HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Maria Juliane Clausing

Analyse eines modernen Bussystems für sicherheitskritische Kommunikation im Kraftfahrzeug

2018

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Analyse eines modernen Bussystems für sicherheitskritische Kommunikation im Kraftfahrzeug

Autorin:
Maria Juliane Clausing

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO15w4-B

Erstprüfer:
Prof. Dr. rer. nat. Christian Hummert

Zweitprüfer:
Dipl.-Ing. (FH) Heiko Polster

Mittweida, 27. August 2018

Faculty of **Applied Computer Sciences and Biosciences**

BACHELOR THESIS

Analysis of a modern bus system for security relevant communication in vehicles

Author:
Maria Juliane Clausing

Study Programme:
General and Digital Forensic

Seminar Group:
FO15w4-B

First Referee:
Prof. Dr. rer. nat. Christian Hummert

Second Referee:
Dipl.-Ing. (FH) Heiko Polster

Mittweida, August 27th, 2018

Bibliografische Angaben

Clausing, Maria Juliane: Analyse eines modernen Bussystems für sicherheitskritische Kommunikation im Kraftfahrzeug, 83 Seiten, 50 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2018

Referat

Diese Bachelorarbeit beschäftigt sich mit der Analyse des Bussystems FlexRay. Dieses System wird vor allem für sicherheitskritische Aspekte im Fahrzeug eingesetzt. Somit kann das FlexRay über das Leib und Leben des Fahrers und aller Verkehrsteilnehmer mitwirken. Gerade in heutigen Zeiten, wo immer mehr Technologie zur Sicherheit des Verkehrs beitragen soll, wird vor allem in die Sicherheit des Systems im Bezug zur Zuverlässigkeit geforscht. Die Absicherung vor Manipulationen oder Angriffen auf den Bus wird jedoch nur wenig untersucht. Daher soll diese Ausarbeitung zur Analyse des Bussystems, bezüglich der IT-Sicherheit dienen. Dabei ist die Thematik dieser Arbeit noch nicht abgeschlossen, sie bildet jedoch eine gute Grundlage für weiterführende Versuche.

Abstract

This thesis focuses on the bus system FlexRay. The system is mainly used for security relevant aspects in an automobile. That is the reason why FlexRay could have an impact on the health and life of the driver and other traffic participants. Nowadays, more and more technologies exist in the vehicles for further safety in the traffic. However, the research is just about the safty of a system yet. There is not much research about the security aspects of a system. Hence, this thesis should anaylsis the IT-security of the bus system FlexRay. Nevertheless, the topic of this elaboration is not completed yet, though it forms a good basis for further experiments.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ABS ..... Antiblockiersystem
BD ..... Bus Driver
BLU ..... Back Light Unit
BM ..... Bus-Minus
BP ..... Bus-Plus
BSC ..... Brake and ESP Control
BSI ..... Bundesamt für Sicherheit in der Informationstechnik
CAN ..... Controller Area Network
CAPL ..... Communication Access Programming Language
CAS ..... Car Access System
CAS ..... Collision Avoidance Symbole
CC ..... Communication Controller
CPU ..... Central Processing Unit
CRC ..... Cyclic Redundancy Check
CSMA/CA ..... Carrier Sense Multiple Access Collision Avoidance
D-CAN ..... Diagnose-CAN
DB ..... Datenbank
DDE ..... Digitale Diesel-Elektronik
DLC ..... Data Length Code
DME ..... Digitale Motor-Elektronik
DSC ..... Dynamic Stability Control
ECU ..... Electronic Control Unit
EEPROM ..... Electrically Erasable Programmable Read-only Memory
ESP ..... Elektronisches Stabilitäsprogramm
FOTA ..... Firmware Over-the-Air
FTDMA ..... Flexible Time Division Multiple Access
FTM ..... fehlertoleranter Mittelwert-Algorithmus
GSM ..... Global System for Mobile Communications
ICM ..... Integrated Chassis Management
ID ..... Identifier
IDS ..... Intrusion Detection System
IG ..... Interaktiven Generator

ISP ..... In-System Programming
JTAG ..... Joint Test Action Group
K-CAN ..... Komfort-CAN
Kfz ..... Kraftfahrzeug
LIN ..... Local Interconect Network
LM ..... Light Machine
MCU ..... Micro Controller Unit
MISO ..... Master IN, Slave OUT
MOSI ..... Master OUT, Slave IN
MOST ..... Media Oriented Systems Transport
NIT ..... Network Idle Time
OBD ..... On-Board-Diagnose
OMNeT++ ..... Objective Modular Network Testbad in C++
PDU ..... Protocol Data Unit
PT-CAN ..... Powertrain-CAN
RAM ..... Random-Access Memory
SCK ..... Serial Clock
SPH ..... Stack Pointer Register High Byte
SPL ..... Stack Pointer Register Low Byte
TCK ..... Test Clock
TCU ..... Telematic Control Unit
TDI ..... Test Data In
TDMA ..... Time Devision Multiple Access
TDO ..... Test Data Out
TMS ..... Test Mode Select
UDS ..... Unified Diagnostic Services
VAG ..... Volkswagen-Audi-Gemeinschaft
VCDS ..... VAG-COM Diagnose-System
VIN ..... Vehicle Identification Number
WLAN ..... Wireless Local Area Network
WUS ..... Wakeup Symbol
ZGM ..... zentrales Gateway-Modul
ZGW ..... zentrales Gateway

V. Danksagung

Ich bedanke mich bei meinem Betreuer Herrn Professor Dr. Christian Hummert, für die Unterstützung der Bachelorarbeit, sowie der Bereitstellung zusätzlicher relevanter Module für das Praktikum. Zu dem möchte ich dem Application Center Microcotroller (ACMC) dafür danken, dass ich dort mein Praktikum für die Bachelorarbeit machen durfte und all sein Equipment mit nutzen konnte, welches für meine Versuche nötig war. Speziell möchte ich meinen Zweitprüfer Herrn Dipl.-Ing. (FH) Heiko Polster sowie Herrn Christian Georgi, M.Sc und Herrn Michael Ziemba, M.Sc für die Einarbeitung und Unterstützung, bei den vielen verschieden Tests danken.

1 Einleitung

1.1 Motivation

In Kraftfahrzeugen nimmt der Fortschritt an der Technologie, die den Fahrer unterstützt, immer weiter zu. Systeme wie das Antiblockiersystem (ABS) und das Elektronische Stabilitätsprogramm (ESP), die früher neu und befremdlich wirkten, tragen heute maßgeblich zur Sicherheit bei und sind inzwischen fester Bestandteil im Automobil. Aktuell erhalten auch immer mehr elektronische Assistenten, wie automatische Einparkhilfen, Einzug in Neuwagen. Diese sollen nicht nur für mehr Komfort sorgen, sondern auch ausschlaggebend zur Sicherheit der Insassen beitragen. Solche Assistenzsysteme, die zur Gefahrenabsicherung dienen, warnen zum Beispiel vor Fußgänger die plötzlich die Straße überqueren oder man dabei ist die Spur zu verlassen. [Buschmann, 2012]

Das bisher gängige Controller Area Network (CAN) Bussystem hat sich inzwischen bei vielen Studien als unsicher und angreifbar erwiesen [Mukherjee et al., 2016]. Durch den Zuwachs an Elektronik benötigt es jedoch ein qualifizierteres Bussystem, das die Übertragung der Daten sicher gewährleistet.

Einige Fahrzeug-Assistenten wie Fußgängerwarnsysteme haben Zugriff auf die Bremsen, deswegen ist es von lebenswichtiger Bedeutung, dass solche Bussysteme echtzeitkritisch zuverlässig und ausfallsicher bei der Nachrichtenübertragung sind. Für solche Fälle wurde das Bussystem FlexRay entwickelt, welches besonders bei sicherheitskritischen Elementen wie bei den Bremsen oder der Lenkung, die maßgeblich für den Schutz verantwortlich sind, zum Einsatz kommen soll. [Reif, 2011]

Falls dieses System, welches für das Leben des Insassen oder weiterer Beteiligter verantwortlich sein kann, angreifbar wäre, könnte es immensen Schaden anrichten. Daher stammt auch die Motivation, in Hinblick auf die Forschung in der Car Forensik. Man möchte herausfinden, ob dieses moderne Bussystem Schwachstellen besitzt und falls es diese hat, sie aufzudecken und ergründen. Damit man mit den erlangten Erkenntnissen bereits geschehene Unfälle aufklärt oder präventiv vor Unglücken schützen kann.

1.2 Zielstellung

Der Schwerpunkt dieser Arbeit liegt in der Analyse des modernen Bussystems FlexRay. Dabei soll erläutert werden, warum der Aufbau und die Funktionsweise dieses Kommunikationssystems für die Übertragung sicherheitsspezifischer Nachrichten besser geeignet ist, als das bisher geläufige CAN-Bussystem.

Weiterhin soll mit Hilfe eines zentralen Gateway-Moduls (ZGM), welches für die Übersetzung der Nachrichten verschiedener Bussysteme genutzt wird, diverse Versuche durchgeführt werden. Dabei soll allgemein die Arbeitsweise des Gateways analysiert werden und welche Informationen darauf gespeichert sind, beziehungsweise was für Erkenntnisse man zunächst ohne das Gerät zu öffnen, erzielen kann.

Insbesondere die Bussysteme, die von dem Gateway verarbeitet werden, sollen auf verschiedene Weise analysiert werden. Dabei ist insbesondere der Austausch von CAN- und FlexRay-Signalen über das Gateway, wie es in Abbildung 1.1 schematisch dargestellt ist, für die unterschiedlichen Versuche wichtig.

Ziel der gewonnen Informationen soll darin liegen, Schwachstellen oder Sicherheitslücken im FlexRay-System herauszufinden, beziehungsweise zu ermitteln inwiefern der Bus geschützt ist. Da es gravierende und lebensbedrohliche Folgen mit sich brächte, Schadcode in die Kommunikation einfließen zu lassen, sowie es bei CAN bereits nachgewiesen ist. [Mukherjee et al., 2016]

Um an diese Informationen zu gelangen, sollen unterschiedliche Methoden verwendet werden, unter anderem durch hardware- und auf softwarebasierte Tests, am zentralen Gateway. Zusätzlich besteht die Möglichkeit über die Auswertung der Daten verschiedener Chips auf der Platine im Gerät an weitere Details zu gelangen.

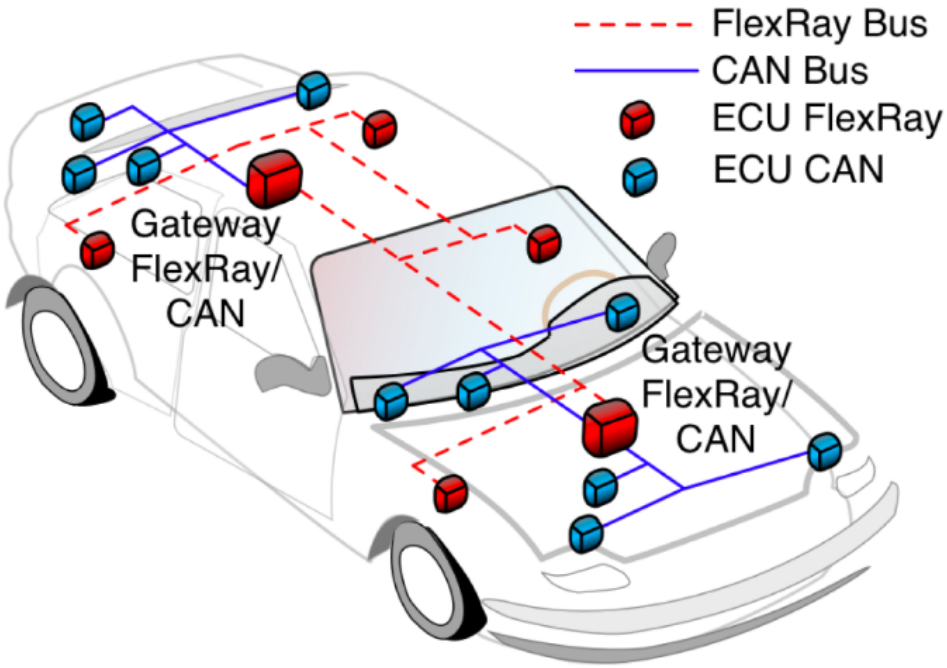


Abbildung 1.1: Darstellung der Vernetzung von CAN und FlexRay über Gateways im Auto [Navet und Simonet-Lion, 2009]

2 Grundlagen

2.1 Eingesetzte Bussysteme im Fahrzeug

Damit die funktionellen Anforderungen verschiedener Fahrzeugkomponenten erfüllt werden können, benötigt es Bussysteme. Diese sind dafür da, um einen Kommunikationsaustausch zwischen den verschiedenen Steuergeräten zu ermöglichen und den Verkabelungsaufwand möglichst gering zu halten. Es existiert dafür ein System, an dem die verschiedenen Teilnehmer (Steuergeräte) über einen Verbindungskanal miteinander verknüpft sind. [Reif, 2012]

Dabei besitzen die Bussysteme prinzipiell drei unterschiedliche Bustopologien. Zum einen gibt es die Sterntopologie, wie in Abbildung 2.1 a). Dieses besteht aus einem zentralen Steuergerät in der Mitte (gekennzeichnet mit "A"), der auch den Namen Sternkoppler trägt. Jeder weitere Teilnehmer am System ist mit diesem Sternkoppler, über eine Punkt-zu-Punkt-Verknüpfung verbunden. Aufgrund dieser Verbindung benötigt die Struktur zwar einen hohen Kabelaufwand, dafür hat sie aber den Vorteil eine hohe Übertragungsrate realisieren zu können, weil sich die Busteilnehmer die Datenleitung nicht teilen müssen. [Reif, 2012]

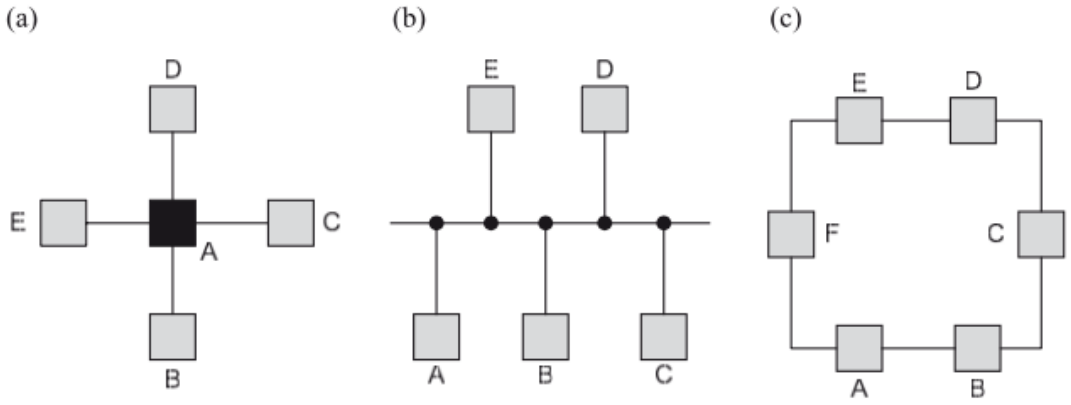


Abbildung 2.1: Mögliche Topologieformen eines Bussystems a) Sterntopologie b) Linientopologie c) Ringtopologie [Reif, 2012]

Eine weitere Form ist die Linientopologie, die in Abbildung 2.1 b) zusehen ist. Diese hat die Eigenschaft, dass alle Teilnehmer des Systems über kurze Stichleitungen an einem Kanal hängen und auch alle gleichermaßen Zugriff auf den Bus haben. Aufgrund dieser Form des Buszugriffs muss es Verfahren zur Kollisionsereknnung- und -vermeidung geben. Der Vorteil ist hierbei, dass ein Ausfall eines Busteilnehmers, am wenigsten das Gesamtsystem beeinflusst. [Reif, 2012]
Die dritte Topologie stellt eine Ringstruktur dar, wie in Abbildung 2.1 c). Dabei sind die Teilnehmer über eine Punkt-zu-Punkt-Verknüpfung, in einem geschlossenen Ring,

miteinander verbunden. Das hat zur Folge, dass das ganze System bei Ausfall eines Steuergerätes gestört werden würde und hier auf eine gute Systemrobustheit geachtet werden muss. [Reif, 2012]

Neben der Topologie unterscheiden sich die Bussysteme unter anderem in den Zugriffsverfahren und in den spezifischen Anforderungen im Fahrzeug.

In dieser Arbeit steht das Bussystem FlexRay im Vordergrund, zuvor folgt jedoch zum besseren Verständnis ein Überblick über die bisher gängigen Bussysteme und deren Aufgabenfelder. Hauptquellen für die Kapitel 2.1.1 Controller Area Network (CAN), 2.1.2 Local Interconect Network (LIN) und 2.1.3 Media Oriented Systems Transport (MOST) sind „Bussysteme in der Fahrzeugtechnik - Protokolle, Standards und Softwarearchitektur“ von Werner Zimmermann und Ralf Schmidgall [Zimmermann und Schmidgall, 2014], sowie „Bosch Autoelektrik und Autoelektronik - Bordnetze, Sensoren und elektronische Systeme“ von Konrad Reif [Reif, 2011] gewesen.

2.1.1 Controller Area Network (CAN)

Das Controller Area Network (CAN) ist ein Feldbussystem, dass die Vernetzung der Steuergeräte im Kaftfahrzeug (Kfz) darstellt und dadurch den Austausch von Daten ermöglicht, ohne intensiven Verkabelungsaufwand aufzuwenden. Gewährleistet wird diese Technik durch den bidirektionalen, bitstrombasierten Zwei-Draht-Linienbus. Diese Kommunikationstechnologie ist bereits seit 1993 im ISO Format 11898 [International Organisation for Standardization, 2015] genormt.

Den CAN-Bus gibt es in drei verschiedenen Klassen, den High-Speed-CAN (Class C), den Low-Speed-CAN (Class B) sowie den Single-Wire-CAN (Class A). Wie der Name vermuten lässt, unterscheiden sich CAN-Low und CAN-High überwiegend durch die Datenrate. Der langsamere erlaubt eine Datenrate von bis zu 125 kbit/s und findet meist im Komfortbereich des Fahrzeugs seine Anwendung. Der High-Speed-CAN ist mit seiner Datenrate von bis zu 1 Mbit/s für kritischere Anwendungen, wie Antrieb-und Fahrwerkbereich verantwortlich, die eine schnellere Verarbeitung benötigen.

Im Vergleich dazu, besteht der Single-Wire-CAN nur aus einer ein-Draht-Ausführung. Die darüber geleiteten Diagnoseanfragen werden nur, bei auftretenden Problemen von Werkstätten abgeschickt. Deswegen wird diese Klasse nur mäßig und in Sonderfällen beansprucht. Inzwischen übernimmt teilweise der High-Speed-CAN schon ihre Funktion.

Für diese unterschiedlichen CAN-Formen findet man in der Literatur verschiedene Bezeichnungen. Die Tabelle 2.1 soll eine Übersicht über die Varianten geben, die im späteren Methodenteil von Bedeutung sind.

CAN-Klassen	Geschwindigkeiten	Praxis-Bezeichnung	Anwendungsgebiete
Class A	Single-Wire-CAN (ca.33-83 kBit/s)	Diagnose-CAN (D-CAN)	Diagnoseabfragen
Class B	Low-Speed-CAN (5...125 kBit/s)	Komfort-CAN (K-CAN)	Karosserielektronik
Class C	High-Speed-CAN (125...1 MBit/s)	Powertrain-CAN (PT-CAN)	Antriebsstrangbereich

Tabelle 2.1: CAN-Varianten [Reif, 2011],[Zimmermann und Schmidgall, 2014],[NewTIS.info, 2018a]

Ein CAN-Netzwerk besteht aus einer Menge von maximal 30 CAN-Knoten, die über ein Übertragungsmedium, dem CAN-Bus, verbunden sind. Dieses kann eine Ausdehnung von bis zu 40 Meter zwischen zwei Busteilnehmern besitzen und ist meist in Form einer Linientopologie ausgeführt.

Der Datenaustausch beim CAN-Bus erfolgt über die CAN-Nachrichten, diese werden jedoch ohne spezifische Angaben von Absender- oder Zieladressen verschickt. Ein wesentliches Merkmal der Nachrichten stellen die CAN-Identifiers (CAN-IDs) dar, die zur Erkennung des Inhalts sowie zur Feststellung der Priorität, einen wesentlichen Einfluss auf das System besitzen. Das Ermitteln der Relevanz der Nachricht kommt zum Einsatz, wenn mehrere Knoten beziehungsweise Steuergeräte, versuchen Botschaften abzuschicken. Für so einen Fall wird festgelegt, dass die Nachricht mit der höheren Priorität zuerst durch geleitet wird. Damit es dabei nicht zu ungewollten Verzögerungen durch Kollisionen kommt, gibt es für solche Zwecke das Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) Verfahren. Dieses überprüft vor dem Senden, ob bereits ein anderer Busteilnehmer zum selben Zeitpunkt sendet. [Hoppe, 2014]

Hinzuzufügen ist, dass die numerische Kennung der CAN-Nachrichten, als 11 Bit Wert ursprünglich 2048 unterschiedliche Nachrichtentypen zugelassen hat. Mit der Erweiterung zum Extended CAN, ist der Wert zu einer 29-Bit-ID geworden, mit dem theoretisch eine Anzahl von 536 870 912 verschiedenen Nachrichtentypen möglich geworden ist. [Hoppe, 2014]

2.1.2 Local Interconnect Network (LIN)

Ende der 90er Jahre wurde ein weiteres Bussystem, namens Local Interconnect Network (LIN) entwickelt. Es sollte insbesondere bei Anwendungen die nur eine niedrige Anforderung, an das System benötigen, eingesetzt werden. Die Namensherkunft folgt daraus, dass sich alle Steuergeräte innerhalb eines begrenzten Bereichs wie zum Beispiel in der Sitz- oder Türelektronik befinden.

Das System ist für maximal 16 Teilnehmer ausgelegt und besitzt lediglich eine Datenrate von 20 kbit/s. Aufgebaut ist das Bussystem über eine eindrähtige Leitung. Es ist meist in linearer Struktur zusammengesetzt (siehe Kapitel 2.1 Abbildung 2.1 b). Aufgrund der begrenzten Übertragungsrate und der beschränkten Topologie wird es eher für Anwendungen verwendet die keine hohe Sicherheitsüberprüfung benötigen, wie für das Schiebedach oder den Wischmotor des Scheibenwischers.
Solche Aufgabenbereiche, wurden zuvor vom Low-Speed-CAN übernommen, wofür nun der LIN-Bus als lokales Subsystem, zur Entlastung des CAN-Busses dient, sowie eine kostengünstigere Alternative darstellen sollte.

Im Gegensatz zu CAN gibt es nur wenige Methoden, um Fehler bei der Übertragung zu erkennen, beziehungsweise gar keine Vorgehensweisen, um den Mangel zu korrigieren. Ebenso ist die Grundidee mit diesem Bussystem Kosten einzusparen, nur bedingt erreicht worden, da CAN-Bauelemente in hoher Stückzahl nur unwesentlich teurer sind.

Zusätzlich schafft der LIN-Bus zwar eine höhere Komplexität im Gesamtnetz, aber sorgt dadurch auch für mehr Potenzial für Fehler. Trotz der Schwachstellen des Local Interconnect Network ist das Bussystem ein fester Bestandteil der Karosserieelektronik geworden.

2.1.3 Media Oriented Systems Transport (MOST)

Des Weiteren existiert noch der Media Oriented Systems Transport (MOST) Bus, der für die Multimedia Anwendung im Auto verwendet wird und daher auch als Infotainment Bus bezeichnet wird. Er verbindet unter anderem Elemente, wie das Navigationssystem mit dem Autoradio. Daher spielt die Übertragungsbandreite eine höhere Relevanz, als die Übertragungssicherheit und Echtzeitfähigkeit des Bussystems. Außerdem muss das System zeitweise mit hohen Datenraten zur Verfügung stehen, durch das Beanspruchen von Detailinformationen, die zum Beispiel das Autoradio wiedergeben soll.

