbildung 10 FlexRay Nachrichtenformat [4]

Die Geschwindigkeit des FlexRay Busses beträgt pro Kanal minimal 10 MBit/s in späteren Ausbaustufen wird kann die Geschwindigkeit, falls der Bedarf entsteht, erhöht werden. Als Kommunikationsmedium kann sowohl eine elektrische als auch ein optische Verbindung eingesetzt werden.

Uhrensynchronisation

Das FlexRay Protokoll beinhaltet einen verteilten Synchronisationsalgorithmus zum ermitteln der globalen Uhrzeit. Der Algorithmus wird auf allen beteiligten Komponenten im Kommu-

nikationskontroller und im Buswächter ausgeführt. Für die Synchronisation der lokalen Uhren wird in der globalen Zeit in Makroticks gezählt. Ein globaler Makrotick besteht dabei aus unterschiedlich vielen lokalen Ticks den Mikroticks (vgl. Abbildung 5). Die Anzahl der Mikroticks, die einen Makrotick formen, ist abhängig von der Taktfrequenz der lokalen Uhr eines Knotens und wird bei der Uhrensynchronisation angepasst.
Der Synchronisationsalgorithmus misst die Abweichung der eigenen lokalen Makroticks von den anhand der Static-Slots empfangenen lokalen Makroticks der anderen Busteilnehmer und bildet daraus eine durchschnittliche Abweichung. Aus dieser Abweichung wird die für den nächsten Zyklus verwendete Anzahl an Mikroticks berechnet, die einen Makrotick bilden. In der Network-Idle-Time am Ende eines Kommunikationszyklus werden die Korrekturen vorgenommen.
Das Netz kann so konfiguriert werden, dass nur die geringste Abweichung zur Berechnung der Anzahl der Mikroticks verwendet wird, um auszuschließen, dass ein defekter Knoten die Uhrensynchronisation verfälscht.

Buswächter

In einem FlexRay System wurde eine Möglichkeit geschaffen, den Buszugriff einer Komponente redundant abzusichern. So können zeitliche Fehler der Komponente vom Bus ferngehalten werden. Dazu kann in jedem FlexRay-Knoten ein Buswächter eingesetzt werden. Dieser verfügt über eine eigene lokale Uhr, die mit dem Synchronisationsalgorithmus an die globale Uhr angepasst wird, und überwacht anhand dieser den Buszugriff des Knotens. Dies geschieht so, dass der Bus-Treiber nur dann auf den Bus zugreifen darf, wenn der Buswächter die Erlaubnis dazu gibt. So kann verhindert werden, dass zeitliche Fehler des Knotens die Kommunikation der anderen Busteilnehmer stören. In einem sternförmigen Netzwerk kann der Buswächter auch im Sternpunkt eingesetzt werden. Dies reduziert zwar die Hardwarekosten, bei einem defekten Buswächter kann so aber die gesamte Kommunikation gestört werden.

Vergleich von Kommunikationstechnologien im KFZ

In diesem Abschnitt wird FlexRay mit anderen verfügbaren Kommunikationstechnologien verglichen [9]. TTP/C und TTCAN sind zwei weitere Technologien für Hard-Real-Time Sys-

	FlexRay	**TTP/C**	**TTCAN**	**CAN**	**LIN**	**MOST**
Anwendung	Hard-Real-Time-Systeme	Hard-Real-Time-Systeme	Hard-Real-Time-Systeme	Soft-Real-Time-Systeme	Low -Level-Kommunikation	Multimedia Telematik
Steuerung	Multi-Master	Multi-Master	Multi-Master	Multi-Master	Single-Master	Multi-Master
Nachrichten-übertragung	synchron asynchron	synchron asynchron	synchron asynchron	asynchron	synchron	synchron asynchron
Buszugriff	zeitgesteuert eventgesteuert	zeitgesteuert	zeitgesteuert eventgesteuert	eventgesteuert	eventgesteuert	zeitgesteuert eventgesteuert
	TDMA FTDMA	TDMA	TDMA plus Arbitrierungs-fenster für CSMA/CA	CSMA/CA	Polling	TDMA CSMA/CA
Bandbreite	10 MBit/s	25 MBit/s	1 Mbit/s	1 Mbit/s	20 kBit/s	25 MBit/s
Datenbyte pro Frame	0 - 254	0 - 236	0 - 8	0 - 8	2, 4 oder 8	0 - 60
Redundante Kanäle	2 Kanäle	2 Kanäle	nicht unterstützt	nicht unterstützt	nicht unterstützt	nicht unterstützt
Fehler-erkennung	CRC 3 Byte	CRC 3 Byte	CRC 15 Bit	CRC 15 Bit	Checksumme 8-Bit	CRC 4 Byte
Physical Layer	Optisch Elektrisch	Optisch Elektrisch	Elektrisch	Elektrisch	Elektrisch	Optisch

Abbildung 11 Vergleich verschiedener Bussysteme

teme und CAN eine Technologie für Soft-Real-Time Systeme. Aus dem Bereich der Fahrzeugkommunikation kommen noch der LIN Bus für Sensorkommunikation und der MOST Bus für Multimediaübertragungen hinzu. Abbildung 11 zeigt zur Übersicht eine begrenzte Auswahl von Eigenschaften der verschiedenen Technologien.
Es ist deutlich zu erkennen, dass FlexRay und TTP/C sehr ähnliche Technologien sind, ein wichtiger Vorteil von FlexRay ist das dynamische Segment in dem die eventgesteuerte Kommunikation stattfindet. Die niedrigere Bandbreite von Flexray sieht das Konsortium bisher als ausreichend.
Im Vergleich zu TTCAN hat FlexRay deutliche Vorteile. Dies liegt daran, das TTCAN eine Softwarelösung basierend auf dem CAN-Bus ist, wodurch die Nachteile des CAN-Busses nur schwierig kompensiert werden konnten. Dies wird unter anderem durch die nicht vorhandene Unterstützung von redundanten Kanälen, die kleine Nachrichtengröße und die geringe Bandbreite deutlich.