Dabei erfolgt der Datenaustausch über Kunststoff-Lichtwellenleiter, die meist in einer Ringstruktur (siehe Kapitel 2.1, Abbildung 2.1 c) vorliegen. Im Vergleich zu anderen Feldbussystemen in Fahrzeugen hebt sich die Bitrate des MOST-Busses deutlich ab. Der MOST25 macht eine Übertragungsrate bis zu knapp 25 Mbit/s möglich, der MOST50 eine Datenrate von 50 Mbit/s und dementsprechend MOST150 eine Rate von 150 Mbit/s.

Inzwischen ist es außerdem möglich geworden, sich mit dem Mobilfunkgerät in ein Steuergerät des MOST-Busses (wie das Autoradio) zu verbinden und damit Zugang zum Ringsystem zu erhalten. In den Mittel- sowie Oberklasse Wagen geht das bereits auch über Wireless Local Area Network (WLAN). Falls das möglich ist, bietet dieses Bussystem für Kriminelle jedoch einen leichten Zugang in das System. [Lecklider, 2014]

2.2 FlexRay

FlexRay ist das bisher modernste Bussystem in der Automobilindustrie. Aufgrund des spezifischen Aufbaus, der in den folgenden Kapiteln detailliert erläutert wird, hat es einige Vorteile gegenüber anderen Feldbussystemen. Für die Entwicklung dieses Busses hat sich im Jahr 2000 ein FlexRay-Konsortium gegründet, welches sich aus den verschiedenen Unternehmen DaimlerChrysler, Motorola, Philips und BMW zusammensetzt. [Reif, 2011] Serienmäßig eingesetzt wurde FlexRay erstmals, sechs Jahre danach, im BMW X5 für die dynamische Dämpferregelung. [BMW Group, 2009]

Eingesetzt wird dieses System besonders bei echtzeitkritischen Anwendungen, wie zum Beispiel im Fahrwerk-und Antriebsbereich des Kraftfahrzeugs. Speziell im *X*-By-Wire Umfeld, wobei das *X* für Brake oder Steer stehen kann, findet das FlexRay-Bussystem seine Anwendung. Da bei diesen Einsatzgebieten mechanische Verbindungen komplett durch elektronische Signale ersetzt werden ist eine schnelle Verarbeitung und hohe Fehlertoleranz notwendig. [Reif, 2011],[BMW Group, 2009]

Ein Ausfall dieser Funktion würde damit auch ein hohes Sicherheitsrisiko darstellen. Zum Beispiel wäre der Fahrer, bei dem das Steer-by-Wire System ausfallen würde, machtlos über die Lenkung seines Fahrzeugs und schwere Unfälle wären nicht auszuschließen. [Reif, 2011]

2.2.1 Vorteile zur Nutzung von FlexRay im Bezug zum CAN-Bussytem

Das Bussystem FlexRay ist speziell in Hinblick auf verbesserungsfähigere Merkmale des CAN-Busses konzipiert worden. Gerade durch die Linienbustopologie und der maximal 1 Mbit/s schnellen Leitungen gewährleistet CAN zwar eine hohe Übertragungssicherheit, jedoch Nachteile in Bezug zur Anforderung der Absicherung. Um die Ansprüche eines sicheren Systems zu erfüllen, benötigt es zum Beispiel die Eigenschaft, eine höhere Fehlertoleranz aufzuweisen, damit die Funktionalität der beanspruchten, sicherheitsrelevanten Komponenten auch erhalten bleibt. [Zimmermann und Schmidgall, 2014]

Des Weiteren sollte das Bussystem eine deterministische Datenkommunikation aufweisen. Im Vergleich dazu ist bei CAN der Datenaustausch ereignisorientiert. Das heißt zwar, dass hochpriorisierte Nachrichten sicher übertragen werden, dafür aber niedrig priorisierte Nachrichten bei einem sehr hohen Botschaftenaufkommen, eventuell erst zu spät gesendet werden oder in der Masse untergehen könnten. [Zimmermann und Schmidgall, 2014]

Diese Eigenschaft des ereignisorientierten Nachrichtenaustausches stellt zudem eine Schwachstelle des Controller Area Networks dar. Beispielsweise kann somit das Bussystem, in Form eines Denial-of-Service zum Abstürzen gebracht werden. Bei so einer

Angriffsform schickt man eine sehr hohe Anzahl von Nachrichten, immer mit der höchsten Priorität an den CAN-Bus. Das System, welches mit dem erhöhten Aufkommen, mit sehr wichtigen Nachrichten überfordert ist, könnte damit zum Absturz gebracht werden. [Mukherjee et al., 2016] Inwiefern FlexRay diese Sicherheitslücke mit seinem Kommunikationsschema schließt, ist im Kapitel 2.2.3 Buszugriff, nachzulesen.

Außerdem ist der einkanalige Aufbau des Systems, ein weiter Nachteil des CAN-Busses. Das System würde beim Versagen der Busverbindung zusammenbrechen. Um den entgegenzuwirken, kann FlexRay einen ein- oder zweikanaligen Aufbau nutzen. Bereits ein einziger Kanal kann eine Datenrate von 10 Mbit/s erreichen. Ein zusätzlicher physisch getrennter Zweiter kann bei einem sicherheitskritischen Nachrichtenaustausch, als redundanter Kanal dienen oder die Datenrate auf das Doppelte erhöhen. Zusätzlich ist es durch diesen Variante möglich, den maximalen Abstand von 24 Metern zwischen zwei Busteilnehmern, zu erweitern. [Reif, 2011]

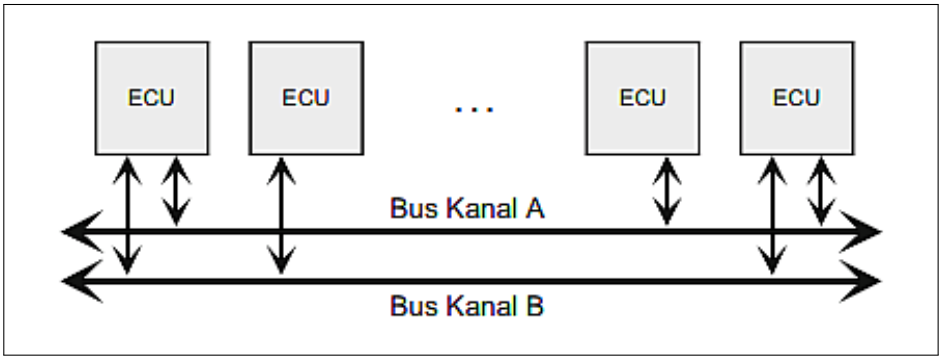


Abbildung 2.2: Struktur eines FlexRay-Busses mit zwei Kanälen [Zimmermann und Schmidgall, 2014]

Dabei ist bei einem zweikanaligen Aufbau zu beachten, dass die Steuergeräte sowohl an Kanal A als auch an Kanal B gekoppelt sein können. Die Kommunikation der Steuergeräte (ECUs) [1] untereinander ist aber nur möglich, wenn beide an demselben Kanal hängen und nicht wenn eine Komponente am Bus Kanal A hängt und der andere am Kanal B (siehe Abbildung 2.2). [Zimmermann und Schmidgall, 2014]

2.2.2 Kommunikation

Topologien

Eine Besonderheit des Bussystems FlexRay ist die Nutzung unterschiedlicher Topologien für den Kommunikationsaustausch. Es kann wie bereits die vorgestellten CAN-und LIN-Busse, als Mehrpunkttopologie (Linie) verwendet werden oder in Form einer Sterntopologie. Der maximale Abstand von 24 Meter trifft sowohl bei der Sternstruktur als auch bei der Linie zu. [Reif, 2011]

[1] Electronic Control Unit (ECU)

Des Weiteren ist die Sterntopologie in aktive und passive Sterntopologie zu unterscheiden. Eingesetzt wird die passive Form vor allem bei niedrigen Bitraten und kurzen Verbindungen. Daher ist es sinnvoll, einen aktiven Sternkoppler mit dem Sternpunkt zu verbinden. Dieser kann aktiv die Signale verstärken oder aufbereiten, sowie die Weiterleitung der Nachrichten vornehmen. Zusätzlich kann er durch die Segmentierung des Netzwerkes beim Bemerken eines Fehlers, den fehlerhaften Zweig vom Kopplungspunkt trennen und eine höhere Ausfallsicherheit gewährleisten. [Zimmermann und Schmidgall, 2014]

Außerdem ist eine hybride Topologie machbar, bei der die Linien-und die Sternstruktur miteinander kombiniert werden. Der Vorteil dieser Kombination kann einerseits eine Einsparung von Kosten sein und andernfalls eine höhere Ausfallsicherheit für das gesamte Netzwerk bewirken. [Reif, 2011]

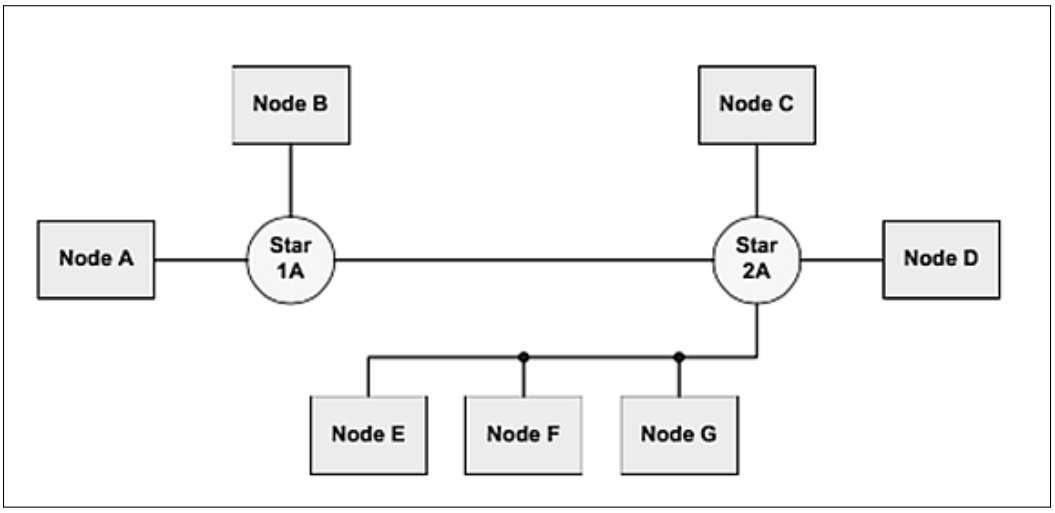


Abbildung 2.3: Hybride Struktur aus Stern-und Bustopologie [FlexRay Consortium, 2010]

Ein Beispiel hierfür zeigt die Abbildung 2.3. Dort wurde zum einen an dem Sternkoppler 2A eine Linientopologie angefügt und zum anderen eine weitere Sterntopologie hinzugefügt. [FlexRay Consortium, 2010] Zusätzlich stellt eine Verknüpfung von mehreren (maximal drei) Sterntopologien eine kaskadierte Sterntopologie dar. [Reif, 2011]

FlexRay-Knoten

Um die Strukturen erfüllen zu können, besitzt das Bussystem unterschiedliche Hardware-Komponenten, die den Kommunikationsaustausch verwirklichen. Eine Hauptkomponente sind die FlexRay-Knoten, die aus verschiedenen Bestandteilen bestehen. Der Knoten besteht wie in Abbildung 2.4 zu sehen, aus einem Host-Prozessor, einem als Transceiver dienenden Bus Driver (BD) pro Kanal, sowie einen Communication Controller (CC), der vom Host Prozessor die Informationen bekommt. Diese Daten wertet der CC gemäß dem Kommunikationsprotokoll aus der FlexRay-Spezifikation aus [FlexRay Consortium, 2010].

Außerdem realisiert er Aufgaben wie das Framing, Aufwecken und Schlafenlegen des FlexRay-Busses, die Fehlererkennung und -behandlung und die Synchronisation. [Reif, 2011]

Der Bus Driver schafft die Verbindung zwischen dem Übertragungsmedium und dem FlexRay-Controller. Dabei wandelt es die empfangenen, logischen Informationen in einen physikalischen Signalstrom um und umgekehrt. Eine weitere Eigenschaft des BD ist es, physikalische Fehler durch Überwachung der Busleitungen auf dem Bus zu erkennen. [Reif, 2011]

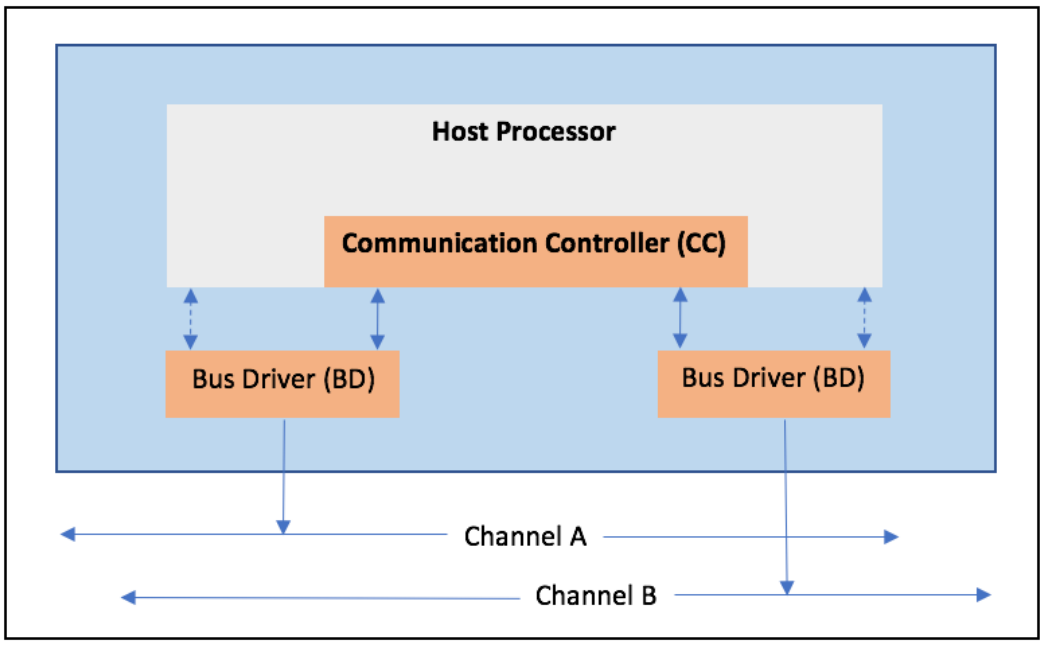


Abbildung 2.4: Aufbau eines FlexRay-Knotens [Eigene Abbildung]

Neben der Überwachung durch einen Bus Drivers, kann außerdem optional ein Bus Guardian (BG) im Knoten eingesetzt werden. Prinzipiell schützt zwar der zweikanalige Aufbau bei Ausfall eines Kanals, jedoch tritt teilweise auch ein Verhalten namens “Babbling Idiot” auf. In so einem Fall ist ein störendes Steuergerät an beiden Kanälen angeschlossen. Um auch vor solchen Szenarien mehr Schutz zu bieten, kann der Bus Guardian eingebaut werden.

Dieser hat die Übersicht über den zeitlichen Ablauf eines Kommunikationsschemas und kann damit entscheiden, ob ein Netzknoten zu einem gewissen Zeitpunkt zum Senden berechtigt ist. Damit kann der BG Übertragungen des problematischen Knotens zwar nicht direkt abschalten, aber verhindern, dass die Kommunikation der anderen Netzknoten gestört wird. [Zimmermann und Schmidgall, 2014]

Eigenschaften des Busses

Das Übertragungsmedium in einem FlexRay-System ist eine Twisted-Pair Verkabelung, die sowohl geschirmt als auch ungeschirmt, zum Einsatz kommen kann. Die Besonderheit ist, dass die Kanäle jeweils aus zwei Adern bestehen, die auch als Bus-Minus (BM) und Bus-Plus (BP) benannt werden. [Reif, 2011] Diese Leitungen übertragen die Bits in Form von Differenzspannungssignalen. Wenn die daraus resultierende Differenzspannung ungleich Null Volt ist, wird dieser Bereich auch als dominant bezeichnet. Im Vergleich zu einem Ruhezustand befinden sich BM und BP auf demselben Niveau von etwa 2,5 Volt (siehe Abbildung 2.5). Zusätzlich werden bei Topologiestrukturen wie Sterntopologie mit aktivem Sternkoppler und Linientopologie an den Leitungsenden zur Vermeidung von Reflexionen festgelegte Abschlusswiderstände benötigt. [Zimmermann und Schmidgall, 2014]

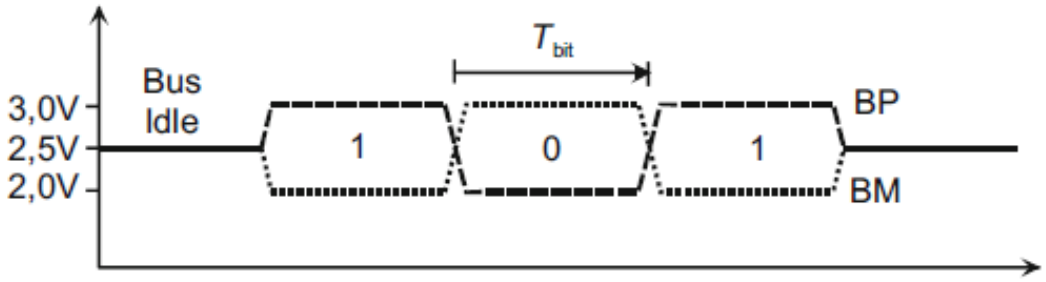


Abbildung 2.5: Signalpegel der FlexRay-Leitungen Bus-Plus (BP) und Bus-Minus (BM) [Zimmermann und Schmidgall, 2014]

2.2.3 Buszugriff

Im Gegensatz zu CAN ist bei FlexRay das Verschicken von Nachrichten nach einem strengen Kommunikationszyklus geregelt. Dieser Ablauf ist über das Time Devision Multiple Access (TDMA) beziehungsweise dem Flexible Time Division Multiple Access (FTDMA) Verfahren strukturiert. Dabei erfolgt eine Einteilung im Zyklus in vier verschiedene Abschnitte, die aus einer definierten Menge von **Zeitslots** bestehen, die wiederum in **Makroticks** untergliedert sind. Die Makroticks sind zusätzlich unterteilt in **Mikroticks**, was der kleinsten Zeiteinheit entspricht, wie es in der Abbildung 2.6 dargestellt ist.

Zwei Teile im Zyklus sind Kommunikationssegmente, wovon eines das statische und das andere ein dynamisches Segment darstellt. Der statische Abschnitt, in den periodischen Nachrichten verschickt werden, muss immer in einer Abfolge vorhanden sein. Das dynamische Segment hingegen ist nur optional vorhanden, weil es nur für ereignisorientierte Nachrichten eingesetzt wird. [Zimmermann und Schmidgall, 2014]

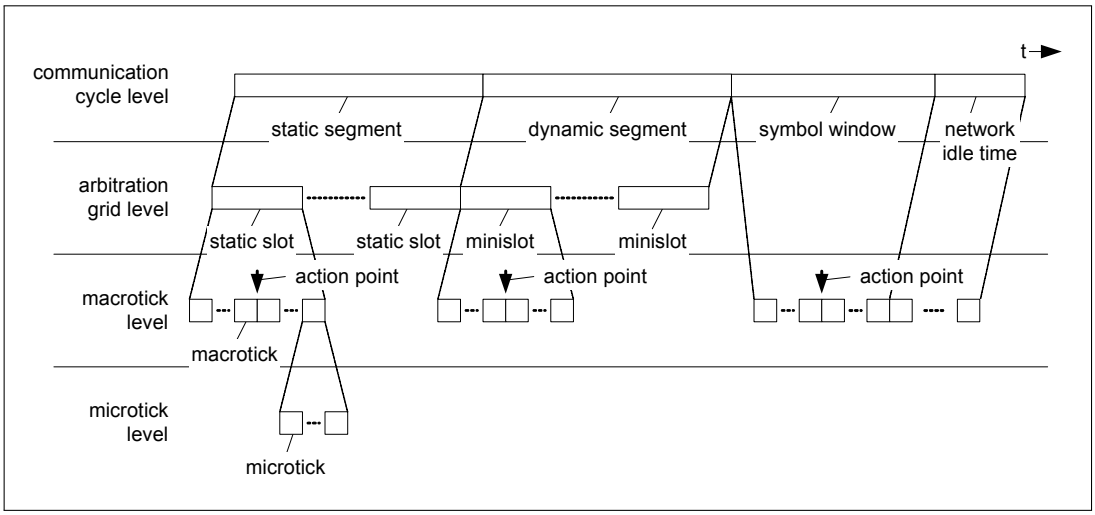


Abbildung 2.6: Kommunikationszyklus [FlexRay Consortium, 2010]

Danach kann optional ein Symbol Window als kurzes Zeitfenster vorhanden sein und am Ende folgt die Network Idle Time (NIT), die verpflichtend in einem Zyklus vorhanden sein muss, damit die Synchronisierung der Busteilnehmer stattfinden kann. Hierzu findet in diesem Zeitraum keine Datenkommunikation statt. [Zimmermann und Schmidgall, 2014]

Der statische Abschnitt ist in eine definierte Anzahl von gleich großen Zeitrastern (static slots) eingeteilt. Der Aufbau von diesen Slots ist speziell festgelegt, sodass genau eine FlexRay-Botschaft darin übermittelt werden kann. Demzufolge wird das Frame mit dem Identifier (ID) eins auch im Slot eins übertragen und nach dem Slot weiter hochgezählt. Dadurch kann für jedes Steuergerät das Recht zu senden, exakt zugeteilt werden und Kollisionen über das TDMA-Verfahren vermieden werden. [Reif, 2011] Zusätzlich ist eine Besonderheit am Aufbau, dass die Nachrichtenübertragung durch das Action Point Offset eingeleitet wird, um eventuellen Signalverzögerungen entgegen zu wirken. [Zimmermann und Schmidgall, 2014]

Auf Basis des Flexible Time Division Multiple Access folgt der Buszugriff im dynamischen Segment. Es gibt dabei ebenfalls definierte Zeitschlitze, die in diesem Segment als Minislots bezeichnet werden. Sie sind alle gleich groß, aber wesentlich kleiner als die Slots im statischen Bereich. Das Frame, das von einem Steuergerät Übertragen werden soll, kann hierbei unterschiedlich lang sein, solange es nicht die Größe des dynamischen Abschnitts überschreitet. Nach dem Übertragen der Nachricht, zählt der Slot-Zähler wieder eine Zahl hoch und für den folgenden Minislot, hätte das nächste Steuergerät das Senderecht. Falls es keine Informationen zum Versenden hat, verzichtet es auf das Recht zu senden und der Slot Counter inkrementiert den Wert weiter. Durch den flexiblen Verlauf des Sendens von unterschiedlich langen Nachrichten kann die Zahl im Slot Counter der verschiedenen Kanäle voneinander abweichen und asynchron verlaufen. [Zimmermann und Schmidgall, 2014]

Im Vergleich zum statischen Abschnitt zeigt der Slot Counter im dynamischen Segment nicht nur, an welches Gerät gerade die Berechtigung zum Senden hat, sondern auch indirekt die Wichtigkeit der aktuellen Nachricht. [Zimmermann und Schmidgall, 2014]

2.2.4 Aufbau von FlexRay-Botschaften

Ein FlexRay-Frame, auch Botschaft oder Nachricht genannt, besteht grundlegend aus drei verschiedenen Teilen, dem Header, der Payload und dem Trailer-Segment (siehe Abbildung 2.7). [Zimmermann und Schmidgall, 2014]

Das Header Segment beginnt mit fünf Indikator- beziehungsweise Steuerbits. An erster Stelle ist das Reserved Bit, dass für spätere Änderungen in der Spezifikation freigehalten, wird. Die weiteren vier Bits sind für zusätzliche Sonderinformationen des Frames vorhanden.