Softwareentwurf in FlexRay Systemen

Der Softwareentwurf eines FlexRay Systems ist aufgrund des TDMA Verfahrens aufwändiger als bei herkömmlichen Kommunikationssystemen, da der Zeitplan für den Kommunikationszyklus berechnet werden muss [7]. Hierfür muss jegliche Kommunikation die im Betrieb des Systems auftreten kann bekannt sein, nachträgliche Änderungen erfordern ein erneutes Berechnen des Zyklus und eine Softwareänderung aller Komponenten.
Der Auftraggeber für das Netzwerk muss den Kommunikationszyklus entwerfen und diesen den Lieferanten der einzelnen Komponenten vorlegen. Eine zeitlich optimale Vorgehensweise wäre hier, dass der Auftraggeber dem Lieferanten, nach Analyse der Funktionen und Aufteilung der Aufgaben auf die Komponenten, die Komponentenspezifikation als erstes, und nach der Konfiguration des gesamten Netzwerks und der Erstellung des TDMA-Zeitplan, diesen erst im späteren Verlauf des Projektes übergibt (vgl. Abbildung 12). Dadurch kann ein Teil der Arbeiten parallelisiert werden [6].
Als Basis für eine FlexRay Komponente bietet sich ein zeitgesteuertes Betriebssystem wie zum Beispiel OSEKtime an [10]. In zeitgesteuerten Betriebsystemen wird der Taskaufruf durch einen Timer-Interrupt gesteuert, da sonst keine Interrupts zugelassen sind vermeidet dieses Komzept, dass innerhalb der Komponenten zu große Latenzen und Jitter wie z. B. durch ereignisgesteuerte Tasks entstehen. Nur mit einem zeitgesteuertem Betriebsystem lassen sich die Hard-Real-Time Anforderungen zur Berechnung der zu übertragenden Werte in den durch FlexRay vorgeschriebenen Zeitabständen garantiert einhalten.

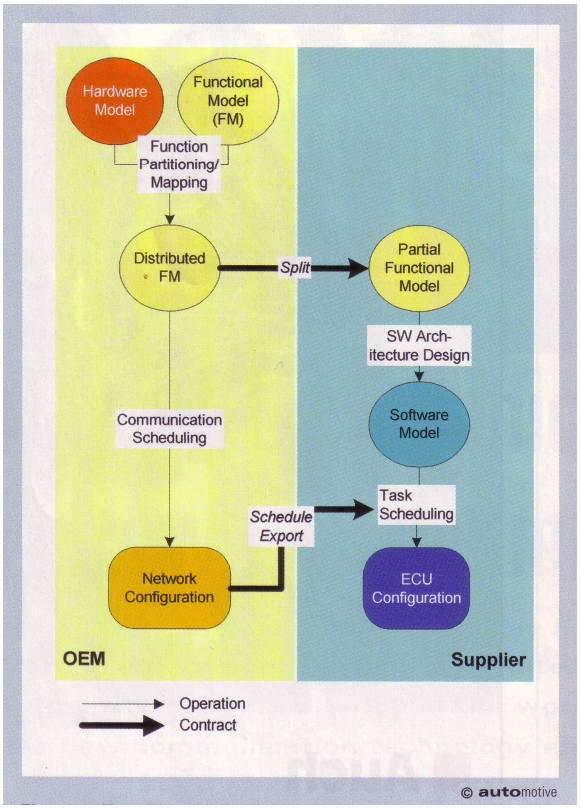

Abbildung 12 Parallelität im FlexRay Entwicklungsprozess [6]

Ausblick

Im Jahr 2006 soll die Serienproduktion des ersten Fahrzeugs mit FlexRay Technologie bei BMW anlaufen. In ihm wird der FlexRay-Bus erst einmal als Backbone verwendet. Im folgenden Jahr soll eine weitere Serien starten die, dann schon mehr als doppelt so viele FlexRay-Knoten enthält. In Zukunft werden bei allen Automobilherstellern des Konsortiums die Oberklassefahrzeuge mit FlexRay-Systemen ausgerüstet.
Durch den zunehmenden Einsatz der FlexRay-Technologie in Serienfahrzeugen werden die Kosten für FlexRay-Produkte sinken, wodurch diese auch für Mittelklassewagen erschwinglich werden.

Thema für eine Masterarbeit

In der Projektarbeit des Masterstudiengangs 2005-2006 wird ein fahrerloses Transportsystem (FAUST) entwickelt, welches über einen Leitstand bedienbar ist. Das Fahrzeug beinhaltet mehrere verteilte Komponenten, die für die Regelung der Antriebs- und Lenkmotoren, Koordinierung der Komponenten und Navigation des Fahrzeugs zuständig sind (vgl. Anhang).
In der Masterarbeit soll die sich bisher auf dem Fahrzeug befindenden ARM-Komponenten durch ein FlexRay-System ersetzt werden. Die Regelung der Motoren wird anschließend eine zentrale FlexRay-Komponente übernehmen, um die Vorteile des minimalen Jitters der zeitgesteuerten Kommunikation zu erproben. Die zweite FlexRay Komponente führt die Beschaffung der Istwerte und das Stellens der Sollwerte für den vorderen Fahrzeugteil durch. Zusätzlich dazu soll diese die Funktionalität des Koordinierungsrechners und das in einer anderen Masterarbeit entwickelte Kollisionsvermeidungssystem beinhalten (vgl. Abbildung 13). Auf