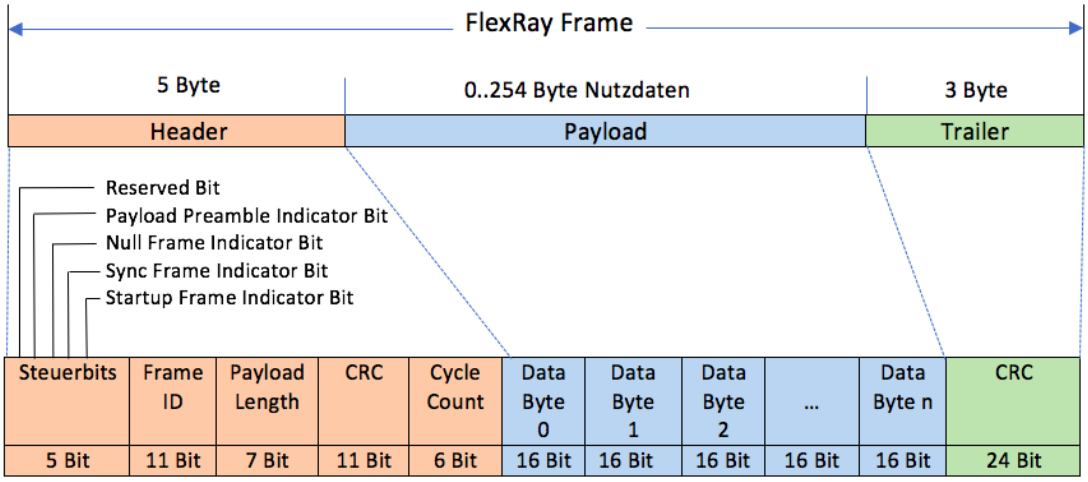


Abbildung 2.7: Aufbau eines FlexRay-Frames [Eigene Abbildung]

Das Payload Preamble Bit enthält weitere Informationen für den statischen, beziehungsweise den dynamischen Part, welche zur Filterung der Nachricht genutzt werden können. Der Null Frame Indicator sagt aus, ob der Inhalt nutzbar oder ungültig ist. Sind nur Nullen im Frame, ist der Inhalt nicht gültig.
Als Nächstes folgt das Sync Frame Indicator Bit, welches zur Zeitsynchronisation für die Knoten gesendet wird und das letzte Bit ist das Startup Frame Indicator Bit, was für den Netzwerkstart benötigt wird, weitere Informationen folgen dazu im Abschnitt 2.2.5 Synchronisationsprinzip. [Zimmermann und Schmidgall, 2014]

Nach den Steuerbits folgt der spezifische Identifier der Nachricht (Frame ID), diese Zahl entspricht gleichzeitig dem belegten Slot, in dem die Botschaft gesendet wird. Darauf folgt das Feld Payload Length für die Größenangabe der Daten, die im Payload Segment

übertragen werden. Diese Daten, die dafür übertragen werden, sind je zwei Word[2] groß. Insgesamt kann das Feld eine Größe von 254 Byte besitzen. Dabei ist die Angabe für den statischen Abschnitt immer dieselbe und variiert nur für das dynamische Segment. [Reif, 2011]

An letzter Stelle des Kopfteils der Nachricht ist der Cylcle Count, der die Anzahl des momentanen Kommunikationszyklus angibt, der am Anfang mit Null beginnt und pro Zyklus inkrementiert wird. Das Feld mit dem Namen CRC steht für Cyclic-Redundancy-Check und ist eine Prüfsumme, die aus bestimmten Generatorpolynomen geildet wird. Es soll zum einen den Header (ohne Cycle Count) als auch das ganze Frame vor Fehlern bei der Übertragung schützen soll. Dafür ist einerseits der elf Bit große Abschnitt im Header vorgesehen und andererseits das ganze Trailer Segment mit einer Länge von 24 Bit. [Zimmermann und Schmidgall, 2014]

2.2.5 Netzwerkstart und Synchronisationsprinzip

Netzwerkstart

Für den Start des Netzwerks benötigt es spezifische Kaltstart-Knoten (Coldstart Nodes), die durch mindestens zwei, besser drei Steuergeräte verkörpert werden. Auslöser zum Starten kann entweder das Zuschalten einer Spannungsversorgung sein oder durch einen Knoten, der bemerkt, dass der Bus im Ruhezustand ist und daraufhin ein Wakeup Symbol (WUS) oder Wakeup Pattern sendet. [Zimmermann und Schmidgall, 2014]

Daraufhin beginnen die Coldstart Nodes auf den vorhandenen Übertragungsleitungen Collision Avoidance Symbole (CAS) zu versenden. Sowohl das CAS als auch das WUS sind von den Bitmustern so individuell unterscheidbar, dass die anderen Steuergeräte daran erkennen, dass der Kommunikationszyklus beginnt. Fortgeführt wird der Start, in dem der Kaltstart-Knoten im Kommunikationszyklus Null mit dem übertragen der regulären Botschaften beginnt. Besonderheit in den Frames ist, dass in dem Header die Indicatoren Sync Node und Startup gesetzt sind. Die weiteren Kaltstart-Knoten starten, wenn diese vom führenden Knoten für den Kaltstart mindestens vier Nachrichten mit den gesetzten Steuerbits erhalten haben. Damit weitere Steuergeräte im System starten können, benötigen diese wiederum von mindestens zwei anderen Coldstart Nodes, vier dieser Start Nachrichten. Im Idealfall wäre der Start des Netzes nach acht vollständigen Zyklen abgeschlossen. [Zimmermann und Schmidgall, 2014]

[2] Dateneinheit

Synchronisationsprinzip

Damit der Kommunikationszyklus erfolgreich ablaufen kann, ist in einem Sendefenster eine spezifische Kennzeichnung freigegeben, die den Knoten eine Erlaubnis zum Senden erteilt. Um diese Voraussetzung zu gewährleisten, benötigen alle Netzknoten dieselbe synchrone Zeitinformation beziehungsweise dasselbe Zeitverständnis. [Reif, 2011]

Für diese Aufgabe werden aus dem statischen Segment Frames gesendet, die als Steuerbit das Sync Node Indicator Bit gesetzt haben. Damit die Nachrichten zum selben Zeitpunkt den Takt vorgeben, ist ein Verfahren zur Synchronisierung aller beteiligten Knoten am Netz notwendig. Für das genannte Prozedere spielen verschiedene Parameter wie die Dauer eines Zyklus, die Anzahl der vorkommenden Mikroticks darin und deren Dauer pro Zeiteinheit eine Bedeutung. [Reif, 2011]

Problematisch ist hierbei, dass der Takt im laufenden Netzwerk, durch äußere Einflüsse nicht konstant oder gleich groß ist. Daher muss es verschiedene Korrekturmaßnamen für die Zeitsynchronisierung geben. Diese Verfahren können entweder die Steigung oder die Phase korrigieren. [Reif, 2011],[Zimmermann und Schmidgall, 2014]

Bei beiden Korrekturen wird ein fehlertoleranter Mittelwert-Algorithmus (FTM) angewendet. Dieser sortiert die Werte der angekommen Sync-Botschaften. Ist die Anzahl der verantwortlichen Synchronisierungs-Nachrichten kleiner als drei, wird der Mittelwert aus denen gebildet. Falls die Anzahl der übertragenen Sync-Frames zwischen drei und sieben beträgt, wird jeweils der größte und kleinste Wert gestrichen werden. Beträgt die Zahl der verantwortlichen Frames mehr als sieben, werden zur Verhinderung der Abweichung, jeweils die zwei kleinsten und zwei größten Werte entfernt. [Reif, 2011]

Steigungs-und Phasenkorrektur

Mithilfe der Steigungskorrektur oder auch Frequenzkorrektur genannt, werden in einem Knoten vorkommende Diskrepanzen, bezüglich der Übertragungsfrequenz während eines Kommunikationsablaufs korrigiert. Dafür werden die benannten Frequenzen aller Netzknoten betrachtet, der FTM angewendet und die eigene Übertragungsfrequenz daran adaptiert. [Reif, 2011],[Zimmermann und Schmidgall, 2014]

Durch die Phasen- beziehungsweise sogenannte Offsetkorrektur werden Unterschiede in der Phase der Kommunikation, durch das Hinzufügen eines Offsets in der Network Idle Time (NIT) verbessert. Dabei vergleicht der Netzknoten die Phasendifferenz der anderen beteiligten Knoten, unter Anwendung des FTM. Mithilfe des Einfügens eines Offsets, wird dadurch berechnet ob die NIT Phase verlängert oder verkürzt werden muss. Da alle Busknoten diese Methode anwenden, ist eine synchrone Übertragung eines Kommunikationszyklus möglich. [Reif, 2011],[Zimmermann und Schmidgall, 2014]

2.3 Zentrales Gateway-Modul

Für die praktische Analyse wurde ein zentrales Gateway-Modul (ZGM) der Marke BMW, mit dem Namen “ZGW-01 High” zur Verfügung gestellt. Dieses Gerät sorgt für die Verbindung der Haupt-Busse im Kraftfahrzeug miteinander, wie bereits erläutert, kann es sich dabei unter anderem um den CAN-, LIN-, MOST- oder FlexRay-Bus handeln (siehe Abbildung 2.8).

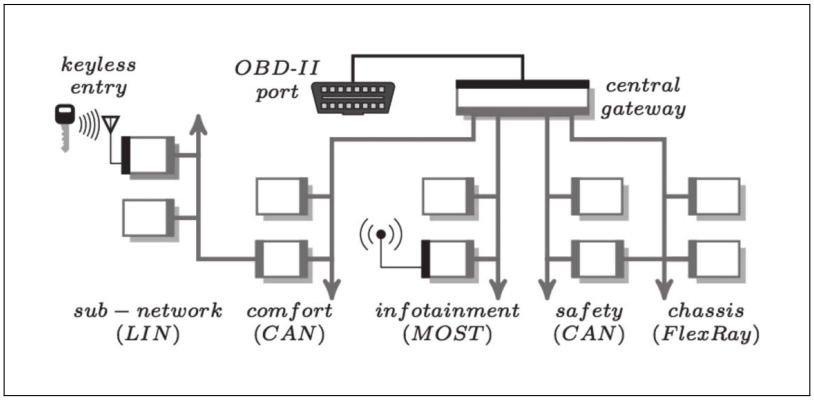


Abbildung 2.8: Platzierung des Gateways [Petit und Shladover, 2015]

Welche Systeme aber miteinander verknüpft werden und damit Daten austauschen können, hängt von der Fahrzeugausstattung sowie dem Herstellungsjahr ab. Relevant ist es zum Beispiel um Diagnoseanfragen über den Diagnose-CAN (D-CAN) an andere Busse, mithilfe des Gateways zu senden, um weitere Informationen bezüglich Problemen zu erhalten. Durch die am Gateway aufgedruckte Identifkationsnummer konnte die Fahrzeugserie ermittelt werden, aus der das Modul stammt. Zu diesem ZGW[3] sind nicht viele Informationen vom Hersteller veröffentlicht, daher wurden die folgenden Daten aus dieser Quelle: [NewTIS.info, 2018a] entnommen. [NewTIS.info, 2018a]

In Abbildung 2.9 ist ein zentrales Gateway-Modul (Index 1), mit seinen Bauteilsteckern zu sehen. Für jeden dieser Stecker gibt es separate Pinbelegungen und einen definierten Aufbau (siehe Tabelle 2.2), die im Methodenteil ausführlich getestet wurden und interessante Ergebnisse lieferten.

Index	Name	Aufbau	Farbe	Funktion
2	A51*2B	2 Steckverbindungen	schwarz	MOST Schnittstelle
3	A51*3B	18 Steckverbindungen	blau	
4	A51*1B	54 Steckverbindungen	schwarz	

Tabelle 2.2: Bauteilstecker des zentralen Gateway-Moduls [NewTIS.info, 2018a]

[3] Die Bezeichnung zentrales Gateway (ZGW) entspricht der Bezeichnung ZGM.

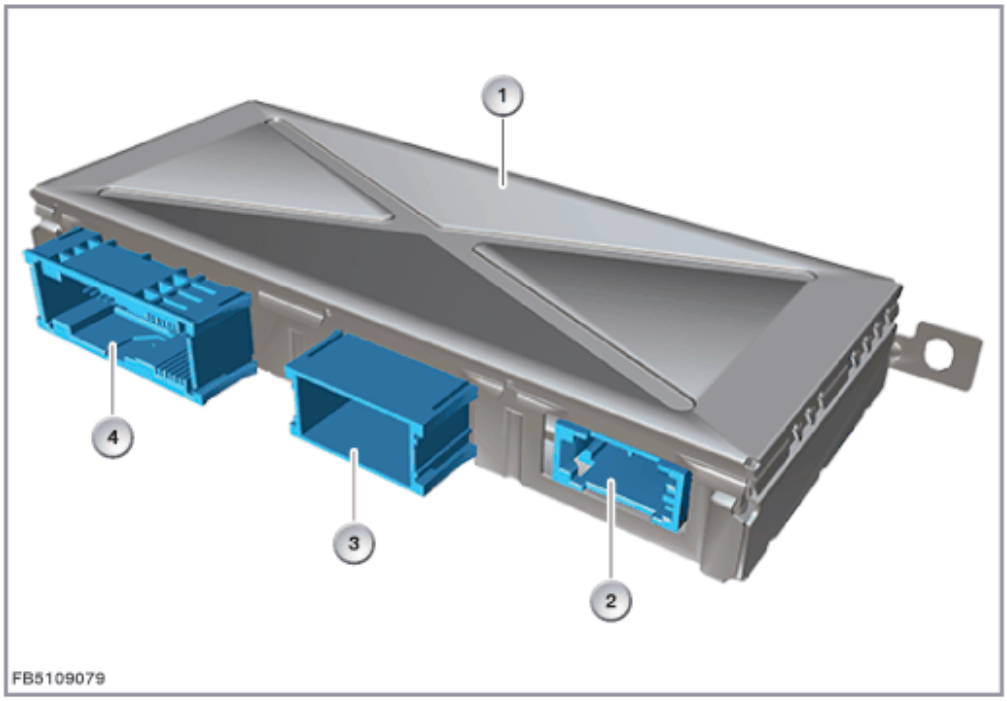


Abbildung 2.9: Zentrales Gateway-Modul [NewTIS.info, 2018a]

Funktionsbeschreibung:

Das Gateway hat vielerlei verschiedene Funktionen, die für das Verständnis der weiteren Analyse im Methodenteil, zuvor in diesem Kapitel erläutert werden. Diese Aufgaben-bereiche betreffen unter anderem die Synchronisierung, das Konfigurationsmanagement, dem Speichern der Fehler sowie das Starten und Runterfahren des Kommunikationsnetzes.

–> Synchronisierung
Für einen reibungslosen Ablauf im System ist notwendig, dass die ECUs dasselbe Zeitverständnis besitzen. Die Synchronisierung muss über den Bus erfolgen. Dafür besitzen zum Beispiel folgende Steuergeräte, die Funktion als Synchronisationsknoten zu wirken:

- Zentrales Gateway-Modul (ZGM)
- Integrated Chassis Management (ICM)
- Dynamic Stability Control (DSC)
- Digitale Motor Elektronik (DME)/Digitale Diesel Elektronik (DDE)

–> Konfigurationsmanagement
Eine Systemfunktion des Moduls ist es Informationen zur Fahrzeugkonfiguration intern abgelegt sind. Diese Funktion erlaubt es wichtige Daten zum Kfz, wie das Fahrzeugprofil oder den Fahrzeugauftrag zentral aufzubewahren. Falls diese Informationen benötigt werden, sind diese über Diagnosebefehle abrufbar.

Für den Fall, dass das Gateway ausgewechselt werden muss, sind diese Daten zusätzlich im Car Access System (CAS) abgesichert.

–> Zentraler Fehlerspeicher
Hervorzuheben ist insbesondere die Aufgabe der zentralen Sicherung von sogenannten Check-Control-Meldungen. Die Nachrichten können zum einen für das ganze Fahrzeug gelten, als auch spezifisch zu den Steuergeräten zugeordnet werden. Aufgrund dieser Funktion wird das zentrale Gateway-Modul auch als Diagnose-Master bezeichnet.

–> Starten und Runterfahren des Kommunikationsnetzes
Das Aktivieren und Herunterfahren des Kommunikationsboardnetzes wird im Zustandsmanagement des Fahrzeugs dokumentiert. Es beinhaltet unter anderem Bedingungen, die alle ECUs dafür erfüllen müssen, sowie definierte Informationen über das gleichzeitige Starten und Abschalten der Busse (Kaskadierung) und den dazugehörigen Weck- und Einschlafspeicher.

Die Kaskadierung stellt den Zeitplan sicher, dass alle Steuergeräte die an dem Bordnetz mit verantwortlich sind, geordnet hochfahren und gleichzeitig wieder „einschlafen“.

Damit diese Funktion des Hoch-und Runterfahrens präzise abläuft, gibt es seitens des ZGM die Funktion problematische Aufwach-und Einschlafprozesse zu ermitteln und den Fehlern entgegenzuwirken. Für diesen Aufgabenbereich ist insbesondere der Weck-und Einschlafspeicher im Fahrzeug Zustandsmanagement verantwortlich. Die Folge eines fehlerhaften Ablaufs würde zu einem erhöhten Stromverbrauch sorgen, was wiederum die Batterie entleeren und zum Stillstand des Kfzs führen könnte.

Bei einem Aufwachvorgang ermittelt das Zustandsmanagement zunächst alle Ursachen, ob das Steuergerät dazu befähigt, ist das Fahrzeug zu wecken. Diese Entscheidung trifft es über festgelegte Identifikationsnummern, mit denen die Steuergeräte versehen sind. Trifft ein Grund zum Wecken zu, wird das dem Einschlaf-und Weckspeicher übermittelt, der im ZGM vorhanden ist. Handelt es sich dabei um einen fehlerhaften Grund das Automobil zu wecken, dokumentiert es das Gateway-Modul inklusive Steuergerät, von dem der Befehl kam mit Weckursache, aktueller Kilometerstand und Uhrzeit.
Eine Gegenmaßnahme für den problematischen Weckbefehl wäre das Senden eines Power-Down Befehls. Falls weiterhin fehlerhafte Wecknachrichten gesendet werden, wird ein Zurücksetzen beziehungsweise Abschalten der Versorgungsklemme - namens 30F, gefordert. Ebenso können fehlerhafte Einschlafvorgänge über den Weck-und Einschlafspeicher detektiert und Gegenmaßnahmen getroffen werden.

2.4 Software-Tool CANoe

Um an ein besseres Vorstellungsvermögen über den Aufbau der Bussysteme von CAN und FlexRay zu gelangen, wurde mit dem Software-Tool CANoe 10.0 SP2 von Vector Informatik GmbH gearbeitet [Vector Informatik GmbH, 2018]. Dieses Tool ermöglicht es einzelne Steuergeräte, sowie ganze Netzwerke mit all seinen detaillierten Komponenten als Software zu generieren. Dafür gibt es eine Darstellung für den Simulationsaufbau, bei der zunächst das gewünschte Bussystem, welches generiert werden soll, auswählt werden kann. Zur Auswahl stellt CANoe bereits eine große Bandbreite an Beispielkonfigurationen zur Verfügung, aus denen ganze Netzwerke oder nur einzelne Teile entnommen werden können. [Vector Informatik GmbH, 2018]

In Abbildung 2.10 ist ein entsprechender FlexRay-Aufbau zu sehen.

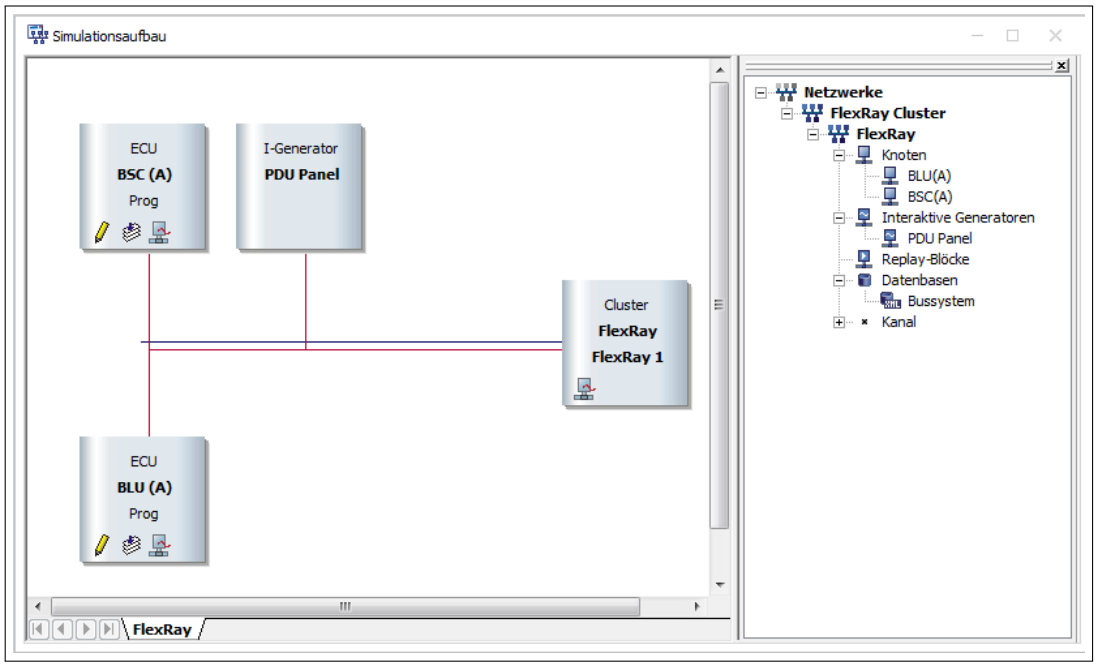


Abbildung 2.10: Selbst erstellter FlexRay-Simulationsaufbau in CANoe [Eigene Abbildung]

Nach diesem Schritt kann man durch Einfügen von verschiedenen Netzknoten die Grundstruktur festlegen. Um diese Knoten miteinander, beziehungsweise mit dem Übertragungskanal zu verbinden, benötigt es die Erstellung einer Datenbank, die von der Wahl des Bussystems abhängig ist. Für CAN gibt es zum Beispiel den „CANdb++ Editor“ und für FlexRay das Programm „AUTOSAR EXPLORER“. Mithilfe dieser Tool-Erweiterung besteht die Möglichkeit, neben den Zusammenhängen auch spezifische Frames für die Steuergeräte zu generieren. Damit ist es zum Beispiel machbar, in einem FlexRay-Netzwerk über die Datenbank die Nachrichten dem definierten Kommunikationssegment, Slot, Signale, Sender und Empfänger etc. zuzuweisen. [Vector Informatik GmbH, 2018]

Nachdem der Aufbau abgeschlossen ist, kann man mithilfe einer CANoe Funktion für das Steuergerät, benutzerdefinierte Systemvariablen anlegen, die systemweit gültig sind. Für eine geeignetere Repräsentation der Oberflächendarstellung ist es möglich, Panels zu designen. Diesen Panels kann man grafisch bearbeiten und Funktionen beziehungsweise Verhaltensmuster übergeben. [Vector Informatik GmbH, 2018]

Eine der wichtigsten Funktionen von CANoe ist es, den Knoten eine gewisse Intelligenz mithilfe von Quellcodes zu vermitteln. Für diese Aufgabe steht der CAPL-Browser zur Verfügung, der dazu übermittelnde Programmcode muss dazu in der gleichnamigen CAPL-Sprache geschrieben werden.
CAPL ist die Programmiersprache, die bei CANoe zur Erstellung von individuellen Anwendungen der Steuergeräte verwendet wird. Sie steht für Communication Acess Programming Language und ähnelt der C-Programmierung. Besonderheit an der Sprache ist, dass sie für alle Fahrzeug Bussysteme angewendet werden kann und eventorientiert auf Sachverhalte einer Funktion im System reagieren kann. Von Vorteil ist, dass die Software für die verschiedenen Bussysteme schon Beispielfunktionen bereitstellt. [Vector Informatik GmbH, 2018]

Ist auch dieser Prozess abgeschlossen und gesichert besteht die Möglichkeit mit den entwickelten Systemen verschiedenen Szenarien simuliert oder hardwarebasiert zu testen und zu analysieren. Um verschieden Rückschlüsse daraus zu ziehen, helfen bei der Auswertung auch die verschiedenen grafischen Oberflächen wie das Trace-oder Grafik-Fenster, welche zur Verfügung gestellt werden. Mit diesen Informationen ist es wiederum möglich, genaue Aussagen zu treffen oder Fehler frühzeitig zu erkennen. [Vector Informatik GmbH, 2018]

3 Methodenteil

Für die Analyse des modernen Bussystems FlexRay sollen unterschiedliche Methoden zum Ziel führen. Zuerst ist eine Einarbeitung in die CANoe-Softwareumgebung erforderlich, um die Grundlagen bezüglich FlexRay zu vertiefen. Dabei ist das Ziel, Ergebnisse einer zu erstellenden FlexRay-Beispielkonfiguration in CANoe zu analysieren, um daraus Erkenntnisse für die praktischen Versuche ableiten zu können.