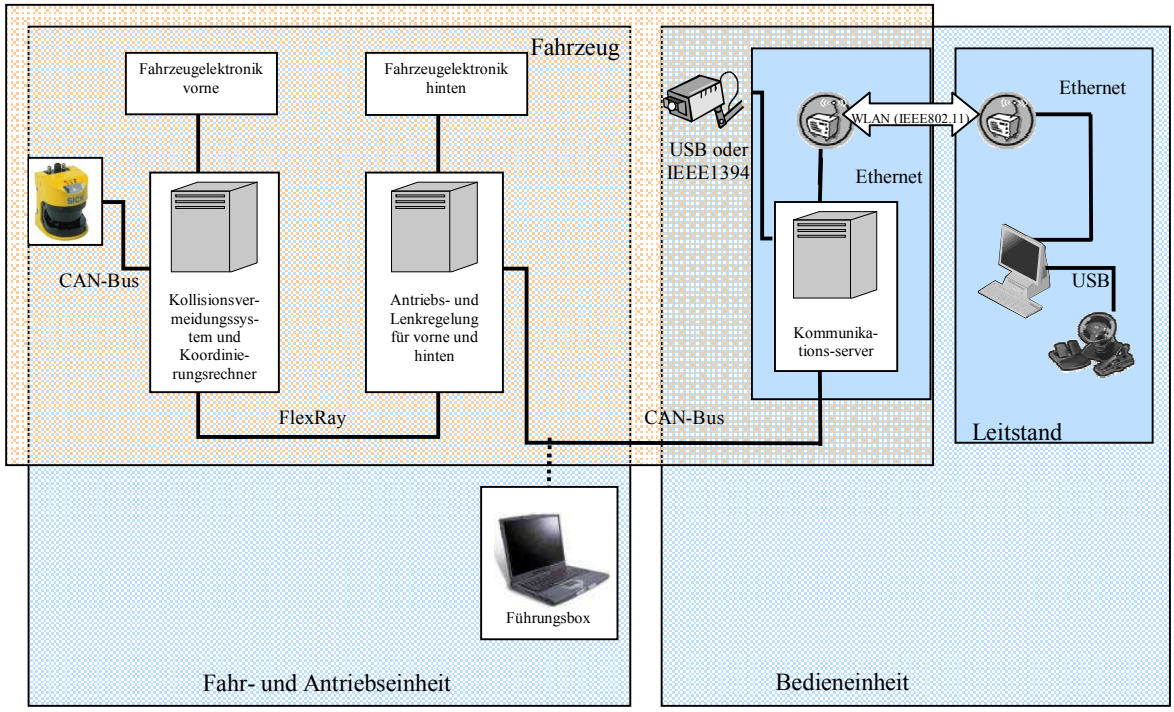

Abbildung 13 Systemübersicht des fahrerlosen Transportsystems

den FlexRay-Komponenten soll als zeitgesteuertes Betriebssystem ein Softwarekonzept basierend auf einem timergesteuerten Dispatcher zum Einsatz kommen [11]. Als FlexRay-Komponenten werden die für die Automobilindustrie gefertigten DECOMSYS::NODE<RENESAS> eingesetzt. Diese beinhalten einen 24MHz 32bit Prozessor, Schnittstellen für CAN und FlexRay sowie diverse Ein- und Ausgänge. Die dadurch im Ge-

gensatz zu den vorherigen Komponenten gegebenen Hardwareeinschränkungen, besonders im Vergleich zu dem Koordinierungsrechner müssen kompensiert werden.
Zusätzlich zu dieser praktischen Herangehensweise wird in der Masterarbeit eine von Firma Decomsys angeregte Themenstellung erarbeitet.

Dafür stehen folgende Themen zur Auswahl:

- **Simulation eines FlexRay Treibers zur Validierung eines signalbasierten Kommunikationslayers**

  In der Firma Decomsys werden auf Basis eines aktuellen FlexRay-Treibers weitere Softwaremodule entwickelt. Für deren Entwicklung und Test wird FlexRay-Hardware benötigt. Ziel der Arbeit ist eine PC-Simulation des FlexRay-Treibers zu entwickeln, sodass für die Entwicklung der Softwaremodule keine Hardware mehr benötigt wird. Die Software soll in der Programmiersprache C entwickelt werden.

- **Erstellung und Test verschiedener Konzepte für einen Kommunikationstask-Scheduler für ein FlexRay Konfigurationstool**

  Bei der Softwareentwicklung einer FlexRay Komponente muss der Steuergeräteentwickler den Schedule seiner zeitgesteuerten Tasks anhand des FlexRay Schedules entwerfen. Ziel der Arbeit ist es diesen Vorgang zu automatisieren und in ein führendes FlexRay Konfigurationstools zu integrieren. Dazu sollen unterschiedliche Konzepte für ein automatisiertes Scheduling der Tasks erstellt, geprüft und miteinander verglichen werden.

Glossar

Latenz	Verzögerungszeit eines Ereignisses zur Reaktion der Verarbeitungseinheit
Jitter	Schwankungen in der Latenz
X-by-wire	Verlegung von mechanischen Übertragungswegen in elektrische Funktionen
TTP	Time Triggered Protocol
ABS	Anti-Blockier System zur Verhinderung von blockierenden Reifen beim Bremsen des KFZ
ASR	Anti-Schlupf Regelung zur Verhinderung von durchdrehenden Reifen beim Anfahren des KFZ
ESP	Elektronische Stabilitätsprogramm zum Verhindern des seitlichen Ausbrechens des KFZ
TÜV	Technische Überwachungsverein
FTU	Fault-Tolerant-Unit
TDMA	Time Division Multiple Access; Zeitmultiplexverfahren bei welchem der Zugriff auf ein Medium in den Komponenten zugewiesene Zeitschlitze von fester Länge eingeteilt wird
FTDMA	Flexible Time Division Multiple Access; Zeitmultiplexverfahren basierend auf TDMA, bei dem die Zeitschlitze eine variable Länge besitzen
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CRC	Cyclic Redundancy Check
OSEKtime	zeitgesteuertes Betriebsystem der Firma 3soft

Literaturverzeichnis

[1] H. Kopetz:
Real-Time Systems - Design Principles for Distributed Embedded Applications, Kluwer Academic 2. Printing 1998

[2] A. Avižienis, J.-C. Laprie, B. Randell, C. Landwehr:
Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE 2004

[3] D. Millinger, R. Nossal:
FlexRay Communication Technology

[4] FlexRay Consortium:
FlexRay Communications System Protocol Specification Version 2.1, 2005

[5] Robert Bosch GmbH:
Fachwissen KFZ-Technik Fahrstabilisierungssysteme, Gelbe Reihe 2004

[6] Automotive electronics + systems:
Special Edition Flexray, Hanser 2004

[7] B. Kleinjohann, G. R. Gao, H. Kopetz, L. Kleinjohann, A. Rettberg:
Design Methods and Applications for Distributed Embedded Systems, Kluwer Academic 2004

[8] Mitteilung der Kommission an den Rat und das Europäische Parlament:
Informations- und Kommunikationstechnologien für sichere und intelligente Fahrzeuge, (SEK(2003) 963)

[9] TTTech Computertechnik AG:
CAN/TTCAN – Byteflight – FlexRay – TTP Technical comparison of protocol properties with a focus on safety-related applications, 2004

[10] M . Homann:
OSEK Betriebssystem-Standard für Automotive und Embedded Systems, mitp-Verlag, 2. Ausgabe 2005

[11] M. J. Pont:
Patterns for time-triggered embedded systems, Addison-Wesley 2001

Anhang

Systemarchitektur für das Projekt FAUST

Inhaltsverzeichnis

Abbildungsverzeichnis

Gesamtkonzept

Das FAUST System wurde in drei Bereiche aufgeteilt:

- Die Bedieneinheit besteht aus dem Leitstand und dem Kommunikationsserver auf dem Fahrzeug, sowie den Kommunikationsendpunkten und der Kamera.
- Die Fahreinheit besteht aus dem Koordinierungsrechner auf dem Fahrzeug und der Führungsbox, welche optional über den CAN-Bus angeschlossen werden kann.
- Die Antriebseinheit besteht aus der Fahrzeugelektronik und zwei ARM-Boards, welche die Ansteuerung der Fahrzeugelektronik übernehmen.

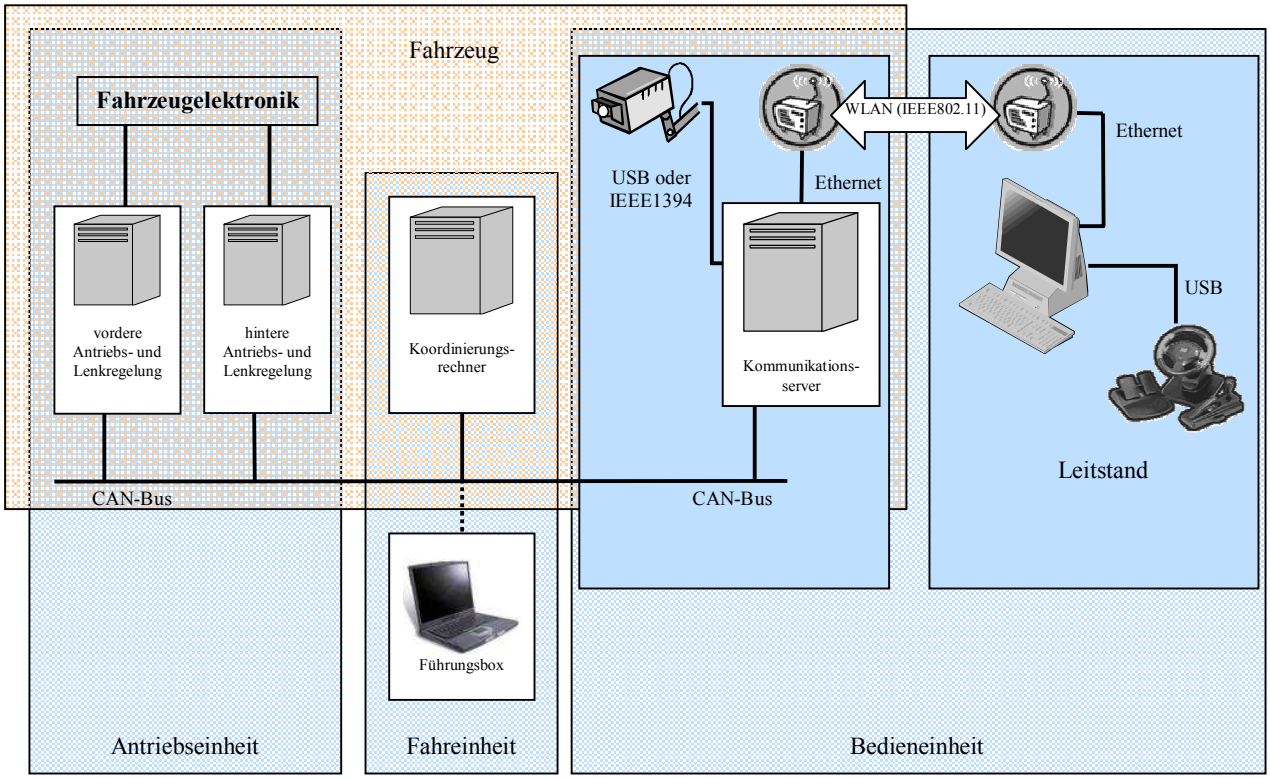

Abbildung 14 Schematische Darstellung der Kommunikationsteilnehmer

In diesem Dokument werden die Aufgaben und Schnittstellen der einzelnen Systemkomponenten erläutert.