Zur praktischen Arbeit steht ein zentrales Gateway-Modul zur Verfügung, mit dem die Informationsweiterleitung zwischen den verschiedenen Kfz-Bussystemen analysiert werden soll.

Dafür ist zuerst festzustellen, welche Bussysteme auf dem Gateway verfügbar sind. Hierzu muss unter anderem die Pinbelegung der einzelnen am Gateway-Modul verfügbaren Interfaces ermittelt werden. Insbesondere soll überprüft werden, ob die CAN- und FlexRay-Busse vorhanden sind. Ist dies der Fall soll mit den neuen Erkenntnissen, weitere Tests durchführt werden, um die Kommunikation zwischen den Bussystemen zu überprüfen. Im weiteren Verlauf ist anzustreben, mit der Analyse des Gateways Schwachstellen oder Probleme der Bussysteme aufzudecken. Sofern das nicht zum gewünschten Ergebnis führt, soll das Gateway geöffnet werden, um Informationen über die verbaute Elektronik zu erlangen. Mit hoher Wahrscheinlichkeit sind Mikrochips zur Steuerung und Speicherelemente enthalten, aus denen verwertbare Daten extrahiert werden könnten.

3.1 Simulationserstellung mit CANoe

Nach der theoretischen Analyse des Bussystems FlexRay erfolgt mithilfe dem Softwaretool CANoe die praktische Betrachtung. Dazu besteht die Möglichkeit, sich eine Beispielkonfiguration eines FlexRay-Bussystems darstellen zu lassen. Zu diesen Bestandteilen der Konfiguration gehören zum Beispiel die:

- verschiedenen Panels für die Fahrzeugkomponenten:
  - der Konsole zum Betätigen von Kupplung und Bremse,
  - des Amaturenbretts zur Anzeige,
  - der Bremslichter
- Auswertungsanzeigen:
  - Grafik-Fenster zur Veranschaulichung der Signale in einem xy-Diagramm
  - State Tracker zur Visualisierung der Zustandsübergange, Signale etc.
  - Trace-Fenster zur Protokollierung der einzelnen Prozesse

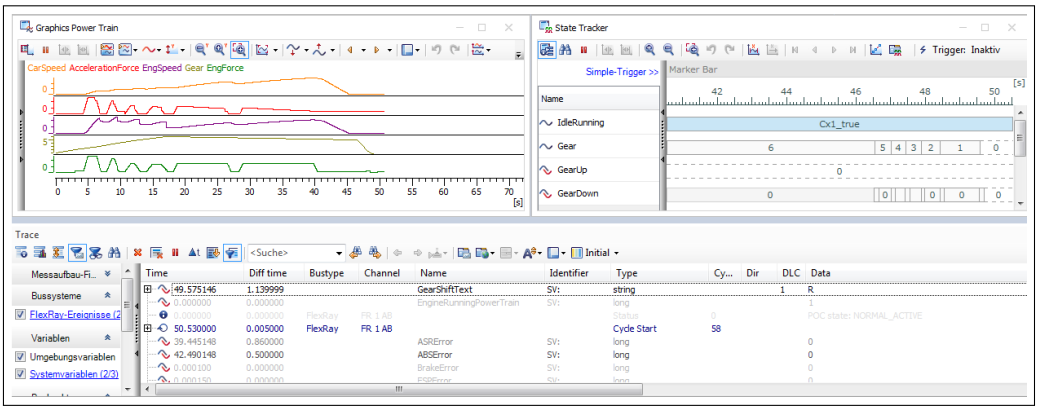


Abbildung 3.1: Auswertungsanzeigen aus Beispielkonfiguration [Vector Informatik GmbH, 2018]

Die oben genannten Anzeigen sind jedoch lediglich für den Visualisierungs-und Evaluierungsaspekt geeignet (siehe Abbildung 3.1). Die genaue Funktionsweise dahinter wird über die vorhandene Datenbank (DB) im AUTOSAR Explorer deutlich.
Diese DB-Datei gewährleistet es, Einblicke in den spezifischen FlexRay-Zeitplan zu erlangen. Es wird detailliert aufgelistet, welche Nachrichten in welchem Segment, Zyklus und Slot gesendet werden und wie dadurch der deterministische Kommunikationsablauf zustande gebracht wird.
Zusammenfassend ist es über diese Software möglich gewesen, für das ZGW weitere Steuergeräte für die Kommunikation zu simulieren und damit verschiedene Szenarien zu konzipieren, auszuprobieren und zu evaluieren, siehe Ergebnis 4.1.

3.2 Analyse des zentralen Gateway-Moduls

Die folgenden Methoden handeln davon, was direkt am zentralen Gateway-Modul über die Stecker getestet wurde. Es wird bei jedem Versuch detailliert erklärt, mit welcher Hardware oder Software gearbeitet wurde. Die daraus entstandenen Ergebnisse für die verschiedenen Tests sind vorwiegend im Kapitel 4 Ergebnisteil aufgelistet.
Da es sich bei dieser Arbeit um ein modernes und wenig erforschtes Thema handelt, basierten viele Tests auf den Ergebnissen des vorherigen Versuchs. Das bedeutet, dass einige Tests ohne des Resultats des vorherigen schwer nachvollziehbar wären. Deswegen ist teilweise das Resultat zum jeweiligen Versuch, bereits im Methodenteil kurz mit aufgegriffen.

3.2.1 Ermittlung der Pinbelegung

Die Website BMW.info hat neben der Funktionsbeschreibung auch Informationen über einen Schaltplan sowie Details über eine Pinbelegung für das zentrale Gateway-Modul [NewTIS.info, 2018b]. Auf Basis dieser Belegung wurden insbesondere die Pins am Stecker A51*1B für die ersten Versuche näher betrachtet.

Pin	Art	Bezeichnung/Signalart	Anschluss/Messhinweise
19	M	Masse	Massepunkt
39	E	Versorgung Klemme 30F	Sicherung F3
41	E/A	Ethernet Datenleitung	Diagnosesteckdose
42	E	Wecksignal Klemme 15	Car Access System
44	E/A	Diagnose Bus-Signal	Diagnosesteckdose
45	E/A	Diagnose Bus-Signal	Diagnosesteckdose
46	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung
47	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung
48	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung
49	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung
50	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung
51	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung

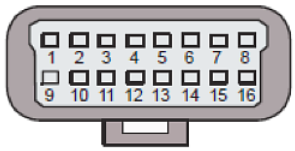
Tabelle 3.1: Pinbelgungsausschnitt [NewTIS.info, 2018b]

Die Tabelle 3.1 zeigt einen Ausschnitt der wichtigsten Pins. Sie gibt Hinweise über die Art, die Bezeichnung/Signalart und über den Anschlusstyp. Die vollständige Tabelle ist im Anhang A zu sehen.

Zur Überprüfung, ob die Informationen eines ZGMs auch mit dem des ZGWs übereinstimmen, wurden alle Pins einzeln nacheinander mit Hilfe eines Oszilloskops getestet. Gleichzeitig konnte so geprüft werden, ob relevante Ausgangssignale über die Pins gesendet werden. Speziell der Aspekt der Signalübertragung ist bei den verschiedenen CAN-Bussen besonders betrachtet worden.
Um die Pins testen zu können, musste davon ausgegangen werden, dass der Pin 19 für die Masse, der Pin 39 für die Versorgungsspannung, als auch der Pin 42 für das Wecksignal zutreffend sind.

3.2.2 Auslesen mit Diagnosegerät

Ein weiterer Schritt, um an Informationen im Gateway-Modul zu gelangen, ist das Auslesen mithilfe des Diagnosegerätes, namens ABRITES von automotive solutios gewesen. Dieses Werkzeug mit der Modellnummer: AVDI72ANC kann insbesondere benutzt werden, um an Fehlermeldungen zu gelangen, die in einem zentralen Gateway-Modul abgespeichert sind. [Abrites LTD, 2014] Dafür benötigt das Tool eine genormte **On-Board-Diagnose (OBD)-II-Buchse** [International Organisation for Standardization, 2016], die im Auto leicht zugänglich ist und sich meist im Fahrerfußraum, in der Mittelkonsole oder am Armaturenbrett befindet. [Reif, 2011]



Pin	Bedeutung
1, 9	nicht von OBD belegt
2, 10	Datenübertragung nach SAEJ 1850
3, 11	Eth Rx+, Eth R-
4	Fahrzeugmasse
5	Signalmasse
6, 14	CAN- High Leitung, CAN-Low Leitung
7, 15	K-Linie, L-Linie
8	Eth Aktivierung
12, 13	Eth Tx+, Eth Tx-
16	Batterie-Plus

Abbildung 3.2: OBD-II-Buchse mit Pinbelegung modifiziert nach [Reif, 2012] und [Zimmermann und Schmidgall, 2014]

Die Abbildung 3.2 zeigt eine OBD-II-Schnittstelle, inklusive ihrer Pinbelegung. Für das Auslesen wurde demzufolge eine passende Schnittstelle entwickelt, damit das Gateway mit einem OBD-Stecker verbunden werden konnte. Die beanspruchten Pins des Steckers waren folgende:

- Pin 4 für die Fahrzeugmasse
- Pin 5 für die Signalmasse
- Pin 6 für CAN-High
- Pin 14 für CAN-Low
- Pin 16 für Batterie-Plus

In der Tabelle 3.2 ist aufgelistet, wie die Pins des Steckers und des Gateways verbunden werden mussten.

	Pin am OBD-Stecker		Pin am ZGW	Netzteil	Kabelfarbe
4	Fahrzeugmasse	19	Masseleitung		lila
5	Signalmasse	19	Masseleitung		lila
		19	Masseleitung	Minuspol	blau
6	CAN-High	44	D-CAN-High		gelb
14	CAN-Low	45	D-CAN-Low		grün
16	Batterie-Plus			Pluspol	rot
		39	Versorgung	Pluspol	rot
		42	Wecksignal	Pluspol	weiß

Tabelle 3.2: Pin-Verbndung für OBD-Stecker und ZGW [NewTIS.info, 2018b],[Reif, 2012]

Der Aufbau mit den benötigten Komponenten zum Auslesen ist in Abbildung 3.3 schematisch zu sehen. In Korrespondenz dazu zeigt Abbildung 3.4 den Aufbau aus dem realen Versuch. Dabei ist das Diagnosegerät, welches mit dem Gateway und dem OBD(II)-Stecker verbunden ist, dargestellt.

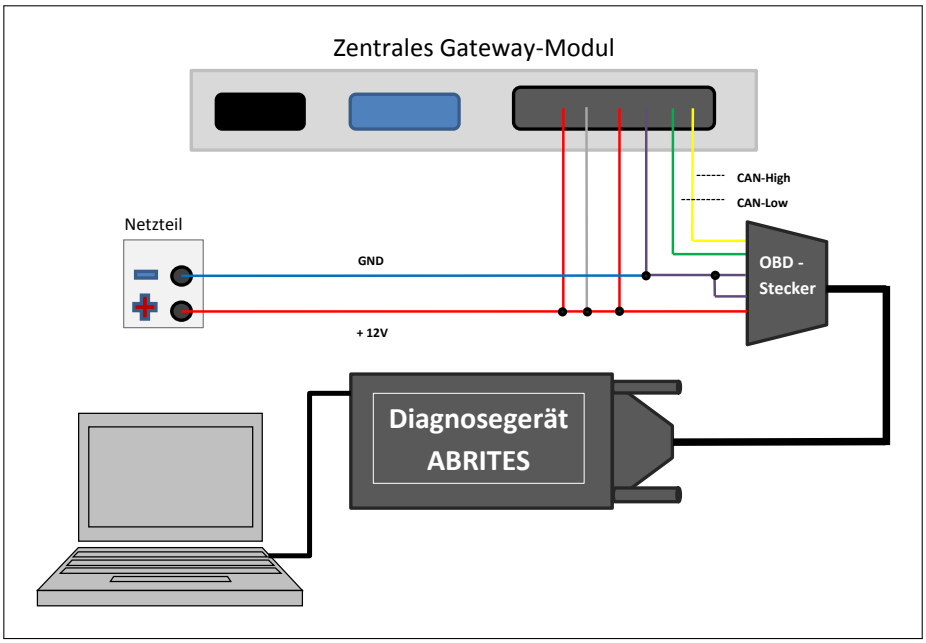


Abbildung 3.3: Schematischer Aufbau zum Auslesen des Gateways [Eigene Abbildung]

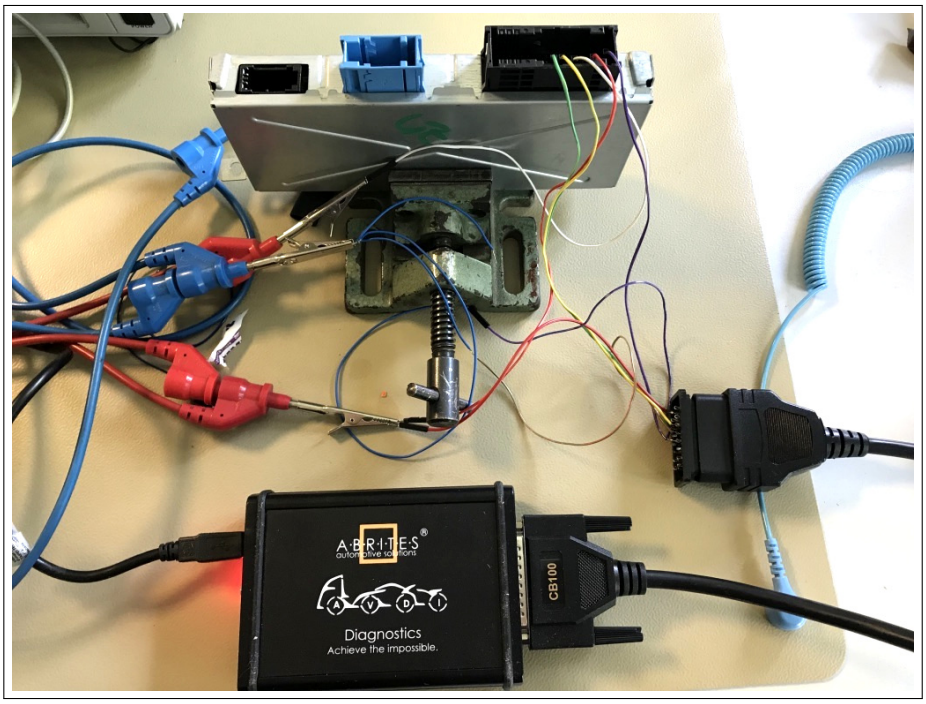


Abbildung 3.4: Aufbau zum Auslesen des Gateways [Eigene Abbildung]

Neben der Hardware ist die Software von ABRITES zum Auslesen notwendig, wofür die BMW-Erweiterung hinzugefügt werden muss. Nach dem Anschalten des Gateways und dem Starten der Simulation waren verschiedene Informationen aus der Software zu entnehmen, die im Ergebnisteil zu finden sind (siehe Kapitel 4.2).

3.2.3 Analyse der vorhandenen CAN-Busse

Auf Grundlage der inzwischen bekannten Pinbelegungen konnte davon ausgegangen werden, dass auf dem Gateway der Diagnose-CAN, der Powertrain-CAN und die zwei Komfort-CANs vorhanden sind.

3.2.3.1 Test der einzelnen Busse

Damit diese verschiedenen Busse mit dem CANoe-Tool getestet werden konnten, mussten einige Vorkehrungen getroffen werden. Dazu gehörten neben der CANoe-Lizenz:

1. die Anschaffung der verschiedenen Bus-Interfaces: für CAN zum Beispiel das VN1610 von Vector [Vector Informatik GmbH, 2017]
2. das Zusammenstellen der Stecker für das VN1610 zum Gateway, benötigte Leitungen sind dafür:

    - CAN-Low, CAN-High und
    - die Masse

3. Starten einer CAN-Konfiguration bei CANoe

    - Simulationsaufbau mit den benötigten Knoten (Bussen) und einem Interaktiven Generator hinzufügen
    - Hardware: der Bus-Hardware ein jeweiliges Bus-Interface inklusive Baudrate zuordnen
    - Arbeitsmodus: Realer Bus einstellen
    - über Interaktiven Generator (IG) ein Frame zum Versenden erstellen
    - Datenbank entwickeln mit den erstellten Netzknoten und den dazugehörigen Nachrichten und den beanspruchten Signalen

Dieser Versuchsaufbau ist für die Tests jeweils am einzelnen PT-CAN, dem K-CAN und dem D-CAN vorbereitet worden. Durch die Arbeit mit dem Oszilloskop ist bereits bekannt, welche Signalleitung an welchen Pin verläuft, deswegen wurden die Busse wie in Tabelle 3.3 verbunden.

Bus	CAN-High	CAN-Low
Diagnose-Bus	Pin 44	Pin 45
Powertrain-Bus	Pin 46	Pin 47
Komfort-Bus	Pin 48	Pin 49

Tabelle 3.3: Zuordnung der CAN-Leitungen [Eigene Ermittlung]

Wie schon in den Versuchen davor, benötigte das Gateway ein Aufwachsignal, eine Masseleitung und eine Versorgung mit Strom. Nach der Inbetriebnahme startete die Simulation. Das Trace-Fenster von CANoe hatte sowohl Signale vom Powertrain-Bus, als auch vom Komfort-Bus aufgezeichnet, jedoch keine vom Diagnose-Bus. Dargestellt sind die Ausgaben im Ergebnisteil 4.3.2.

Empfang beider Bussignale gleichzeitig

Für den Test beide Signale des PT-CANs und K-CANs gleichzeitig zu erfassen, wurde ein weiterer Stecker für das Interface VN1610 entwickelt. Für den Test hatte man einen Bus mit dem CANoe auf einem PC verbunden. Der andere Bus war an einen Laptop angeschlossen, auf dem ebenfalls eine CAN-Konfiguration über CANoe lief. Der Aufbau musste so gestaltet werden weil man nicht zwei unterschiedliche Simulationen auf einen Endgerät starten konnte. Nach dem Anschalten des Gateways und dem Starten der beiden Konfigurationen empfingen beide Programme gleichzeitig Signale.

3.2.3.2 Überprüfung der Kommunikation zwischen den Bussen

Weiterleitung von Diagnosenachrichten

Für den Diagnose-Bus gibt es vereinheitliche Diagnosenachrichten namens Unified Diagnostic Services (UDS). Die sind insbesondere für Werkstätten gedacht, damit sie speziell Abfragen nach Fehlermeldungen machen können. Zu diesen Nachrichten gehört zum Beispiel die CAN-Nachricht mit der ID 0x7DF, die eine Anfrage an alle Steuergeräte sendet. Dementsprechend gibt es auch vordefinierte IDs für die Antworten auf die Anfragen. [Zimmermann und Schmidgall, 2014] Hierbei sollte überprüft werden, ob der PT- oder K-CAN diese Nachrichten vom Diangose-CAN erhalten und darauf antworten können.

Der zuvor bereits entwickelte Interaktive Generator und die erstellte Datenbank kommen für diesen Versuch zum Versenden verschiedener Anfragen und Antworten zum Einsatz. In Abbildung 3.5 sieht man einen Ausschnitt der mithilfe des von CANoe gestellten CANdb++Editor erstellten Datenbank.

Darin wurden die beanspruchten Steuergeräte hinzugefügt, die auch den Netzknoten im Simulationsaufbau entsprechen. Des Weiteren wurden die zu versendenden Nachrichten erstellt. Danach besteht die Möglichkeit im Hauptfenster dem Interaktiven Generator (IG) die Nachrichten aus der Datenbank hinzuzufügen. Während die Simulation gestartet ist ermöglicht der IG, im laufenden Betrieb des Gateways beziehungsweise der Konfiguration, Nachrichten zu verschicken.

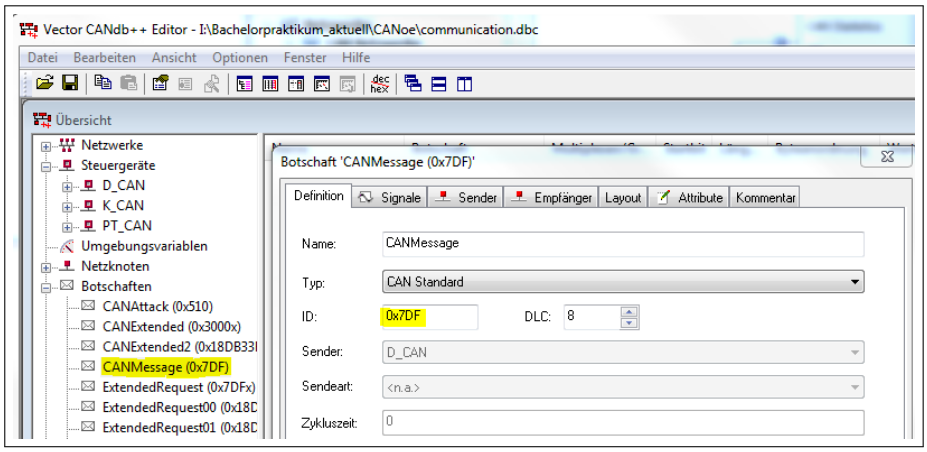


Abbildung 3.5: Ausschnitt aus selbst erstellter CANdb++Editor Datenbank im CANoe Tool [Eigene Abbildung]

Brute-Force Test mithilfe der CAN-IDs

Nachdem die Weiterleitung der Diagnosenachrichten sichergestellt wurde, stellte sich die Frage, welche noch weitergeleitet werden.
Dafür wurde ein CAPL-Programm geschrieben, dass vom Diagnose-CAN alle IDs aufsteigend, zum einen an den PT-CAN und zum anderen an den K-CAN schicken sollte. Damit konnte überprüft werden, welche Nachrichten neben den UDS durchgelassen werden. Dieses Programm enthielt für die Aufgabe eine Schleife, die die ID hochzählt, angefangen mit der Datenlänge=0 (Data Length Code (DLC)). Der dazugehörige Quelltext befindet sich im Anhang A.

Gleichzeitig wurde die Abfrage mitgelogged, mit dem Ziel einen Freischaltcode zu ermitteln. Zweck des Codes soll sein, anzuzeigen, ob die Nachricht weitergeleitet wird. Nach diesem Versuch fand zusätzlich noch die Abfrage vom K-CAN zum PT-CAN sowie umgedreht, statt. Dafür kam der selbe Programmcode zum Einsatz, wie bei der Nachrichtenabfrage vom Diagnose-Bus zuvor. Die zahlreichen Erkenntnisse aus den Versuchen sind im Kapitel 4.3.2.2 festgehalten.