Bedieneinheit

Auf Basis der Überlegungen des letzten Semesters, haben wir uns für die Realisierung eines über WLAN angebundenen Leitstandes entschieden.
Für die Funkanbindung war WLAN die einzige Möglichkeit, in einer späteren Ausbaustufe eine Fahrzeug-Zu-Fahrzeug-Kommunikation realisieren zu können.
Die Bedieneinheit gliedert sich in zwei Komponenten:

1. den Leitstand, der die Mensch-Maschine-Schnittstelle darstellt
2. den Kommunikationsserver, der die Schnittstelle der Bedieneinheit zu den anderen Komponenten des Fahrzeugs realisiert

Dieser Sachverhalt wird in Abbildung 14 nochmals veranschaulicht.

In der ersten Ausbaustufe soll die Bedieneinheit wie folgt umgesetzt werden:

1. Realisierung eines Leitstandes ohne Integration des Videobildes
2. Eingabe der Steuerinformationen mittels Lenkrad
3. Programm für die Umsetzung von CAN ó WLAN auf dem Kommunikationsserver
4. Datenübertragung Leitstand ó Kommunikationsserver mittels eigenem Anwendungsprotokoll
5. Videobildübertragung mittels einer netzwerkfähigen Kamera mit integriertem Videostreamserver, alternativ ein Videostreamserver auf dem Kommunikationsserver
6. Videobilddarstellung auf dem Leitstand durch Browser

Im Folgenden werden nun Leitstand und Kommunikationsserver feiner spezifiziert.

Design des Leitstandes

Hardware

Als Plattform für den Leitstand soll ein Standard- PC mit WLAN- Interface dienen. Letzteres kann ein USB- Dongle oder ein WLAN- Router sein.

Software

Die Realisierung des Leitstandes soll unter LabView erfolgen. Ausschlaggebend für die Verwendung von LabView, statt einer herkömmlichen Programmiersprache (z.B. Java), ist das Vorhandensein der umfangreichen Bibliotheken, die den Programmieraufwand gering halten.

Prinzipieller Aufbau

Aus dem Anforderungskatalog lassen sich folgende Funktionsblöcke für die Bedieneinheit und den Leitstand ableiten:

1. Initialisierung (Plug and Play)
2. Videobildanzeige
3. Einlesen von Steuerinformationen
4. Drahtlose Kommunikation mit dem Fahrzeug
5. Visualisierung der Prozessvariablen
6. Alarmmanager

7. Datenbank basierter Prozessdatenlogger

Die nächsten Abschnitte dienen dazu, diese Blöcke detaillierter zu beschreiben.

Initialisierung (Plug and Play)

Wird der Leitstand mit einem Fahrzeug verbunden, so muss sich die Anwendung an das Fahrzeug anpassen. Die fahrzeugspezifischen Daten werden beim Anmeldevorgang vom Fahrzeug übermittelt. Die fahrzeugspezifischen Daten setzen sich aus folgenden Parametern zusammen:

- Fahrzeugname
- Lenkwinkelbereich
- Geschwindigkeitsbereich
- Vorderachse lenkbar
- Hinterachse lenkbar

Siehe Anforderungskatalog Punkt 2.10

Videobildanzeige

Für die Anzeige des Videobildes wird im ersten Anlauf ein Browser verwendet, der in der Lage ist (evtl. über ein geeignetes PlugIn), einen Videostream darzustellen.

Siehe Anforderungskatalog Punkt 2.5

Einlesen von Steuerinformationen

Diese Komponente, im Folgenden Lenkradkomponente genannt, ist der Programmteil, welcher Steuerinformationen (Radfeststellung, Richtungs- und Geschwindigkeitsvorgabe) eines PC-Lenkrades einliest und diese der Kommunikationskomponente zur Verfügung stellt.

Siehe Anforderungskatalog Punkte 2.1, 2.2, 2.3

Drahtlose Kommunikation mit dem Fahrzeug

Die Kommunikationskomponente muss die Steuerinformationen von der Lenkradkomponente entgegennehmen, in noch zu spezifizierende Pakete umwandeln und an den Kommunikationsserver (auf dem Fahrzeug) versenden.
Des Weiteren muss die Kommunikationskomponente die vom Kommunikationsserver kommenden Pakete:

- entgegennehmen
- die übertragenen Daten in die plattformabhängige Darstellung überführen
- gegebenenfalls weiterleiten an Visualisierungskomponente und/oder Prozessdatenlogger

Eine Kommunikationsunterbrechung zum Fahrzeug ist sofort zu signalisieren und für eine Dauer von maximal 30 Sekunden zu tolerieren. Danach muss der Leitstand mit dem Fahrzeug erneut verbunden werden.
Für die Überwachung der Bedieneinheit ist periodisch eine KeepAlive-Message zu senden. Diese muss vom Kommunikationsserver erzeugt und an den Leitstand gesendet werden. Der Leitstand reflektiert diese spezielle Meldung, so dass sie wiederum vom Kommunikationsserver empfangen wird. Der behandelt diese Meldung wie jede andere, d.h. er gibt sie auf dem CAN-Bus aus. Somit ist jeder der anderen Kommunikationspartner in der Lage, den Ausfall der Bedieneinheit zu bemerken und kann gegebenenfalls darauf reagieren. Diese Methode

stellt sicher, dass auch der Ausfall eines Teils der Bedieneinheit (Kommunikationsserver, Leitstand, Funkverbindung) detektiert wird.

Siehe Anforderungskatalog Punkt 2.4, 2.8

Visualisierung der Prozessvariablen

Der Leitstand hat die Aufgabe, ausgewählte vom Kommunikationsserver kommende Prozessvariablen zu visualisieren. Zu den ausgewählten Prozessvariablen gehören:

1. Ist-Geschwindigkeit Vorderrad
2. Ist-Geschwindigkeit Hinterrad
3. Ist-Lenkwinkel Vorderrad
4. Ist-Lenkwinkel Hinterrad
5. Batteriefüllstand

Anstelle der gewünschten Motordrehzahlen, die nicht direkt zur Verfügung stehen, aber linear abhängig von den Ist-Geschwindigkeiten der Räder sind, werden die beiden Ist-Geschwindigkeiten der Räder dargestellt.

Siehe Anforderungskatalog Punkt 2.6

Alarmmanager

Der Alarmmanager soll vom Fahrzeug/Leitstand gemeldete Alarme mit der Zeit des Eintreffens auf dem Leitstand in einem Textfeld anzeigen.

Siehe Anforderungskatalog Punkt 2.9

Prozessdatenlogger

Der Prozessdatenlogger speichert die ihm übergebenen Daten in einer Datenbanktabelle mit zugehöriger CAN-ID und Empfangszeit.

Alternative:
Der Prozessdatenlogger kann die Daten zeilenweise, als Semikolon separierte Werte in einer Datei ablegen. Eine Zeile besteht dann nur aus druckbaren Zeichen und hat den folgenden Aufbau:

Zeitstempel	**;**	CAN-ID (hexadezimal)	**;**	Wert (dezimal)	**CR/LF**

Abbildung 15 Format der Logdatei

Die erste Zeile in der Datei ist gleichen Formats, enthält aber die Feldnamen statt der Werte. Dieses Dateiformat kann sowohl von MS-Access, als auch von MS-Excel importiert werden. Damit können diese Programme zu Analysezwecken verwendet werden.

Siehe Anforderungskatalog Punkt 2.7

Design des Kommunikationsservers

Hardware

Für die Realisierung des Kommunikationsservers kommt ein GEME- Rechner mit Ethernet- und CAN- Interface zum Einsatz. Um den erhöhten Speicherbedarf gerecht zu werden, ist er mit einer Festplatte auszustatten.
Die Kommunikation mit dem Leitstand erfolgt über ein WLAN- Access Point, der auf dem Fahrzeug zu installieren ist.
Anzustreben ist der Einsatz einer Ethernet- fähigen Kamera, die über einen integrierten Videostreamserver verfügt.
Alternativ kann ein V4L- (Video for Linux) fähiges Modell verwendet werden. In diesem Fall muss der Videostreamserver auf dem Kommunikationsserver implementiert werden.

Software

Das zugrunde liegende Betriebssystem soll LINUX sein, um bei Bedarf eine Hardwareunterstützung für die Kamera zu haben. Sollte der Videostreamserver auf dem Kommunikationsserver laufen müssen, so lässt sich für LINUX auch ein geeignetes OpenSource Projekt finden.
Des Weiteren ist der zugehörige C-Compiler für die Programmentwicklung zu verwenden.

Plug and Play

Verbindet sich ein Leitstand mit dem Fahrzeug, so muss dieses seine fahrzeugspezifischen Daten beim Anmeldevorgang übermitteln. Die fahrzeugspezifischen Daten setzen sich aus folgenden Parametern zusammen:

- Fahrzeugname
- Lenkwinkelbereich
- Geschwindigkeitsbereich
- Vorderachse lenkbar
- Hinterachse lenkbar

Siehe Anforderungskatalog Punkt 2.10

Umsetzung CAN ó WLAN

Für die Übertragung der Prozessdaten zum und zur Übernahme der Daten vom Leitstand wird ein Programm benötigt. Es muss:

- die CAN- Messages vom Bus lesen, sie in das WLAN- Paketformat wandeln und dieses Paket dann versenden
- umgekehrt die vom Leitstand gesendeten Steuerinformationen in CAN- Messages umwandeln und über den Bus versenden

Siehe Anforderungskatalog Punkt 3.8

Übertragung der Videobilder

Die Videobilder sind als Videostream an den Leitstand zu übertragen. Dies kann mittels eines in der Kamera integrierten oder alternativ mittels eines auf dem Kommunikationsserver laufenden Videostreamserver erfolgen.

Siehe Anforderungskatalog Punkt 3.9

Fahreinheit

Spezifikation Führungsbox

1. Die Führungsbox soll CAN-Nachrichten versenden, die vom Koordinierungsrechner ausgewertet werden. Der Koordinierungsrechner sorgt dafür, dass die CAN-Nachrichten transformiert werden und Steuersignale für die ARM-Boards auf den CAN-Bus gelegt werden.

2. Die Führungsbox wird in zwei Phasen umgesetzt. In der ersten Phase werden die Nachrichten, die vom Koordinierungsrechner ausgewertet werden sollen von einem Notebook mit der CANalyzer Software generiert. In der zweiten Phase wird es ein Box mit einem Joystick geben, der eine eigenen CAN-Controller enthält.

3. Die Signale der Führungsbox werden zyklisch auf den CAN-Bus gelegt. Bleibt ein Signal für längere Zeit aus, wird das Fahrzeug angehalten.

4. Damit Leitstand und Führungsbox nicht zur gleichen Zeit Nachrichten an den Koordinierungsrechner senden, ist durch Initialisierungsroutine dafür zu sorgen, dass die Software nach der Initialisierung nur noch Signale der Führungsbox verarbeitet. Der Fahrbetrieb wird entweder durch eine CAN-Nachricht oder ein digitales Eingangssignal am Koordinierungsrechner von Führungsboxbetrieb nach Leitstandbetrieb bzw. umgekehrt umgestellt.