Extended CAN

Neben den aus 11 Bit bestehenden CAN-IDs wurden die 29 Bit langen Extended CAN-IDs getestet. Diese konnten zwar erfolgreich generiert und von einem Bus abgeschickt werden, jedoch werden diese nicht von den anderen CAN-Bussen empfangen. Dafür wurde eine kleine Stichprobe zum einen zufällig ausgewählter und zum anderen bereits getesteter (Quelle: [Nutzername im Forum:mikeb, 2017]) Extended CAN-IDs ausgewählt. Eine komplexe Abfrage mit allen anderen IDs würde den Zeitrahmen in erheblichen Maßen sprengen.

3.2.4 Analyse des FlexRay-Busses

Zunächst sollte das gestellte Gateway darauf überprüft werden, ob FlexRay-Signale vorhanden sind. Dazu wurde noch mal die Pinbelegung von [NewTIS.info, 2018a] überprüft und speziell die Pins, bei denen es in der ersten Betrachtung schon Ausschläge am Oszilloskop zu sehen waren, genauer analysiert.

Analyse der FlexRay-Pins

Mithilfe des Messgeräts konnte feststellt werden, dass auf spezifischen Pins (siehe aktualisierte Pinbelegung 4.3.3.1) FlexRay-Signale zu erkennen sind. Für diese musste noch die Zuordnung zusammengehöriger Paare für die Signalübertragung erfolgen. Erneut wurde die Internetquelle NewTis.info dafür verwendet.
Mit diesen Informationen wurden verschiedene Pins getestet. Dafür sind diese jeweils in zweier-Paaren am Oszilloskop angesteckt und deren Signale in Kombination betrachtet worden. Diese waren je nach Versuch entweder beide gleichlaufend oder gegenläufig. Diese Informationen ergaben Hinweise, um welche Leitungsart es sich handelt. Gesucht war in diesem Fall ein gegenläufiges Signal, weil das bedeutete, dass jeweils eine Leitung Bus-Plus und die andere Bus-Minus sein musste.

3.2.4.1 FlexRay am CANoe

Damit FlexRay-Signale vom CANoe-Tool empfangen werden können, benötigt es wieder ein bestimmtes Interface. Für die Versuche ist das Vector Bus-Interface VN7610 verwendet worden, welches sowohl CAN- als auch FlexRay-Signale verwerten kann. Zum Verwenden dieser Hardware musste ebenfalls ein passender Stecker erstellt und nach der Pinbelegung in Abbildung 3.6 entsprechend mit Kabeln verbunden werden.

Pin	Belegung
1	1051cap CAN Low
2	1082cap BM A
3	FR GND
4	1082cap BM B
5	Nicht verbunden
6	CAN GND
7	1082cap BP A
8	1082cap BP B
9	1051cap CAN High

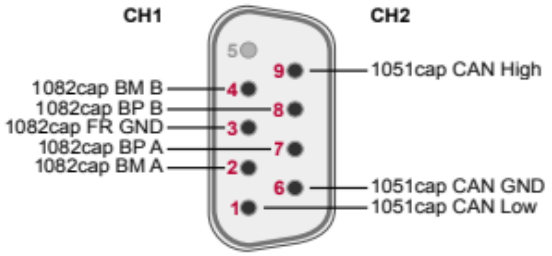


Abbildung 3.6: Pinbelegung Bus-Interface VN7610 [Vector Informatik GmbH, 2016]

Neben der benötigten Hardware musste auch die Software bearbeitet werden, damit CANoe auch die Signale von dem Bussystem FlexRay auswerten kann. Für diese Vor-

aussetzung wurde eine entsprechende FlexRay-Konfiguration neu generiert. Diese hatte dieselben Steuergeräte, wie bereits in der getesteten Bussystem-Simulation (siehe 4.1), nur mit dem Unterschied, dass kein simulierter Bus, sondern ein realer Bus eingestellt worden ist.

Danach erfolgten verschiedene Tests, in denen die ausgewählten Kanäle in unterschiedlichen Kombinationen analysiert wurden. In der Tabelle 3.4 sind alle verschiedenen Verknüpfungen der FlexRay-Pins aufgelistet, die am Stecker 1*B gemacht wurden.
Die ausführliche Auswahl der verwendeten Pin-Kombinationen ist im Kapitel 4.3.3.2 dargelegt. Zu beachten ist, dass neben den aufgezählten FlexRay-Pins immer die Pins für Stromversorgung, Wakeup-Signal und Masse mit angesteckt waren, wie auch bei den CAN-Versuchen zuvor.

Test	Verwendete Pin-Kombination	Information
0	13 A und 35 B	13 auf Kanal A und 35 auf Kanal B
1	13, 16 und 31 + PT-CAN	
2	13 (BM) A und 16 (BM) B	nur Bus-Minus (BM)
3	31 (BP) und 35 (BP) + PT-CAN	nur Bus-Plus (BP)
4	13 (BM) und 34 (BP)	auf Kanal A
5	13 (BM) und 35 (BP)	auf Kanal A
6	16 (BM) und 34 (BP)	auf Kanal B
7	16 (BM) und 35 (BP)	auf Kanal B
8	13 (BM) A, 31 (BP) A, 16 (BM) B und 34 (BM) B	

Tabelle 3.4: Testen der FlexRay-Pin-Kombinationen [Eigene Ermittlung]

3.2.4.2 Kommunikationsversuch zwischen FlexRay und CAN

Des Weiteren war der Versuch, FlexRay und CAN miteinander Daten austauschen lassen, zu implementieren. Es sollte wie in den Testreihen zuvor ein Brute-Force Test mithilfe CAN-IDs vom Sender-PC stattfinden, um herauszufinden, ob spezielle CAN-Nachrichten beim FlexRay (Empfänger-PC) ankommen. Die kombinierten Pins für den Test sind wieder in einer Tabelle 3.5 zusammengefasst.

Test	Verwendete Pin-Kombination	Information
9	13 (BM) A und 35 (BP) B	Zusätzliche FlexRay-Nachricht
10	13 (BM) A und 35 (BP) B	manuelles Aufwachen
11	13 (BM) A und 35 (BP) B + K-CAN	Abfrage vom K-CAN
12	13 (BM) A und 35 (BP) B + PT-CAN	Abfrage vom PT-CAN

Tabelle 3.5: Testen der FlexRay-Pins mit CAN [Eigene Ermittlung]

Den dazugehörigen Aufbau des Versuchs mit den verschiedenen Interfaces, dem Gateway, Netzteil und den zwei Endgeräten, bestehend aus Sender- und Empfänger-PC, sieht man in Abbildung 3.7 schematisch dargestellt.

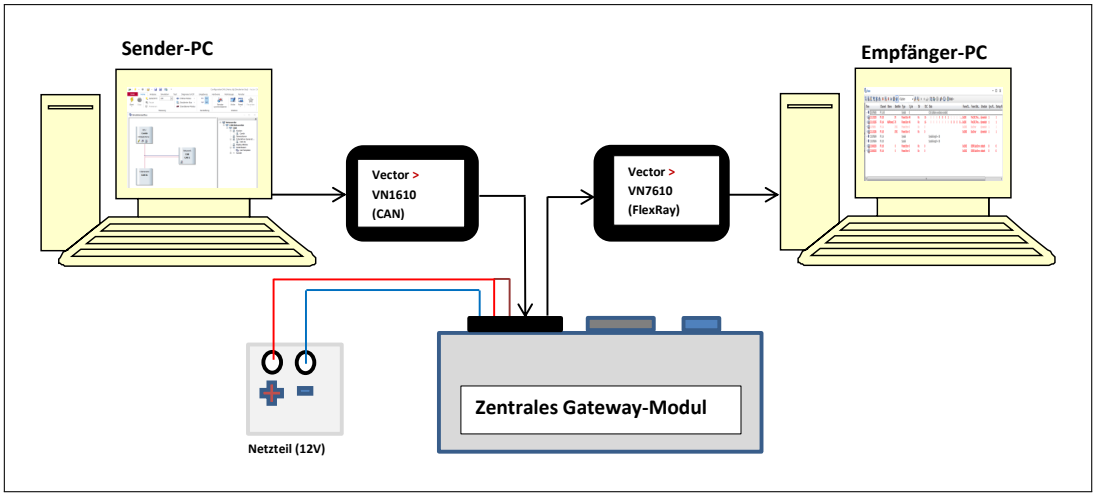


Abbildung 3.7: Schematische Darstellung der Abfrage von CAN auf FlexRay [Eigene Abbildung]

Extended CAN

Der FlexRay-Bus wurde ebenfalls auf den Empfang von Extended CAN-IDs stichprobenartig getestet. Wie schon in den CAN-Kommunikationsversuch aus Kapitel 3.2.3.2 wurde ein Request als Extended Data über den Interaktiven Generator von CANoe gesendet.

3.2.4.3 Synchronisationsknoten

Wie bereits in den Grundlagen beschrieben, benötigt FlexRay zur Synchronisation mehrere Knotenpunkte. Daher war eine Annahme, dass fehlende Synchronisationsknoten Auslöser des geringen Datenaufkommens waren. Möglicherweise sind die erstellten Knoten auf Softwareebene im CANoe nicht ausreichend gewesen. Deswegen wurde auf Hardwarebene eine Erweiterung entwickelt. In der Quelle NewTIS.info ist neben dem ZGM das Integrated Chassis Management (ICM) und Dynamic Stability Control (DSC) unter anderem als Synchronisationsknoten mit aufgelistet [NewTIS.info, 2018a]. Deswegen wurden am ZGM die Bus-Signale für das DSC, ICM und Bus-Plus beziehungsweise Bus-Minus verbunden und an dem Vector Tool VN7610 angeschlossen (siehe Abbildung 3.8).

Am Bus-Plus-Signal vom Bus-Interface wurden folgende Pins miteinander verknüpft:

- Bus-Plus FlexRay-Pin 31
- Bus-Plus DSC-Pin 14
- Bus-Plus ICM-Pin 32

Am Bus-Minus-Signal vom Interface VN7610 wurden folgende Pins miteinander verknüpft:

- Bus-Minus FlexRay-Pin 13
- Bus-Minus DSC-Pin 15
- Bus-Minus ICM-Pin 33

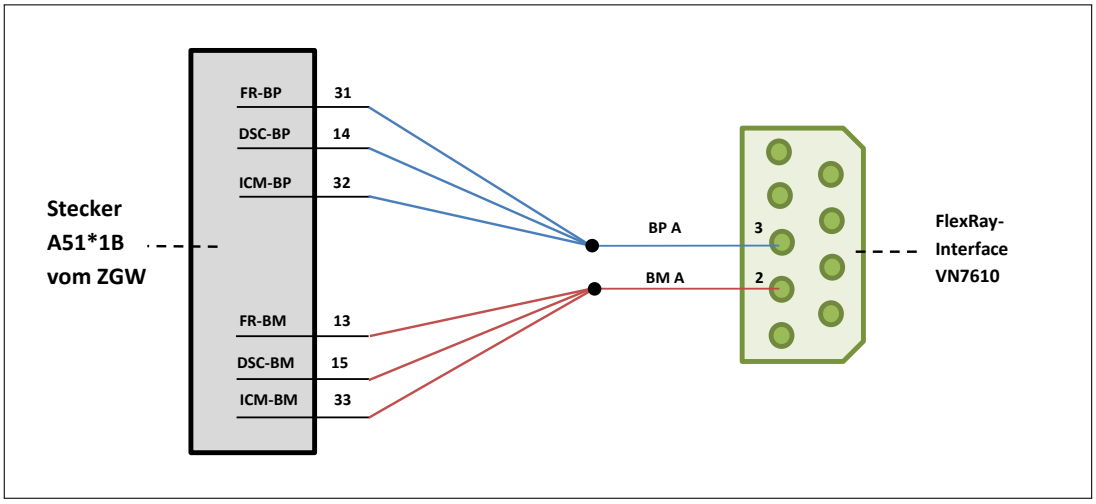


Abbildung 3.8: Verbindung des ZGWs mit Bus-Interface [Eigene Abbildung]

3.3 Reverse Engineering

Nachdem alle Tests die von außen über die Stecker an dem zentralen Gateway abgeschlossen waren, wurde das ZGM geöffnet. Ziel ist es gewesen, an weitere Informationen bezüglich FlexRay über die verarbeiteten Mikrochips und Speicherelemente auf der Platine zu gelangen. Nach dem Entfernen des Metallgehäuses konnten die Bezeichnungen der vorhandenen Chips auf der Platine mithilfe eines Lichtmikroskops abgelesen werden. Die ermittelten Chips sind mit ihrer vollständigen Bezeichnung im Kapitel A vorzufinden.

Auslesen ausgewählter Mikrochips

Nach der visuellen Begutachtung der Leiterplatte wurden einzelne Chips ausgelesen, darunter waren EEPROM, Mikrocontroller und der Prozessor. In den folgenden Unterkapiteln werden die spezifischen Verfahren zum Extrahieren der Daten aus den jeweiligen Segmenten, näher erläutert. Diese Methoden unterschieden sich grundlegend darin, entweder den Mikrochip von der Platine zu lösen und extern Auszulesen oder die Daten aus den Schaltkreisen mit einem Programmier- beziehungsweise Debugging-Interface zu extrahieren. [Le-Khac et al., 2018]

3.3.1 EEPROM

Als Erstes wurde der Electrically Erasable Programmable Read-only Memory (EEPROM) versucht auszulesen. Dazu wurde der elektrisch löschbare Speicherbaustein von der Platine ab gelötet. Für das Auslesen des Speichers stand das Programm buSpy von Denny Vogt zur Verfügung [Vogt, 2018].
Der EEPROM wurde in eine Halterung eingesetzt, sodass er gemäß den Vorgaben des zugehörigen Datenblattes verdrahtet werden konnte. [Microchip technology Inc., 2017] Der Vorgang war erfolgreich, weil Daten mithilfe des Tools ausgelesen werden konnten. Diese Dateien liegen als Hexzahlen mit ASCII Werten zum Analysieren bereit.

3.3.2 Mikrocontroller

Als Nächstes ist der ATmega48-Mikrocontroller von der Firma Atmel zum Auslesen näher betrachtet worden. Diese Micro-Controller-Unit (MCU) musste im Vergleich zum EEPROM nicht abgelöst werden. Stattdessen kommt hier die In-System Programming (ISP) Variante zum Einsatz. Das ISP ist ein herstellerspezifisches Protkoll zum Auslesen von ATmega Controllern.
Allgemein sind dafür schon auf vielen Leiterplatten spezifische Debugschnittstelle vorhanden. Diese können zum testen, debuggen oder entwickeln genutzt werden. Ist diese Steckverbindung nicht vorhanden, kann eine hinzugefügt werden. [Le-Khac et al., 2018]

Pin	Signal Name	Signal Beschreibung
15	MOSI	Master OUT, Slave IN
16	MISO	Master IN, Slave OUT
17	SCK	Serial Clock
29	RESET	

Tabelle 3.6: Relevante Pins zum Auslesen des ATmega [Atmel, 2016]

Dafür verwendet man als Zugang spezifische Testpunkte auf der Platine, die zu Pins des Mikrocontrollers führen. Hierzu mussten genau die Testpunkte gefunden und kontaktiert werden, die für das Auslesen mithilfe eines Lesegerätes relevant sind. Diese benötigten Kontaktstellen sind in Tabelle 3.6 aufgelistet. Zum Auslesen mussten die Testpunkte auf der Platine mit Kabeln kontaktiert werden und nacheinander an einer Pinreihe befestigt werden.

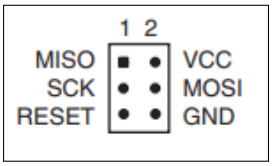


Abbildung 3.9: ISP-6-Pinout [Atmel Corporation, 2006]

Nach dieser Vorbereitung kam das Auslesegerät AVR JTAGICE mkII von der Firma ATMEL zum Einsatz. Dieses wurde nach dem ISP-6-Pinout, zu sehen in Abbildung 3.9, mit den relevanten Pins beziehungsweise den an der Platine befestigten Kabeln verbunden. In Abbildung 3.10 ist dargestellt wie das Auslesegerät mit der Platine verbunden ist.

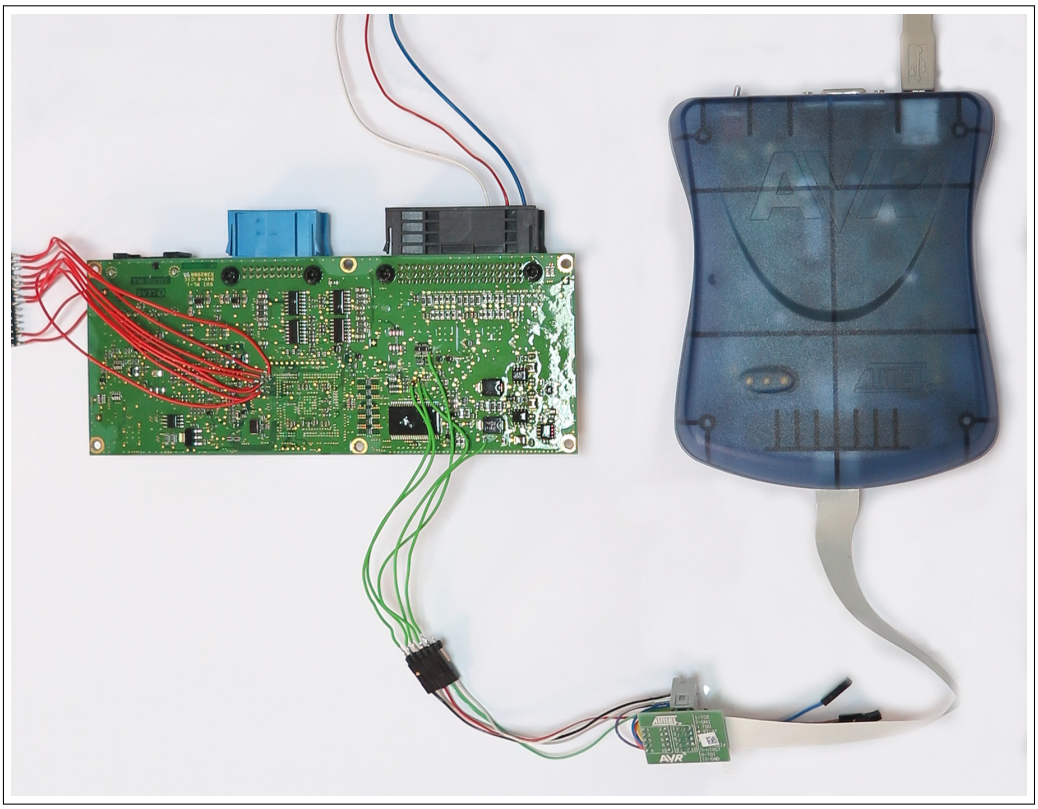


Abbildung 3.10: Aufbau zum Auslesen des ATmega48 mit JTAG [Eigene Abbildung]

Danach konnte das Auslesegerät mit dem Computer verbunden und die Software ATMEL Studio gestartet werden. Mit diesem Programm lässt sich das verwendete Gerät sowie der auszulesenden Chip explizit auswählen. Als Letztes benötigte das Gateway über die Pins 39 (Versorgung) und 42 (Wake-Up) sowie dem Masse-Pin noch den Strom. Danach ließ sich über die Software zum einen der Flashspeicher und zum anderen der vorhandene EEPROM, der im MCU enthalten war, auslesen.

Zur Interpretation der Daten wurden der Flash Inhalt dissassembliert. Dadurch wurde die Binärdatei in eine Assemblerdatei konvertiert. Daraus entstand eine Flash.asm-Datei mit diversen Befehlen. Aus dieser Auflistung von Anweisungen konnten anhand des Datenblatts des ATmega48 verschiedene Informationen aus den Kommandos entnommen werden (siehe Kapitel 4.4.2). [Atmel, 2016]

3.3.3 Prozessor

Für weitere Details wurde der Prozessor MPC5567 auf der Platine analysiert. Dabei sollte das Vorgehen zum Auslesen an der Central-Processing-Unit (CPU) ähnlich dem, des Mikroprozessors angewendet werden. Hierzu wurde der Joint Test Action Group (JTAG) zum Extrahieren der Daten genutzt. Ebenfalls zeigte das Datenblatt die CPU-Pins, die zum Programmieren verwendet werden können. [Freescale Inc., 2012]

Die relevanten Pins für den Prozessor sind:

- Test Data In (TDI)
- Test Data Out (TDO)
- Test Clock (TCK)
- Test Mode Select (TMS)
- TEST
- RESET

Problematisch in diesem Fall ist jedoch, dass alle Lötstellen nicht offensichtlich mit Testpunkten verbunden sind. Deswegen wurden alle in Frage kommenden Testpunkte in der Umgebung kontaktiert. In Abbildung 3.11 sind die roten Kabel für die Testpunkte des Prozessors verwendet worden.

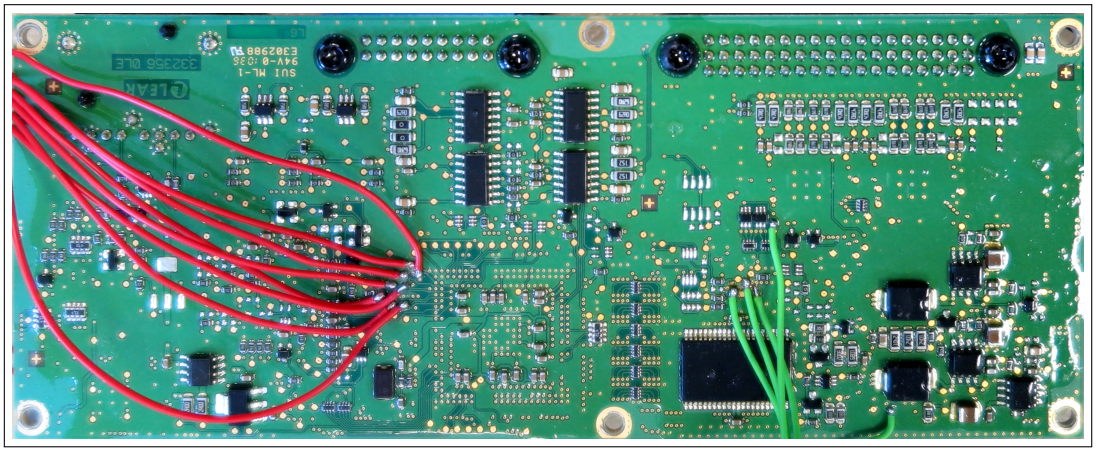


Abbildung 3.11: Kontaktierte Testpunkte für MCU und CPU [Eigene Abbildung]

Um zu überprüfen, welche kontaktierten Pins zu welchem Testpunkt gehören, wurde zusätzlich ein Arduino mit einem geeigneten JTAG-Testprogramm genutzt. Dieses Programm konnte man dem Onlinedienst GitHub entnehmen und heißt jtagenum.ino. Dieses Werkzeug besitzt die Funktion, die Zugehörigkeiten, der Pins zu ermitteln. Als Voraussetzung dafür, müssen alle möglichen Signalnamen dem Programmcode übergeben werden.

Die Recherche nach einem Auslesegerät für den MPC hat XPROG ergeben, welches auch beschafft worden ist [ELDB electronics store, 2011-2018].

Die Inbetriebnahme des Geräts gestaltete sich schwierig, aufgrund von fehlender Dokumentation. Außerdem hatte Antivirus angeschlagen, weswegen ein extra Rechner aufgesetzt wurde.

Auf der mitgelieferten CD waren verschiedene Schaltpläne für diverse Chips abgespeichert. Der vorgesehene Schaltplan für den MPC5567 (siehe Anhang Abbildung A.9) unterscheidet sich in mehreren Punkten zu den Angaben im Datenblatt. Dafür enthielt der Schaltplan für den Chip mit der Bezeichnung MICRONASCDC32 zumindest die benötigten Pins (TDI, TCK, TDO, TMS, Test und Reset). Daher wurde der Prozessor nach dem Schaltplan in Abbildung A.10, im Anhang mit dem XPROG verbunden. Der reale Aufbau ist in Abbildung 3.12 zu betrachten.

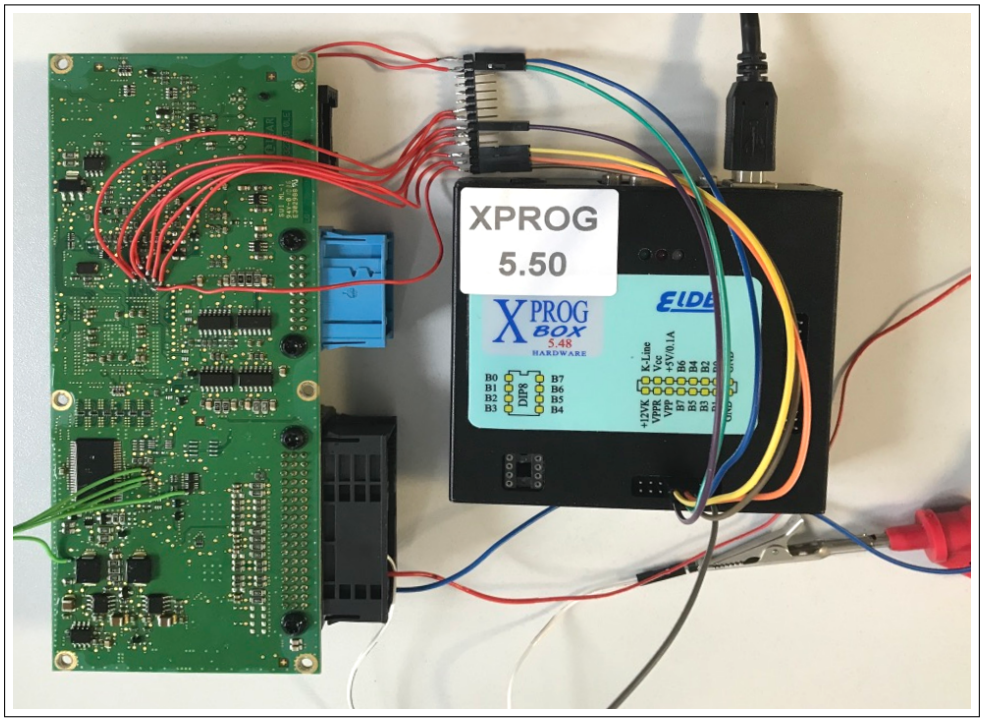


Abbildung 3.12: Auslesen des MPC5567 mit XPROG [Eigene Abbildung]

4 Ergebnisteil

In diesem Kapitel werden chronologisch die Resultate aus den verschiedenen Versuchen wiedergegeben, die bereits im Methodenteil erläutert wurden. Dafür wird anfangs auf die Software CANoe, darauffolgend auf die Ergebnisse aus den Versuchen mit den CAN- und dem FlexRay-Bussystemen und zuletzt auf die Informationen aus den Mikrochips eingegangen.

4.1 CANoe

Mit den Informationen aus der Datenbank der Beispielkonfiguration ist es möglich gewesen, ein separates FlexRay-System aufzubauen (siehe Abbildung 4.1) und zu simulieren. Sie sollte die Aufgabe erfüllen, dass die Bremslichter leuchten, wenn die Bremspedale per Mausklick betätigt wird.

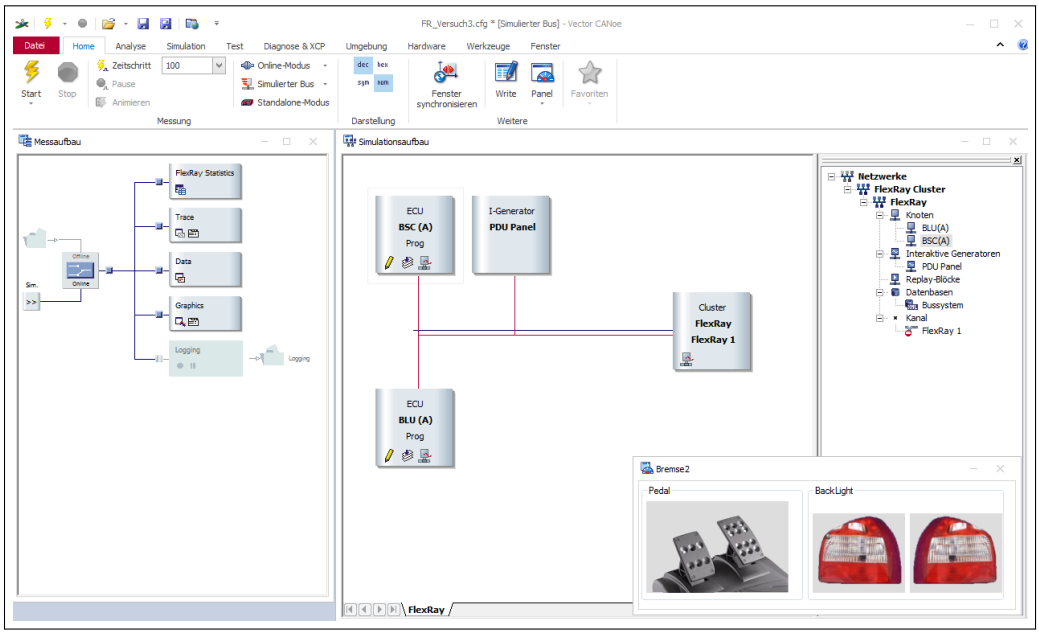


Abbildung 4.1: Ausschnitt der selbst generierten Konfiguration im CANoe-Tool [Eigene Abbildung]

Für die erste Simulation ist ein zweikanaliges FlexRay-System, mit zwei verantwortlichen Steuergeräten generiert worden. Diese ECUs hatten über die Datenbankdatei, die Nachrichten inklusive Signalen zum Senden und Empfangen zugeteilt bekommen.
Außerdem wurden zur Visualisierung der Aufgabenstellung, die entsprechenden Panels erstellt und spezifischen Funktionen im CAPL-Browser übertragen. Der Versuchsaufbau konnte erfolgreich in Betrieb genommen werden und ist mit in einer detaillierten Anleitung dem Anhang A hinzugefügt.

Des Weiteren dienten die unterschiedlich gestaltbaren Simulationsumgebungen, als Basis für die verschiedenen Tests am CAN- und FlexRay-Bus. Hinzuzufügen ist, dass durch das Trace-Fenster der Software die Möglichkeit gegeben ist die Ergebnisse der Versuche festzuhalten und zu evaluieren.

4.2 Auswertung der Informationen durch ABRITES

Nach der Inbetriebnahme und dem Starten ABRITES-Software konnte man das Fabrikat BMW auswählen. Danach hat das Programm unter anderem folgende Informationen aus dem Gateway ermittelt:

- die Gerätenummer: 379052
- der Hersteller: Loewe/Lear
- **die Fahrzeugidentifikationsnummer: WBAMX31060C524214**
- das Datum: 16.11.2010
- Anzahl der Fehlermeldungen : 9

Besonders relevant ist hierbei vor allem die Fahrzeugidentifikationsnummer, die häufiger unter der englischen Bezeichnung Vehicle-Identification-Number (VIN) bekannt ist. Die Nummer gibt weitere Details über das Gateway beziehungsweise das Fahrzeug, in dem das Modul eingebaut war.
Es konnte unter anderem ermittelt werden, dass das Gerät in dem Fahrzeug BMW 525 der Version F11 war, welches in Deutschland gebaut wurde und Kraftstoff Diesel benötigt. [VIN-Info Sp. z o.o, 2001-2018]
Außerdem brachte die Software ABRITES einige Fehlermeldungen aus dem Speicher des Gateways zum Vorschein. Diese sind in der Tabelle 4.1 aufgelistet und beschreiben, was für Probleme zwischen spezifischen Steuergeräten existieren.

Code	Steuergeräte	Beschreibung
CD0414	ZGM, K-CAN	Kommunikationsfehler
CD0468	ZGM, K-CAN2	Kommunikationsstörung
CD040A	ZGM, PT-CAN	Kommunikationsfehler
CD0487	ZGM	Synchronisationsvorgang für FlexRay fehlgeschlagen
CD0421	ZGM, FlexRay	Leitungsfehler auf Pfad 1
CD0423	ZGM, FlexRay	Leitungsfehler auf Pfad 2
801C15	Unknown DTC	auf Kanal A
CD1400	Botschaft (Fahrzeugzustand, 0x3A0) fehlerhaft, Empfänger ZGM, Sender Car Access System (CAS), DME	

Tabelle 4.1: Fehlermeldungen die mithilfe ABRITES ausgelesen wurden [Abrites LTD, 2014]

Durch diese Meldungen wird ersichtlich, dass das ZGW mehrere Probleme aufweist. Im Bezug zur Kommunikation zum K-und PT-CAN scheint nur ein Fehler zu sein, wohingegen eine Kommunikation zum K-CAN2 eine Störung vorliegt. Des weiteren kann der Analyse entnommen werden, dass FlexRay auf dem Gateway existiert aber es Probleme bei der Synchronisierung als auch auf den Leitungen gibt. Somit erschwert das die Auswertung der Ergebnisse aus den folgenden Tests, da die vorhandenen Störungen mit in die Resultate einwirken können.

4.3 Ergebnisse aus der Analyse des ZGW

4.3.1 Aktualisierung der Pinbelegung für CAN

Mithilfe des Oszilloskops konnte festgestellt werden, auf welchem CAN-Bus-Pin, jeweils die CAN-High- und CAN-Low-Leitungen liegen. Zudem hat man ermittelt, mit welcher Baudrate diese laufen. Die Ergebnisse sind in der Tabelle 4.2 wiedergegeben.

Pin	Art	Bezeichnung	Anschluss/Messhinweise	CAN-Zuordnung
44	E/A	Diagnose Bus-Signal	Diagnosesteckdose	CAN-High
45	E/A	Diagnose Bus-Signal	Diagnosesteckdose	CAN-Low
46	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung	CAN-High
47	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung	CAN-Low
48	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung	CAN-High 500kBaud
49	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung	CAN-Low 500kBaud
50	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung	CAN-High 100kBaud
51	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung	CAN-Low 100kBaud

Tabelle 4.2: Pinbelgungsausschnitt (CAN) [NewTIS.info, 2018b],[Eigene Ermittlung]

4.3.2 Ergebnisse der Tests mit den CAN-Bussen

4.3.2.1 Signale an den einzelnen Bussen

Nachdem man alle drei vorgestellten CAN-Busse aus Versuch 3.2.3 einzeln über die CANoe-Konfiguration analysiert hatte, gab es folgende Ergebnisse.

–> D-CAN

Die Protokollierung hat beim Diagnose-Bus keine Signale empfangen. Das liegt daran, dass über den Bus lediglich Diagnose-Abfragen gesendet werden. Demzufolge dient dieser Bus nur als Weiterleitung und enthält selbst keine Informationen.

–> PT-CAN

Im Vergleich dazu wurden beim Powertrain-CAN-Bus Signale registriert. Das Trace-Fenster registriert gleich zu Anfang eine CAN-Fehlermeldung, was mit dem Ergebnis von ABRITES einhergeht. Danach zeichnet es drei CAN-Nachrichten mit den **IDs 0x510, 0x3A5 und 0xC0** auf. Anhand der Protokollierung ist durch das Feld "Dir" erkennbar, dass es die Nachrichten empfangen hat, weil die Abkürzung Rx für Empfangsbotschaft steht. [Vector Informatik GmbH, 2018]
In Abbildung 4.2 sind die empfangenen Botschaften inklusive Zeitstempel, ID, Event Typ, Richtungsverlauf (ob Nachricht gesendet oder empfangen wurde), die Datenlänge sowie der Dateninhalt dargestellt.

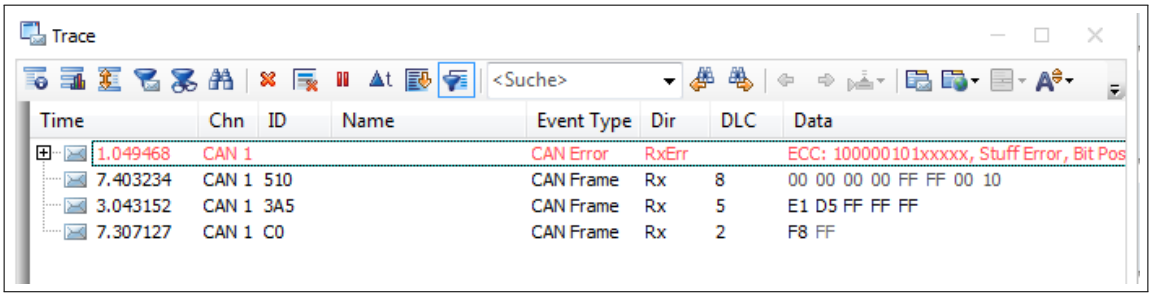


Abbildung 4.2: Trace Fenster für PT-CAN aus CANoe-Software [Eigene Abbildung]

–> K-CAN

Ähnlich dem PT-CAN registrierte die Software auch beim Komfort-CAN-Bus Signale. Der Unterschied hierbei ist, dass die Fehlermeldung nicht aufgezeichnet wird. Außerdem ist statt der CAN-Nachricht mit der ID 0xC0, die Botschaft mit der ID 0x6F0 vorhanden (siehe Abbildung 4.3).

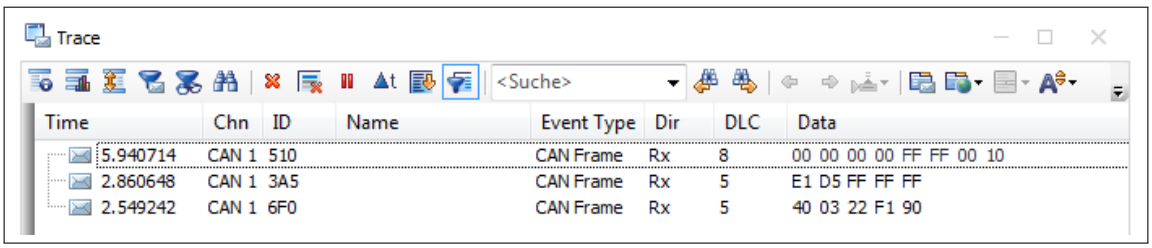


Abbildung 4.3: Trace Fenster für K-CAN aus CANoe-Software [Eigene Abbildung]

4.3.2.2 Nachrichtenaustausch zwischen des CAN-Bussen

Diagnose CAN –> PT- bzw. K-CAN

Der Versuch, die vereinheitlichten Diagnose Nachrichten abzuschicken, ist erfolgreich gewesen. Der K- sowie PT-CAN erhielten beide eine Empfangsbotschaft (Rx) mit 0x7DF und konnten ebenfalls Antwortnachrichten, zum Beispiel über die ID 0x7EF, zurückschicken. Somit konnte man dadurch feststellen, dass dieser Kommunikationsaustausch zwischen den CAN-Bussen über das ZGW möglich ist.

Aufgrund der Ergebnisse wurde geprüft, ob noch weitere Nachrichten weitergeleitet werden. Dafür ist der bereits im Methodenteil beschriebene Quellcode zur Abfrage abgeschickt worden. Das Ergebnis vom **Diagnose-CAN zum PT-und K-CAN** war nahezu identisch. Neben den IDs, die bei den einzelnen Bussen schon aufgetreten sind, waren alle Diagnosenachrichten von **0x7DF bis 0x7F1** empfangen worden. Lediglich beim PT-CAN ist eine weitere CAN-Message mit der ID 0x47E aufgetreten.

Daraus kann man schließen, dass die Busse beim Empfangen schon eine gewisse Filterung der CAN-Nachrichten vornehmen. In Folge dessen ist ersichtlich, dass bestimmte Sicherheitsmerkmale dadurch erreicht werden, in dem nicht alle IDs willkürlich durchgereicht werden.

K-CAN –> PT-CAN

Wesentlich mehr Nachrichten ergab das Abfragen aller IDs vom K-CAN –> PT-CAN und umgedreht. In der Tabelle 4.3 und 4.4 sind die Botschaften, deren Bedeutung bekannt ist, aufgelistet. Dabei ist zu beachten, dass es keine öffentlich zugänglichen Quellen über die Bedeutung von CAN-IDs gibt und die Informationen größtenteils aus Foren verwendet worden. Deswegen sind die Ergebnisse ohne Gewähr.
Im Anhang A ist zur Vollständigkeit die komplette Auflistung aller empfangenen Botschaften seitens des PT-CANs und des K-CANs vorhanden.

Insgesamt hat sich herausgestellt, dass genau die zwei Nachrichten mit der **ID 0x95 und 0x292** sowohl vom PT-CAN als auch vom K-CAN empfangen werden konnten. Damit kann über diese IDs einen direkte Verbindung vom Powertrain-und Komfort-CAN sichergestellt werden. Das Mitloggen der Versuche erbrachte jedoch keine neuen Erkenntnisse.

Auswertung der IDs vom K-CAN –>PT-CAN

ID	Bedeutung
0x95	unbekannt
0x1F6	Blinken
0x202	Dimmung
0x21A	Lampenzustand
0x226	Regensensor-Wischergeschwindigkeit
0x23A	Status Funkschlüssel [Nutzername im Forum:l86, 2018]
0x252	Wischerstatus
0x292	unbekannt
0x2FC	ZV (Zentralverriegelung) und Klappenzustand [Nutzername im Forum:l86, 2018]
0x380	Fahrgestellnummer
0x388	Fahrzeugtyp
0x3B0	Status Gang Rückwärts
0x3BE	Signal Nachlaufzeit-Stromversorgung [Nutzername im Forum:l86, 2018]

Tabelle 4.3: ID-Auswertung K-CAN –> PT- CAN [CarX24 - Andrea Faltinek, 2008]

Auswertung der IDs vom PT-CAN –> K-CAN

ID	Bedeutung
0x95	unbekannt
0xDE	Bedienung Sitzverstellung
0x1FE	Crash
0x292	unbekannt
0x2CA	Außentemperatur
0x2F7	Einheiten
0x2F8	Uhrzeit/Datum
0x328	Relativzeit-Sekunden seitdem die Batterie entfernt wurde
0x330	Kilometerstand/Reichweite/ Tank Level
0x381	Elektronischer Motorölmessstab [Hoppe, 2014]
0x394	Datenablage
0x3B3	Powermanagement Verbrauchersteuerung

Tabelle 4.4: ID-Auswertung PT-CAN –> K- CAN [Nutzername im Forum:chris30o0, 2015]

4.3.3 FlexRay

Nachdem die Kommunikation der CAN-Bussysteme über das Gateway sichergestellt werden konnte, kommen nun die Versuchsergebnisse zu FlexRay. Dabei stand vor allem im Fokus, einen FlexRay-Signal-Austausch über das ZGM darstellen zu können und der Versuch CAN und FlexRay miteinander kommunizieren zu lassen.

4.3.3.1 Analyse der FlexRay-Pins

Bei der Analyse der möglichen FlexRay-Pins mithilfe des Oszilloskops, hatte man an folgenden Pins des Steckers A51*1B ein kurzes und **zyklisches Signal** empfangen:

- Pin 13 (lt. [NewTIS.info, 2018a] Pinbelegung keine Funktion)
- Pin 16
- Pin 31 (lt. [NewTIS.info, 2018a] Pinbelegung keine Funktion)
- Pin 34
- Pin 35 (lt. [NewTIS.info, 2018a] Pinbelegung keine Funktion)

Demzufolge kann man daraus schlussfolgern, dass an diesen Pins FlexRay-Signale vorhanden sind. Im Vergleich dazu war am Pin 32 und 33 (Integrated Chassis Mangement) nur ein sehr kurzer Ausschlag beim Einschalten des Gateways messbar.

Um festzustellen, welche Pins zusammengehörig sind und ein gewünschtes gegenläufiges Signal ergeben, wurden zunächst Pin 31 und Pin 35 zusammen betrachtet. Die Oszilloskop-Ausgabe ergab ein gleichlaufendes Signal. Daraufhin wurden Pin 16 und Pin 31 in Kombination analysiert, was ein gegenläufiges Signal hervorbrachte, wie es in Abbildung 4.4 komplett zu sehen ist.

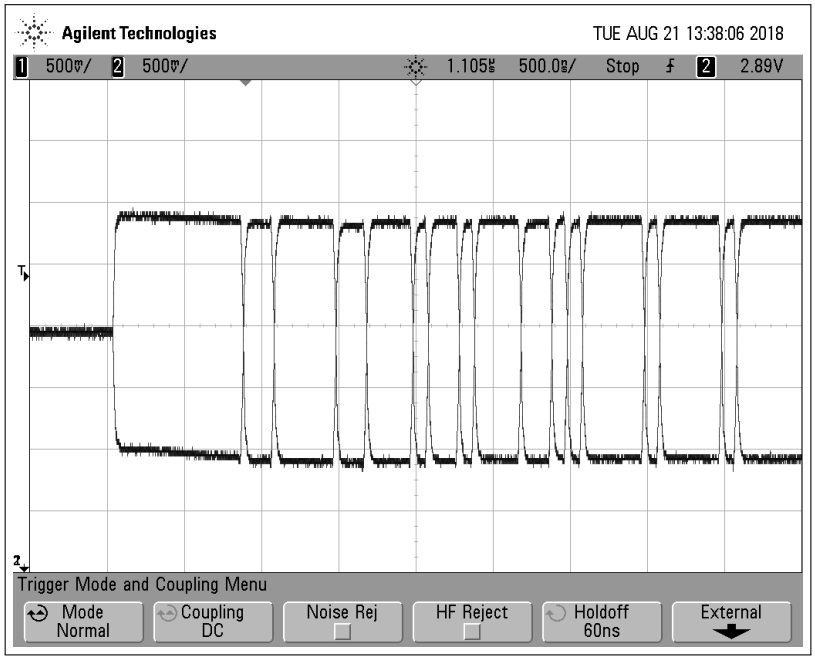


Abbildung 4.4: Komplettes FlexRay-Signal am Oszilloskop [Eigene Abbildung]

Ermittlung von Bus-Plus und Bus-Minus

Für die Ermittlung der Zugehörigkeit hatte man beim verwendeten Zeigerinstrument die Möglichkeit das Signal zu stoppen. Dadurch konnte beobachtet werden, welche Leitung beim Übergang vom rezessiven zum dominanten Buspegel zuerst in den höheren

beziehungsweise niedrigeren Signalpegel übergeht. In Abbildung 4.5 ist zu erkennen, dass das untere Signal, welches Pin 16 war, zuerst in den oberen dominanten Bereich geht. Dadurch entspricht das der **Bus-Minus**-Leitung. Das obere Signal am Pin 31 hingegen verläuft vom rezessiven Bereich nach unten, in den dominanten Abschnitt und ist dadurch **Bus-Plus** zuzuordnen. [Kurt Veit - Mixed Mode, 2007]

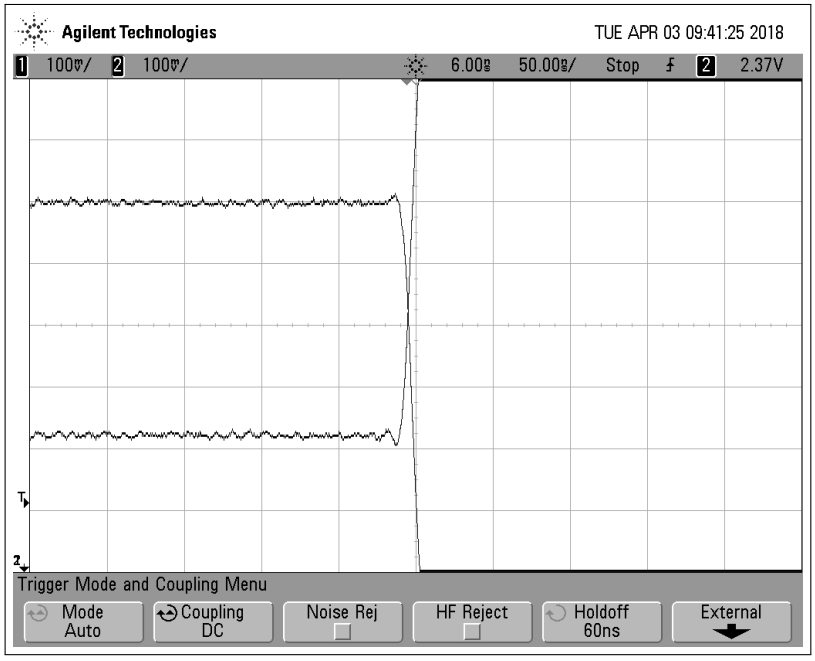


Abbildung 4.5: Ausschnitt eines FlexRay-Signals am Oszilloskop [Eigene Abbildung]

Diese Analyse des Signalverlaufs wurde mit allen weiteren Pins ausgeführt, auf denen ein zyklisches Signal zu erkennen war. Damit ergab sich diese Verteilung von Bus-Plus und Bus-Minus an den Pins:

Bus-Plus	Bus-Minus
Pin 14	Pin 13
Pin 31	Pin 15
Pin 32	Pin 16
Pin 34	Pin 33
	Pin 35

Tabelle 4.5: Pin-Zuordnung der FlexRay-Leitungen [Eigene Ermittlung]

Aktualisierung der Pinbelegung für FlexRay

Aufgrund der verschiedenen Tests an den Pins konnte die Pinbelegung um die Signalart der jeweiligen Pins erweitert werden, siehe unter der Spalte: “FlexRay-Zuordnung” in der Tabelle 4.6. Sie dient zur Zusammenfassung der genannten Fakten über die Pins.