5. Es soll möglich sein den Fahrmodus zu ändern. Der Fahrmodus kann nur verändert werden, wenn das Fahrzeug im Stillstand ist.

Die Nachrichten die an den Koordinierungsrechner gesendet werden:

Lenken (links,rechts)
Geschwindigkeit(vor,zurück)
Initialisierung(Box,Leitstand)

Die Fahrmodi, die auswählbar sind:

Vorderradlenkung
Hinterradlenkung
synchrone Lenkung
Seitenfahrt

Spezifikation des Koordinierungsrechners

1. Als Hardware für den Koordinierungsrechner steht ein GEME Rechner mit einem QNX Betriebssystem zur Verfügung. Dieser GEME Rechner verfügt über eine Ethernet- und eine CAN Bus Schnittstelle.

2. Er dient sowohl als Schnittstelle/Übersetzer zwischen Kommunikationsserver und ARM Controller, als auch als Schnittstelle/Übersetzer zwischen Führungsbox und ARM Controller. Hierzu generiert der Koordinierungsrechner kontinuierlich CAN Nachrichten aus den Sollwertvorgaben für Geschwindigkeit und Lenkwinkel.

3. Der Koordinierungsrechner hat außerdem die Aufgabe die verschiedenen Fahrmodi zu realisieren. Der Leitstand/Führungsbox geben lediglich den gewünschten Gesamtlenkwinkel vor. Je nach dem in welchen Fahrmodus das Fahrzeug betrieben werden soll, rechnet der Koordinierungsrechner den gewünschten Gesamtlenkwinkel auf Lenkwinkelsollwerte für jedes Rad um und sendet diese an den ARM Controller weiter.

4. Die Übergänge zwischen den Fahrmodi müssen ebenfalls vom Koordinierungsrechner koordiniert und überwacht werden. Die Umschaltung erfolgt über einen Schalter an der Führungsbox. Es soll nur im Stillstand auf einen anderen Fahrmodus umgeschaltet werden können. Wird während der Fahrt auf einen anderen Fahrmodus umgeschaltet, bremst der Koordinierungsrechner das Fahrzeug bis zum Stillstand ab. Anschließend schaltet er in den gewünschten Fahrmodus, und beschleunigt das Fahrzeug wieder auf den vorgegebenen Sollwert.

5. Außerdem empfängt der Koordinierungsrechner die Istwerte für Geschwindigkeit und Lenkwinkel für jedes Rad über den CAN Bus und wertet diese aus. Auch hier müssen die IST-Lenkwinkel für beide Räder in einen Gesamt-IST-Lenkwinkel umgerechnet werden, der dann evtl. dem Leitstand übermittelt werden kann.

6. Die Koordinierung des Übergangs zwischen Führungsboxbetrieb und Leitstandbetrieb ist ein weiteres Problem, welches in den Aufgabenbereich des Koordinierungsrechners fällt. Die Umschaltung erfolgt über einen Schalter am Fahrzeug. Der Status dieses Schalters wird dem Koordinierungsrechner gemeldet. Ist dieser auf Führungsboxbetrieb eingestellt, werden nur die CAN Nachrichten der Führungsbox ausgewertet. Analog dazu werden nur die CAN Nachrichten des Kommunikationsrechners ausgewertet, wenn der Schalter auf Leitstandbetrieb steht. Auch diese Umschaltung soll nur im Stillstand vorgenommen werden dürfen. Wenn die Umschaltung während des Fahrbetriebs vorgenommen wird, bringt der Koordinierungsrechner das Fahrzeug zum Stillstand und verarbeitet erst dann die CAN Nachrichten des gewünschten Steuermoduls.

Antriebseinheit

Hardwarekomponente: ARM-Boards

Die Regelung der Antriebs- und Lenkmotoren soll auf zwei phyCORE-ARM7 der Firma Phytec realisiert werden.
Diese integrieren einen ARM7-Mikrocontroller AT91M55800, 4 Megabyte Flash und 2 Megabyte SRAM. Das Mikrocontroller-Board wird auf dem Development-Board HD200 betrieben, welches die Spannungsversorgung zur Verfügung stellt und alle Schnittstellen nach außen führt (vgl. Bild).