Pin	Art	Bezeichnung	Anschluss	Signal am Oszi	FlexRay-Zuordnung
13	–	nicht belegt	–	zyklisches	FlexRay Bus Signal: Bus-Minus
14	E/A	FlexRay Bus-Signal	Dynamische Stabilitäts-Control (DSC)	kurzer Ausschlag am Anfang	Bus-Plus
15	E/A	FlexRay Bus-Signal	DSC	kurzer Ausschlag am Anfang	Bus-Minus
16	E/A	FlexRay Bus-Signal	Elektromech. Servo-lenkung	zyklisches	Bus-Minus
19	M	Masse	Massepunkt		
31	–	nicht belegt	–	zyklisches	FlexRay Bus Signal: Bus-Plus
32	E/A	FlexRay Bus-Signal	Integrated Chassis Management (ICM)	kurzer Ausschlag am Anfang	Bus-Plus
33	E/A	FlexRay Bus-Signal	ICM	kurzer Ausschlag am Anfang	Bus-Minus
34	E/A	FlexRay Bus-Signal	Elektromech. Servo-lenkung	zyklisches	Bus-Plus
35	–	nicht belegt	–	zyklisches	FlexRay Bus Signal: Bus-Plus

Tabelle 4.6: Pinbelegungsausschnitt (FlexRay) [NewTIS.info, 2018b], [Eigene Ermittlung]

4.3.3.2 Ergebnisse aus den Versuchen der FlexRay-Pin-Kombinationen

Test 0

Erfolgreiche Datenübermittlungen waren möglich, als jeweils ein Bus-Minus mit Pin 16 und ein Bus-Plus mit Pin 35 über Kanal A des Interfaces angesteckt und die Konfiguration gestartet wurde.

Dabei erschien wie in Abbildung 4.6 zu sehen, eine empfangene FlexRay-Null-Frame-Nachricht mit der ID 70 auf Kanal A. Dieses hatte jedoch nur Nullen im Datenfeld, was bedeutet, dass der Inhalt entweder ungültig oder nicht nutzbar ist (siehe Grundlagenkapitel 2.2.4). Trotzdem war das die erste erfolgreich empfangene FlexRay-Nachricht. Dieser Fehler beim FlexRay bestätigt das Ergebnis aus ABRITES. Trotz dieses Aspekts wurden verschiedene Pins zusammen kombiniert und deren Trace-Ausgabe ausgewertet, um an

weitere Informationen zu gelangen.

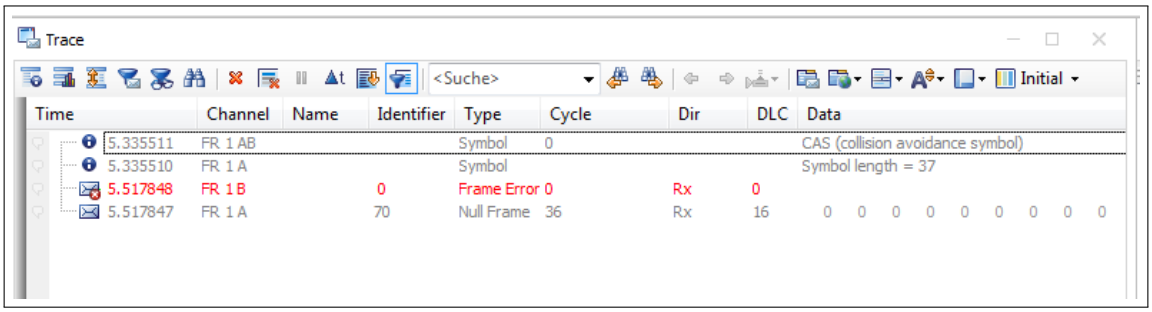


Abbildung 4.6: FlexRay-Null-Frame im Trace Fenster von CANoe [Eigene Abbildung]

Da die weiteren Tests auf Basis des Ergebnisses von Pin 16 und Pin 35 entwickelt worden sind, ist dieser als Test 0 gekennzeichnet. Außerdem wurde überprüft, ob über das neue Bus-Interface VN7610, noch die PT-CAN Signale weitergeleitet werden. Gleichzeitig wurde überprüft, ob es neue Signalmeldungen gibt, wenn der CAN-Bus zusammen mit dem FlexRay an einer Schnittstelle hängt. Der einzige Unterschied zu den bereits erkannten Signalen über das VN1610 ist, dass die IDs und der Dateninhalt in Dezimalzahlen anstelle von Hexadezimalzahlen dargestellt sind.

Test 1

Bei Test 1 sind Pin 13, 16 und 31 mit dem PT-CAN zusammen über das Bus-Interface Vn7610 überprüft worden. Da jeweils ein Bus-Minus sowie ein Bus-Plus mit verbunden waren, ist wieder das Null Frame auf Kanal A mit der ID 70 aufgetaucht. Zusätzlich ist dazu ein Error Frame 44 mit der ID 70 über Kanal B empfangen worden. Zur Veranschaulichung sind alle Ergebnisse der Tests zwei bis acht in der Tabelle 4.7 zusammengefasst.

Bei allen Ausgaben ist das Channel FR 1 AB ein Symbol gewesen, welches im Datenfeld als CAS (collision avoidance symbole) spezifiziert wurde. Außerdem ist je nach Verwendung des Kanals ein gleichnamiges Symbol aufgetaucht, welches im Datenfeld eine Länge von 37 anzeigte.

Lediglich bei den Pins 32 und 33 sind keine Signale im Trace-Fenster erschienen. Dadurch liegt die Vermutung nahe, dass es sich am Anfang um eine Art Abfrage-Signal handelt. Da es eventuell von benötigten Steuergeräten keine Antwort erhält, verfällt das Signal wieder in den Sleep-Modus. Insgesamt lassen sich trotzdem folgende Aussagen über die Ergebnisse machen.
Für den Empfang von FlexRay-Nachrichten, war definitiv jeweils eine BP- und BM-Leitung notwendig gewesen. Außerdem waren erfolgreiche Übertragungen nur über Kanal A möglich. Auf Kanal B hingegen könnte man eine Störung annehmen, weil lediglich Fehlernachrichten über diesen gesendet wurden.

Test	Pin-Kombination	Information	Null Frame	Frame Error
0	13 A und 35 B		Kanal A: ID 70	Kanal B: ID 00
1	13, 16 und 31 + PT-CAN		Kanal A: ID 70	Kanal B : ID 70, ID 0
2	13 (BM) A und 16 (BM) B + PT-CAN	nur Bus-Minus (BM)		Kanal A: ID 70; ID 00, Kanal B: ID 70; ID 00
3	31 (BP) und 35 (BP) + PT-CAN	nur Bus-Plus (BP)		Kanal A: ID 70; ID 00, Kanal B: ID 70; ID 00
4	13 (BM) und 34 (BP)	auf Kanal A	Kanal A: ID 70	
5	13 (BM) und 35 (BP)	auf Kanal A	Kanal A: ID 70	
6	16 (BM) und 34 (BP)	auf Kanal B		Kanal B: ID 70
7	16 (BM) und 35 (BP)	auf Kanal B		Kanal B: ID 70
8	13 (BM) A, 31 (BP) A, 16 (BM) B und 34 (BM) B		Kanal A: ID 70	Kanal B: ID 70

Tabelle 4.7: Ergebnisse aus den Tests der FlexRay-Pins [Eigene Ermittlung]

4.3.3.3 Kommunikationsversuch zwischen FlexRay und CAN

Wie bei der CAN-Nachrichtenabfrage im Kapitel 3.2.3.2 sollten nun alle CAN-Nachrichten an FlexRay gesendet werden. Dadurch sollte geprüft werden, ob das sicherheitskritische System Nachrichten von CAN empfängt. Problematisch beim Testen ist dabei gewesen, dass der FlexRay-Bus nach kurzer Zeit in den Sleep-Modus fällt. Visualisiert wird das, durch das Ausgrauen der Signale im Protokoll-Fenster. Um dem Einschlafen entgegenzuwirken, wurden zwei Methoden versucht:

1. ein zweites FR Null Frame über AUTOSAR Datenbank erstellen mit den Details die das andere Null Frame bereits hatte: Name NullFrame2 Channel A, ID 70, DLC 16, Base Cycle 46, Dynamic, Sender BSC und Empfänger BLU,
2. manuelles Aufwecken, in dem der Wake-up-Pin kurz abgetrennt und wieder angefügt wurde

Die Intention von der ersten Variante war, durch das Absenden weiterer Nachrichten das Einschlafen zu verhindern. Die Ergebnisse dieser Versuche sind in Tabelle 4.8 zusammenfassend dargestellt.

Test	Pin-Kombination	Information	Null Frame	Frame Error
9	13 (BM) A und 35 (BP) B	nachdem das weitere Null-Frame erstellt wurde		Kanal B: ID 70; ID 1792 Frame State: Bus Error
10	13 (BM) A und 35 (BP) B	manuelles Aufwachen		Kanal A: ID 70, ID 0, ID 1792, ID 768 Kanal B: ID 70, ID 1792, ID 0, ID 256
11	13 (BM) A, 35 (BP) B	Nachrichten-Abfrage vom K"=CAN (Pin 48 und 49) + manuelles Aufwachen		Kanal A: ID 70, ID 0, ID 1792, ID 768 Kanal B: ID 70, ID 1792, ID 0, ID 256
12	13 (BM) A, 35 (BP) B	Nachrichten-Abfrage vom PT"=CAN + manuelles Aufwachen		Kanal A: ID 70, ID 0, ID 1792, ID 768 Kanal B: ID 70, ID 1792, ID 0, ID 256

Tabelle 4.8: Ergebnisse aus den Tests der FlexRay-Pins im Zusammenhang mit CAN [Eigene Ermittlung]

Test 10

Als Beweis, dass der Versuch des manuellen Aufweckens gelingt, wird zum Beispiel bei Test 10 durch das Trace Fenster, das in Abbildung 4.7 gezeigt wird, deutlich. Eingeschlafene Signale sind in einem helleren Rot-Farbton. Die Signalbotschaften die nach dem erneuten Anstecken des Wake-Up Kabels erneut Daten schicken, sind in einem dunkleren Rot dargestellt. Neben der Schriftfarbe ist die aufgezeichnete Zeit ein weiterer Hinweis, dass das Aufwecken funktioniert.

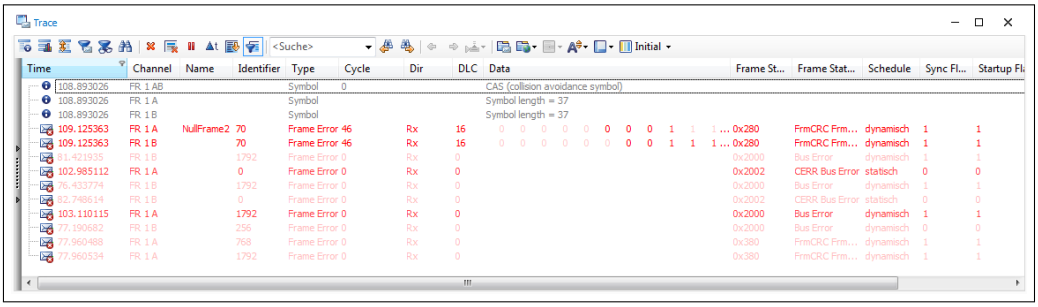


Abbildung 4.7: Trace-Ausgabe im CANoe nach manuellen Aufwecken [Eigene Abbildung]

Außerdem ist ersichtlich, dass das sonst vorhandene NullFrame aus den vorhergehenden FlexRay-Tests, nicht mehr angezeigt wird. Stattdessen ist nur noch das neu generierte

NullFrame2 dargestellt, welches das Problem des Einschlafens verhindern sollte. Dabei könnte es eventuell das alte NullFrame überschrieben haben.

Synchronisationknoten

Durch die Verbindung der drei Steuergeräte an der Bus-Plus und Bus-Minus Leitung des Interfaces ergaben sich neue Fehlermeldungen im Trace-Fenster. Diese waren zum einen dieselben, wie schon bei den Tests mit den verschiedenen Pin-Kombinationen zum anderen ergaben sich neue. Deren Ursprung oder Bedetung konnte aufgrund fehlender öffentlicher Dokumentation nicht geklärt werden. Jedoch kann man die Aussage treffen, dass es Einfluss auf die Signale hatte, die anderen Steuergeräte mit zu verknüpfen. Zudem ist das FlexRay-System auch mit den Synchronisationsknoten schnell in den Sleep-Modus gefallen und konnte keine CAN-Nachrichten empfangen.

4.3.4 Fehleranalyse

Extended CAN

Neben den 11 Bit langen CAN-IDs existieren noch die 29 Bit großen Extended-CAN-IDs. Prinzipiell könnte unter denen auch eine zugelassene IDs vorkommen, die FlexRay empfangen könnte. Dafür hatte man mit Hilfe des Interaktiven Generators zufällige Extended-CAN-Identifier generiert und abgeschickt. Diese wurden nachweislich abgesendet, jedoch nicht als empfangen gekennzeichnet.
Der Versuch, vom CAN alle 2048 möglichen Nachrichten abzufragen, benötigte das Testsystem etwa 30 Minuten. Beim Test mit Extended-CAN-IDs würden nun 536 870 912 Nachrichten abgefragt werden müssen, was approximiert knapp 15 Jahre benötigen würde und damit eindeutig das Spektrum der Praktikumszeit überschreitet. Daher ist dieser Versuch nur mit einigen ausgewählten IDs durchgeführt worden, die aber keine nützlichen Informationen lieferten. Deswegen wurde darauf verzichtet, hierbei weiter ins Detail zu gehen.

Sleep-Modus

Im Gegensatz vom D-CAN zum PT-oder K-CAN registrierte FlexRay keine ankommenden IDs vom Diagnose-CAN. Das ist zum einen positiv, weil nicht so einfach Nachrichten dieses Bussystem ansprechen können. Zum anderen muss man jedoch beachten, dass der FlexRay-Bus sehr schnell in den Sleep Modus gefallen ist. Dieser konnte lediglich manuell aufgeweckt werden. Dadurch könnten eventuell CAN-IDs in dem kurzen Zeitraum des Einschlaf-und Aufwachprozesses nicht vom System registriert werden oder es lag an den Problemen die das Diagnosegerät ABRITES analysiert hat.

Fehlende Freischaltung

Aufgrund weitergehender Recherche ist bekannt, dass es im Datenbereich einer speziellen Nachricht eine Art Freischaltung enthält, die zulässt, dass Botschaften weitergeleitet werden. In welcher Nachricht diese versteckt ist, beziehungsweise, um welche Kennung es sich dabei handelt, ist jedoch unbekannt und obliegt dem Firmengeheimnis der einzelnen Unternehmen. Eine Abfrage aller möglichen Dateninhalte von jeder CAN-ID hätte den Zeitrahmen der praktischen Arbeit überschritten. Aufgrund der maximalen Größe des Datenfelds von 64 Byte.
Deswegen war ein anderer Versuch, an interne Daten heranzukommen, das Innere des Gateway näher zu analysieren. Ziel sollte es sein, die verbauten Chips auf der Platine auszulesen und so an die Firmware zu gelangen, um neue Informationen bezüglich FlexRay, beziehungsweise dieser Freischaltung zu erhalten.

4.4 Ergebnisse des Reverse Engineering

4.4.1 Ermittelte Mikrochips

Die Abbildung 4.8 wird im Folgenden als Vorderseite deklariert. Anhand der Darstellung wurden die vorhandenen Mikrochips nummeriert. In der Tabelle 4.9 sind alle relevanten Chips der Vorderseite genannt, inklusive ihrer Artikelbezeichnung und Produktart, die bereits im Methodenteil thematisiert wurden. Die Bezeichnungen aller anderen Chips sind im Anhang A hinterlegt.

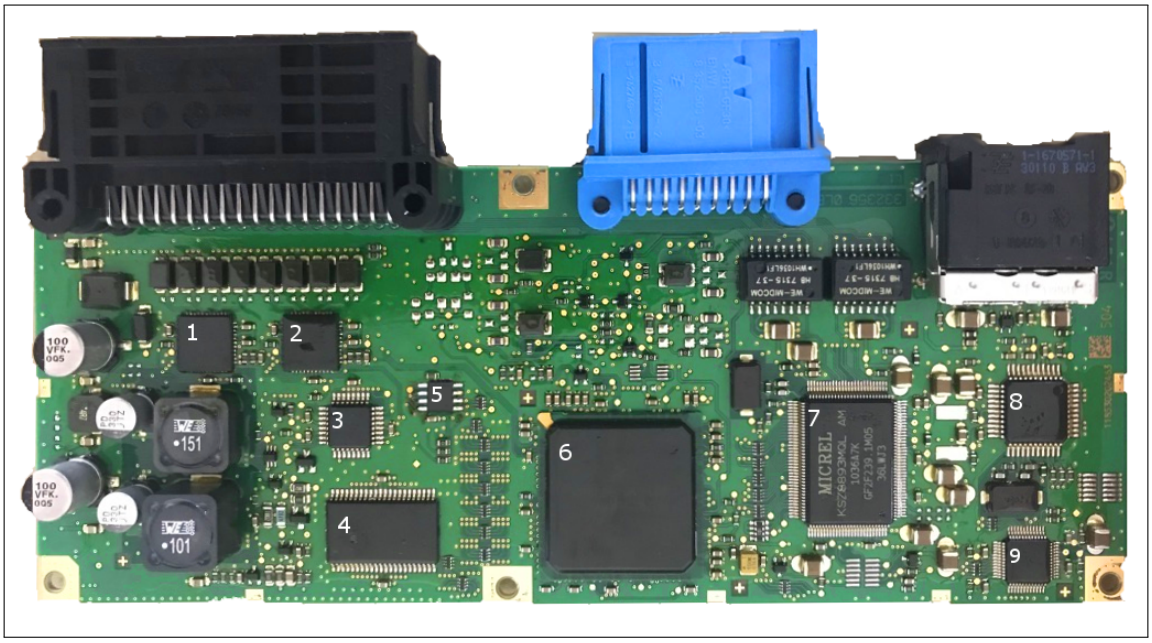


Abbildung 4.8: Vorderseite der Platine mit nummerierten Chips [Eigene Abbildung]

Nummer	Bezeichnung	Information
3	ATMEL MEGA48 15AT1 1035 G6711	8-bit Mikrocontroller - MCU [Atmel, 2016]
5	25LC256E SN e3 1025 9H4	Serial EEPROM [Microchip technology Inc., 2017]
6	MPC5567 MVR132 QMM1038 TGGUQ	CPU/Mikrocontroller [Freescale Inc., 2012]

Tabelle 4.9: Mikrcontroller-Bezeichnungen der relvanten Chips [Eigene Ermittlung]

Demzufolge ist Abbildung 4.9 als Rückseite bezeichnet worden. Zum Nachlesen der Bezeichnungen und Funktionen dient die Tabelle A.4, die ebenfalls im Anhang A hinterlegt ist.

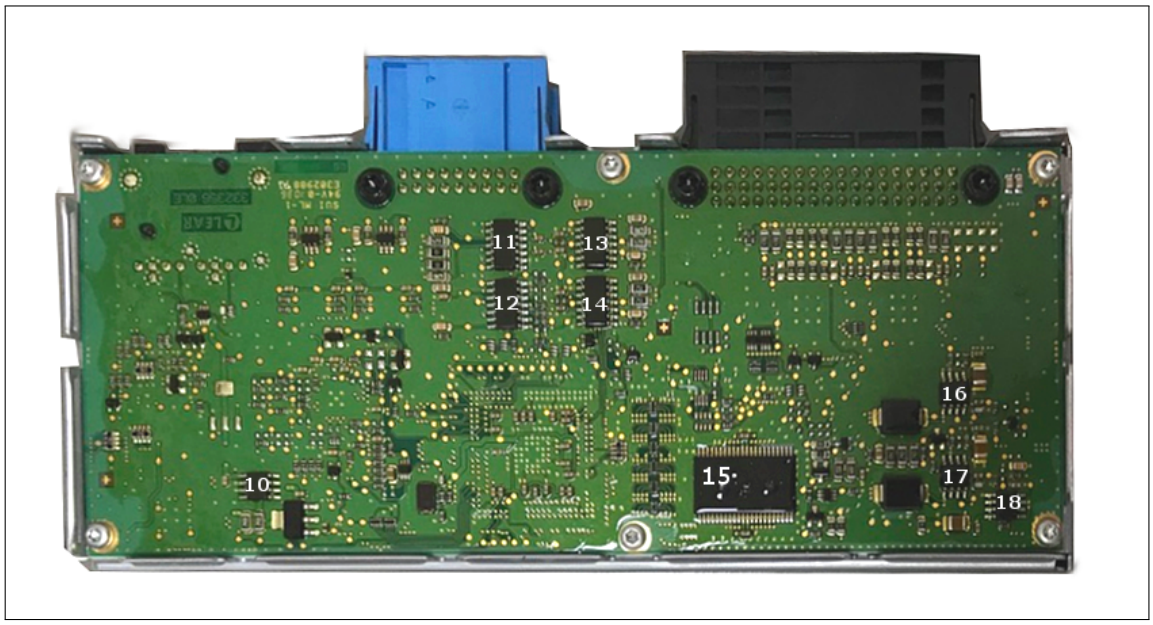


Abbildung 4.9: Rückseite der Platine mit nummerierten Chips [Eigene Abbildung]

4.4.2 Informationen aus den Mikrochips

EEPROM

Die ausgelesenen Dateien wurden mithilfe eines Hexeditors ausgewertet. Im Datenbereich 0x94 - 0xCD des EEPROMs ist die VIN im ASCII Bereich, sowie die Gerätenummer im Datenbereich von 0x4D0 - 0x5DB abgelegt. Für die restlichen Daten war keine Entschlüsselungsmatrix vorhanden. Den Inhalt komplett zu reverse-engineeren, wäre sehr aufwendig und könnte als Aufgabe in einer eigenständigen Bachelorarbeit realisiert werden.