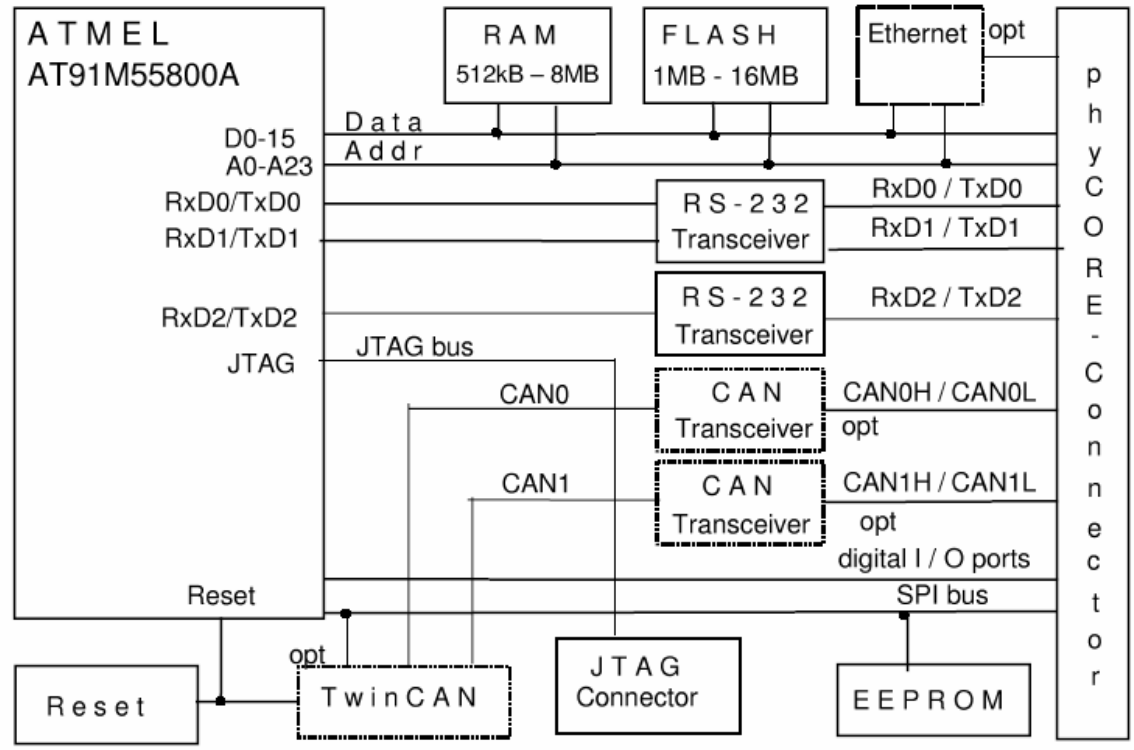

Abbildung 16 Blockdiagramm des Mikrocontroller-Board phyCORE-ARM7

Das Mikrocontroller-Board enthält, neben dem Mikrocontroller, Speicher und diversen Ein- und Ausgängen, einen TwinCAN-Controller 82C900 mit zwei unabhängigen CAN2.0B-Schnittstellen und einem Ethernet-Controller [MAY05].

Schnittstellen zur Fahrzeugelektronik

Da zwei Controller für die Lenk- und Fahrregelung vorgesehen sind, wird je ein Controller die Regelung für ein Antriebsrad übernehmen. Diese Ansteuerung soll über die digitalen Ein- und Ausgänge umgesetzt werden.

Da die Servos (FSA23) der Motoren mit je +/- 10 V angesteuert werden, müssen die digitalen Ausgangssignale erst über einen DA-Umsetzer auf 0 bis 3,3 V und anschließend mit einem Operationsverstärker auf die +/- 10V umgewandelt werden. Zusätzlich zu der Ansteuerung

der Motoren sollen über die Servos Informationen vom Zustand des Motors (Überhitzung, Kurzschluss) über einen digitalen Eingang ausgelesen werden.

Die Information bezüglich der aktuellen Geschwindigkeit des Fahrzeuges und über den Lenkwinkel, soll mit Hilfe zweier Inkrementalgeber, die je an einem Antriebsrad angeschlossen sind, eingelesen werden. Diese liefern über je zwei um 90° phasenverschobene digitale Ausgänge Signale, aus denen die Fahrt- und Lenkrichtung, sowie die Geschwindigkeit interpretiert werden kann. Der Lenkwinkelsensor verfügt zusätzlich über ein Nulllagensignal.

Die Impulse der Inkrementalgeber werden mit Hilfe von einer zusätzlichen Auswertungslogik gezählt.
Diese Auswertung und die Umwandlung der Steuersignale für die Leistungselektronik sind auf einem Hardwareinterface realisiert, welches im Rahmen einer Diplomarbeit entstand [MAY05].

Zusätzlich zu den Informationen der Inkrementalgeber sollen die Endlagenschalter der Lenkmotoren ausgewertet werden.

An dem Fahrzeug ist eine Bumperleiste installiert, die bei Kollision des Fahrzeuges den Motoren die Spannungsversorgung entzieht. Ein digitales Signal über den Zustand der Leiste wird von einem der beiden ARM-Controller ausgewertet.

Für die Umschaltung von Führungsbox bzw. Leitstandbetrieb wird ein Schaltersignal über einen Digitaleingang eines ARM-Controllers eingelesen.

Aufgaben und Funktion der ARM-Controller

- Die ARM-Controller werten die auf dem CAN-Bus liegenden Sollvorgaben für Lenkwinkel und Geschwindigkeit aus, und generieren daraus die geregelten Ausgangssignale für die Leistungselektronik.
- Die aktuellen Informationen über den Fahrzeugzustand werden eingelesen, für die Regelung verwendet und auf den CAN-Bus gelegt. Hierzu gehören:
    - Lenkwinkel der beiden Räder
    - Geschwindigkeit der beiden Räder
    - Motorströme von Lenk und Fahrmotoren
- Für die interne Berechnung der Ausgangssignale werden zusätzlich
    - Nulllagenschalter der Lenkmotoren
    - Endlagenschalter der Lenkmotoren
    - (Batteriestand)
- Aus den Eingangssignalen und internen Zuständen werden Fehlersituationen erkannt und eine entsprechende Fehler-/Infonachricht auf den CAN-Bus gelegt. Hierzu gehören bisher:
    - Batteriespannung zu niedrig
    - Zu große Regelabweichung des Lenkwinkels
    - Zu große Regelabweichung der Geschwindigkeit
    - Bumperleiste ausgelöst
    - Schalterstatus Führungsboxumschalter
- Das Signal der Bumperleiste wird ausgewertet und die Parameter der Regler werden zurückgesetzt.
- Anhand der Endlagensignale der Schalter an den Lenkmotoren werden die Motoren abgeschaltet, damit diese nicht gegen den Anschlag fahren.

- Der Schalter zur Umschaltung zwischen Führungsbox und Leitstandbetrieb wird über einen digitalen Eingang eingelesen und eine entsprechende CAN-Nachricht mit der Information über den Status auf den Bus gelegt.