Mikrocontroller

Aus dem ATmega48 konnten erfolgreich die Daten aus dem Flashspeicher und dem EEPROM extrahiert werden. Um die erhaltenen Informationen erfolgreich interpretieren zu können, musste man das Verfahren des Reverse Engineering anwenden. Die Vorgehensweise wurde dazu schon im Methodenteil erläutert. Daher folgt anhand des Ausschnittes in Abbildung 4.10 der asm-Datei ein Beispiel, wie man ein Kommando interpretiert.

```
Flash.hex:     file format ihex

Disassembly of section .sec1:

00000000 <.sec1>:
       0:       90 c4           rjmp    .+2336          ;  0x922
       2:       54 c0           rjmp    .+168           ;  0xac
       4:       6a c0           rjmp    .+212           ;  0xda
       6:       18 95           reti
       8:       18 95           reti
```

Abbildung 4.10: Ausschnitt aus der Datei Flash.asm [Eigene Abbildung]

Beispiel:
In Abbildung 4.10 beginnt das Hauptprogramm. Es ist das Dissambly der Firmware zu sehen, wie man erkennt ist am Anfang eine Sprunganweisung (rjmp) zur Adresse 0x922 zu sehen. Demzufolge wird zur Adresse 0x922 geleitet, (in Abbildung 4.11) wo Folgendes steht:

```
02 e6        ldi        r16, 0x62        ; 98
```

Das “ldi” bedeutet laut Datenblatt das Laden einer Konstante. Somit heißt der Ausdruck soviel das Kommando 02 e6 lädt eine Konstante namens 0x62 in das Register r16. In der darauffolgenden Zeile mit der Adresse 0x924 ist wieder das Register r16 vorhanden. Nur wird etwas mit dem Befehl “out” ausgegeben. Zusätzlich hat die Recherche im Datenblatt des Controllers ergeben, dass an Port 0x3D der Stack Pointer Register Low Byte (SPL) initialisiert wird. [Atmel, 2016]

```
922:       02 e6           ldi     r16, 0x62       ; 98  //Laden der Konstante 0x62 in Register r16
924:       0d bf           out     0x3d, r16       ; 61  //Ausgabe des Registers r16 am Port 0x3e
926:       02 e0           ldi     r16, 0x02       ; 2   //Laden der Konstante 0x02 in Register r16
928:       0e bf           out     0x3e, r16       ; 62  //Ausgabe des Registers r16 am Port 0x3e
```

Abbildung 4.11: Ausschnitt der Datei Flash.asm Teil 2 [Eigene Abbildung]

An Adresse 0x926 wird erneut eine Konstante (0x02) in Register r16 geladen und in Adresse 0x928 auf Port 0x3e im Register r16 ausgegeben. Der genannte Port entspricht im Datenblatt des ATmega48 dem Stack Pointer Register High Byte (SPH). [Atmel, 2016]

Der Stack Pointer setzt sich aus SPL und SPH zusammen. Daher steht der Stack Pointer nach dieser Befehlssequenz am RAM-Ende.

Prozessor

Wie bereits im Methodenteil erläutert wurde, waren die relevanten Testpunkte des MPC5567 nicht so plausibel, wie beim ATmega zu ermitteln. Daher wurden alle Pins, die auch in der Umgebung der Programmierpins lagen mit kontaktiert, um das wahrscheinlichste Spektrum abzudecken.

Die Prüfung der Pins mit dem Arduino ergab allerdings abweichende Ergebnisse, in Bezug auf die visuelle Beurteilung. In Abbildung 4.12 ist mit schwarzer Schrift die optische Zuordnung der Testpunkte und in roter Schriftfarbe die, des Arduinos ermittelten.

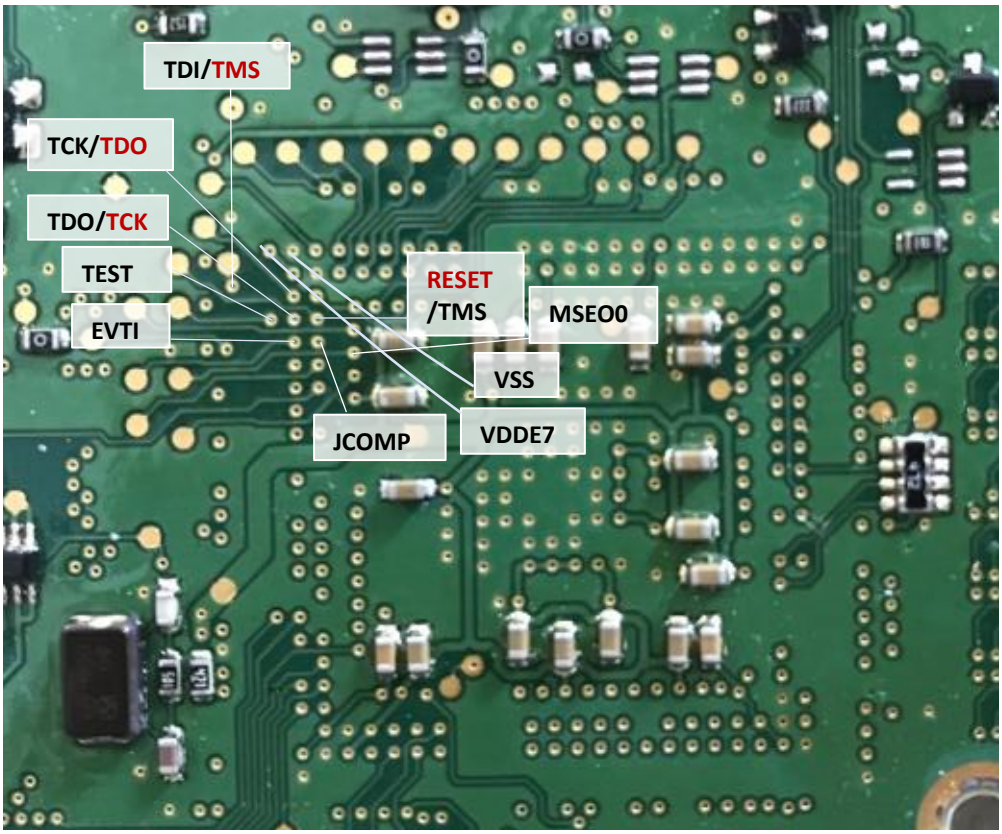


Abbildung 4.12: Zuordnung der Testpunkte [Eigene Abbildung]

Nach der Inbetriebnahme des XPROGs hatten die Status-LEDs zu keiner Zeit aufgeleuchtet, was zumindest eine Stromaufnahme signalisiert hätte. Neben der Verbindung zum USB-Port hatte man vermutet, dass noch ein Netzteil fehlen würde. Daraufhin wurde ein Passendes mit verknüpft, was auch keine Veränderung bewirkte.

Trotzdem wurde das Auslesegerät mithilfe der optischen Zuordnungen, als auch mit der ermittelten Zuordnung vom Arduino mit den Pins verknüpft. Jedoch ist bei keiner Variante irgendein Auslesen möglich gewesen. Stattdessen erschien bei jedem Test nur die Fehlermeldung, dass das Gerät still sei, siehe 4.13. Ob es sich dabei um das Auslesegerät oder den auszulesenden Prozessor handelt, konnte nicht geklärt werden.

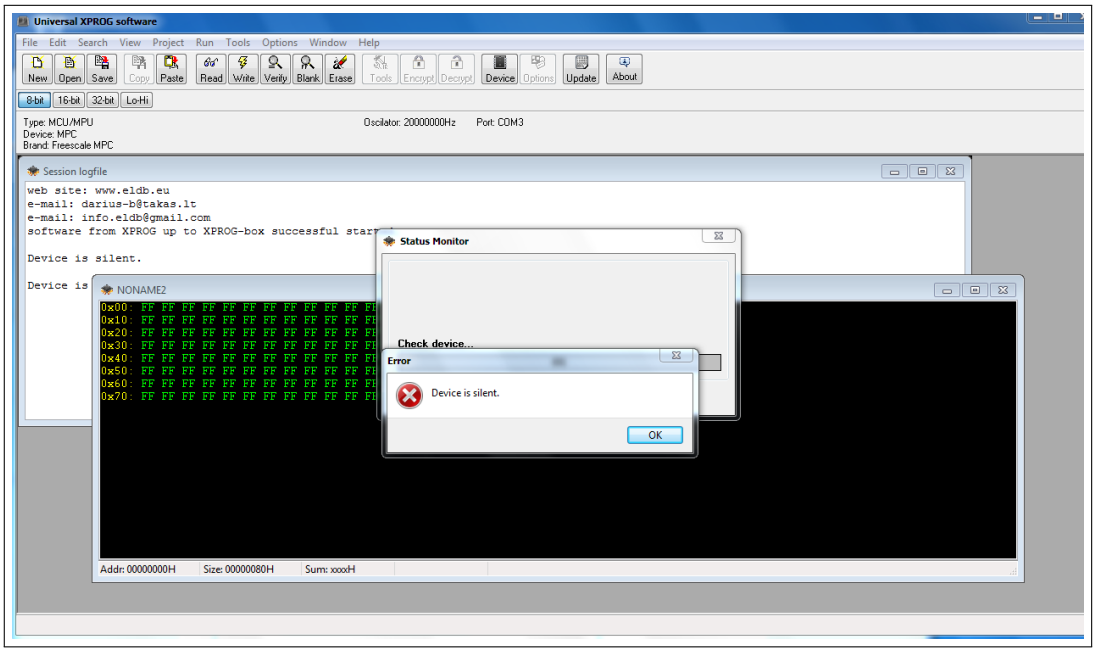


Abbildung 4.13: Ausgabe der XPROG-Software [Eigene Abbildung]

5 Diskussion

In diesem Kapitel wird deutlich gemacht, was bisher erreicht worden ist und was die Beweggründe dieser Thematik und der Arbeitsschritte sind. Für die bessere Auseinandersetzung der Problematik werden die entstandenen Ergebnisse mit anderen wissenschaftlichen Papern verglichen. Mithilfe dieses Vergleichs können Resultate in der Gesamtleistung erbracht werden.

Der Fokus dieser Arbeit lag darauf, das Bussystem FlexRay, was speziell für sicherheitsrelevante Prozesse im Fahrzeug verantwortlich ist, zu analysieren. Dafür hat man den Bus zuerst nach seinen theoretischen Grundlagen betrachtet und die Funktionsweise erörtert. Darauf folgte die praktische Analyse mithilfe eines zentralen Gateway-Moduls in Form von umfangreichen Versuchen und Testreihen. Im Vorfeld musste dazu geklärt werden, ob die Funktion des Gateways noch vorhanden ist. Der Zweck dieses Moduls ist es, Nachrichten von einem Bussystem in ein anderes zu übersetzen. Deswegen wurde die Übertragung von CAN-Nachrichten zwischen verschiedenen Controller-Area-Network-Bussen getestet. Dabei hat sich herausgestellt, dass man spezifische Anfragen über CAN-IDs schicken konnte und der angefragte Kommunikationsteilnehmer darauf eine Antwort weiterleiten kann.

Weiterhin hat sich herausgestellt, dass FlexRay definitiv auf dem Gateway vorhanden ist. Dafür wurden alle Pins an dem Stecker des Gateways überprüft und die vorhandenen Signale am Oszilloskop analysiert. Die Kommunikation von CAN zu FlexRay hatte sich jedoch als schwierig erwiesen. Beim Verschicken aller CAN-Nachrichten auf den FlexRay-Bus wurde keine Botschaften von FlexRay erfasst. Die Ursache dafür könnte unterschiedliche Gründe haben. Zum Beispiel könnten die Fehler, die das Diagnosegerät ermittelt hat, die Kommunikation behindern oder die benötigten Zeichen im Datenfeld fehlen, um ein Weiterleiten der Nachrichten zu gewährleisten.

Daher ist die Recherche in Bezug auf die verarbeiteten Mikrochips auf der Platine durchgeführt wurden, wobei man unterschiedliche Methoden zum Extrahieren der Daten angewendet hat.
Um den EEPROM auszulesen, musste dieser von der Hauptplatine gelöst werden. Die Firmware des Mikrocontrollers war ebenfalls erfolgreich mit einem Programmier- und Debugging-Interface zu extrahieren. Die Inhalte der Mikrochips waren nicht verschlüsselt, was eine Schwachstelle und gleichzeitig noch Potenzial für weitere Forschungen darstellt. Besonders relevant für die Kommunikation von Bussystemen ist jedoch der Prozessor. Dieser war mit den vorhandenen Mitteln nicht auslesbar gewesen und benötigt weitere Analysen in dieser Thematik.

5.1 Begründung dieser Thematik

FlexRay wird aufgrund seines Aufbaus und seiner Funktionsweise explizit für sicherheitsrelevante Module, wie die Bremsen und die Lenkung im Automobil eingesetzt.
Ein weiteres Anwendungsgebiet ist zum Beispiel die *X*-by-Wire Umgebung, bei der die mechanischen Signale durch elektronische ersetzt werden. Bei solchen Prozessen ist es von sehr hoher Relevanz, dass die Systeme echtzeitfähig und ausfallsicher sind. Beispielsweise hätte ein Ausfall der Signale für die Lenkung die Konsequenz, dass die Manövrierbarkeit des Fahrzeugs beeinflusst werden könnte. Demzufolge wäre auch die Sicherheit des Fahrers und aller anderen Verkehrsteilnehmer stark gefährdet.
Manipulationen in Automobilen über den bekannten CAN- oder den MOST-Bus sind wissenschaftlich dargelegt. [Tencent Keen Security Lab, 2018] Die Hürden dafür sind zum Teil nicht allzu anspruchsvoll, könnten dafür jedoch umso mehr Schaden anrichten.

Die bisherigen Forschungen haben größtenteils nur Bezug zur Sicherheit der Zuverlässigkeit des Systems (im englischen **Safety**). Über die Absicherung vor Problemen auf der IT-Ebene, welche auch im englischen als **Security** bezeichnet wird, ist bisher nur wenig in Erfahrung gebracht worden. [Nillson et al., 2009] Deswegen spezialisiert sich die Arbeit insbesondere auf die Security des Bussystems.

Bezüglich der Analyse mit FlexRay gibt es nur wenig vergleichbare Thematiken, die als Leitfaden hätten dienen können. Deswegen wurden viele Schritte nacheinander abgearbeitet, deren Ergebnisse evaluiert und daraus neue Versuche entwickelt, die zur Erreichung des Ziels dienten.
Zum Beispiel wurde zuerst die Simulation mit CANoe erstellt, um sich mit den theoretischen Grundlagen von FlexRay auseinanderzusetzen. Gleichzeitig diente die Software als Simulationsumgebung für die Tests an den verschiedenen Bussystemen. Danach war das Auslesen mithilfe des Diagnosegerätes genutzt worden, um einen Überblick über das Funktionsspektrum des zur Verfügung gestellten ZGWs zu erhalten.

Zur Sicherstellung der Arbeitsweise des Gateways sind verschiedene Tests erst mit den etablierten CAN-Bussen gemacht worden, da es bezüglich des Controller Area Network bereits mehrere Analysen und Recherchen gibt. Sowohl mit den positiven als auch negativen Resultaten konnten neue Erkenntnisse erlangt und weitere Tests mit FlexRay durchgeführt werden. Dafür hatte man zum einen Versuche an den einzelnen Pins und zum anderen viele unterschiedliche Tests mit Pin-Kombinationen gemacht.

Weiterhin sind viele verschiedene Wege angegangen worden, um eine Kommunikation mit dem Gateway aufzubauen und zu testen, wie das Bussystem angesprochen werden kann. Falls das FlexRay-System, wie der CAN-Bus, ohne Hindernisse hätte angesprochen werden können, wäre durch den veränderbaren Dateninhalt einer Nachricht ein Angriff kein Problem mehr. Die daraus resultierenden Folgen könnten ausschlaggebend für die Sicherheit der Verkehrsteilnehmer sein können.

Mit den zur Verfügung gestellten Mitteln war eine weitere Analyse der Kommunikation von außen nicht mehr machbar gewesen.
Deswegen ist man zur Analyse der Platine übergegangen, um Erkenntnisse aus der Firmware von den zur Steuerung eingesetzten Mikrochips und Speicherelementen zu erlangen.

Ausbaumöglichkeiten

Für die Car Forensic Forschungsgruppe, in der die Arbeit entwickelt wurde, ist FlexRay ebenfalls ein neues Themengebiet. Somit ist es schwierig zu sagen, was man besser gemacht haben könnte, da es keine bekannten vergleichbaren Arbeiten in dieser Form gibt. Ebenfalls problematisch ist bei dem Themengebiet, dass viele Details dem Firmengeheimnis obliegen und somit nicht öffentlich zugänglich sind. Auch auf Anfrage hin wollte BMW keine Informationen zu dieser Thematik herausgeben.

Im Nachhinein betrachtet, sollte das Ethernet und der MOST-Bus intensiver mit in die Recherche einbezogen werden. Dafür waren auch Pins am Stecker vorgesehen, die nicht mit getestet wurden. Zudem besteht noch die Möglichkeit, beim Testen der FlexRay-Kommunikation, den Code zu erweitern beziehungsweise weitere Nachrichten zu generieren, um das FlexRay vom Einschlafen abzuhalten.

Außerdem ist es sinnvoller, bereits zu Anfang neben den theoretischen Grundlagen, auch Beispiele anderer Arbeiten mit ähnlichen Schwerpunkten sich intensiver anzueignen. Das Wissen nach dem Vergleich mit anderen wissenschaftlichen Arbeiten hätte vor der Entwicklung der verschiedenen Versuche geholfen, aber ermöglicht zugleich noch Potenzial für weitere Forschungen.

5.2 Vergleich zu anderen wissenschaftlichen Arbeiten

In diesen Kapiteln wird die Thematik dieser Ausarbeitung mit anderen wissenschaftlichen Lektüren in Relation gesetzt. Dafür wurden Arbeiten herangezogen, die sich mit einzelnen Schwerpunkten dieser Bachelorarbeit ebenfalls auseinandergesetzt haben. Dabei ist die Reihenfolge der ausgewählten Ausarbeitungen der Abfolge in dieser Arbeit angeglichen.

Daher folgt zuerst etwas bezüglich der Simulationsumgebung CANoe aus den Grundlagen, mit der das Verständnis für die notwendige Theorie dieser Arbeit vertieft worden ist. Danach sind vergleichende Aspekte mit den Methoden chronologisch betrachtet worden. Schwerpunkte liegen dabei auf der Diagnosesoftware, einem Angriff ins FlexRay-Netz und dem Aufdecken von Schwachstellen bei BMW-Fahrzeugen mithilfe von Reverse Engineering.

5.2.1 Alternative zu CANoe

In dieser Arbeit ist die Software CANoe von Vector zum Einsatz gekommen. Damit man mit diesem Tool arbeiten kann, musste im Vorfeld eine hohe Lizenzgebühr bezahlt werden. Stefan Buschmann stellt in seiner Ausarbeitung eine Alternative dar, um ein FlexRay-Netzwerk zu simulieren. Dafür verwendet er das Objective Modular Network Testbad in C++ (OMNeT++). Dabei handelt es sich ebenfalls um eine eventbasierte Simulationsumgebung, die hauptsächlich zur Nachstellung von Kommunikationsnetzwerken genutzt wird. [Buschmann, 2012]

Im Vergleich zu CANoe ist es ebenfalls möglich, notwendige Komponenten zu generieren und mit der NED-Sprache (bei CANoe CAPL-Sprache) Funktionen zu übermitteln. Zusätzlich ist die private Nutzung des Tools kostenlos.[Buschmann, 2012] Vector Informatics bietet zwar ebenfalls eine kostenlose Testversion an, die jedoch nicht für die Simulation von FlexRay-Netzwerken geeignet ist.

Prinzipiell war es für Buschmann möglich gewesen, über OMNeT++ ein FlexRay-Netzwerk zu erstellen, mit dem Ziel es mit anderen Bussystemen zu vergleichen. Der Aufwand war jedoch sehr hoch. Im Vergleich dazu ist in CANoe ein FlexRay-Netzwerk in wenigen Schritten aufgebaut, ohne jede Eigenschaft des Bussystems programmieren zu müssen. Zudem hat es eine viel bessere grafische Oberfläche und visuelle Erweiterungen zur Evaluierung. Trotzdem kann das OMNeT++ für die Einarbeitung und Auseinandersetzung des Aufbaus eines Bussystems hilfreich sein. [Buschmann, 2012]

5.2.2 Diagnosesoftware VCDS

Das aktuelle Paper von Le-Khac et al. beschäftigt sich ebenfalls mit der Fahrzeug-Forensik. Das Ziel der Autoren war gewesen, herauszubekommen wie viel Daten in einem Fahrzeug abgespeichert werden und welche Informationen davon als Beweise bei einer Straftat dienen könnten. Für die Versuche benutzten die Forscher ein Infotainment-System aus einem Volkswagen. Außerdem diskutieren sie, welche Hardware oder Software genutzt werden kann, um dem Fahrzeug forensische Informationen zu entnehmen.

Im Vergleich mit dieser Arbeit haben sich die Forscher nicht mit FlexRay, sondern intensiver mit MOST (siehe Kapitel 2.1.3) aus einem VW auseinandergesetzt. Eine Gemeinsamkeit ihrer Ausarbeitung ist jedoch, dass sie in ihrer Fallstudie ebenfalls damit beginnen, das Gateway auszulesen, um mehr Informationen zu erhalten.

Weiterhin wurde dafür auch ein spezielles Auslesegerät verwendet. [Le-Khac et al., 2018] Der Unterschied liegt hierbei an dem Produkt selbst. Die Software, welche für diese Bachelorarbeit zur Verfügung steht, ist ABRITES gewesen, welches herstellerübergreifend genutzt werden kann. Das Forscherteam hingegen hat das VAG-COM Diagnose-System

(VCDS) [4] von RossTech genutzt. Das Produkt ist Windows-basiert und etabliert für das Auslesen von Fahrzeugdiagnosen. Außerdem braucht es zum Anwenden der Software einen Adapter namens HEX-NET derselben Firma, siehe Abbildung 5.1. [Le-Khac et al., 2018]

Abbildung 5.1: VCDS von RossTech [Vetter, 2018]

Beide Diagnosegeräte geben ähnliche Aussagen, zum Beispiel über gefundene Module im Fahrzeug, die VIN und den Kilometerstand aus. Daher wäre es hierbei interessant gewesen, die Ergebnisse des VCDS mit denen vom ABRITES abzugleichen, ob diese am Gateway dieselben Informationen wiedergeben. Das Produkt von Ross-Tech ist jedoch hauptsächlich auf die Fahrzeuge der VW-Audi Gruppe spezialisiert. Gemäß Vetter soll es auch schon zusätzliche Interfaces für markenübergreifende Automobile (unter anderem Mercedes Benz, BMW) geben. [Vetter, 2018]

Insgesamt zeigt das Paper eine ähnliche Vorgehensweise zum Analysieren eines Bussystems. Die Wissenschaftler wollten zwar an das Entertainment-System von einem VW, sind aber ähnliche Schritte gegangen, wie in dieser Recherche für die Bachelorarbeit. Es wurde zuerst das für den Kommunikationsaustausch notwendige Gateway über die OBD-II-Buchse ausgelesen. Danach wollten sie an die Daten über die verbauten Mikrochips aus dem Infotainment-System gelangen. Dazu hatten die Autoren ebenfalls zwei Methoden, entweder ein JTAG verwenden oder den Mikrochip ablösen. Ihr Vorteil war jedoch gewesen, erfolgreich alle Daten aus den Chips extrahieren zu können. Aus denen konnten Le-Khac et.al. einige interessante Informationen entnehmen. Jedoch stießen sie dabei auf dieselbe Problematik wie in dieser Arbeit, dass die meisten Nachrichten-Bedeutungen nur den jeweiligen Herstellen bekannt sind. [Le-Khac et al., 2018] Prinzipiell zeigt der Vergleich, dass die Vorgehensweise zur Analyse eines Bussystems, in dieser Arbeit FlexRay, denen der renommierten Forscher entsprochen hatte.

[4] Volkswagen-Audi-Gemeinschaft (VAG)

5.2.3 Wireless Gateway

Eine Arbeit, die sich spezifisch mit der Sicherheit im Sinne von dem englischem Security des FlexRay-Protokolls beschäftigt hat, ist das Paper von Nillson et al. Dabei wurde zuerst das Bussystem, sowie weitere Technologien der Automobilindustrie theoretisch betrachtet. Besonders alarmierend ist für sie der aufkommende Trend Firmware-Updates kabellos, also über das Protokoll Firmware Over-the-Air (FOTA) auf das Automobil spielen zu können. Damit spart der Fahrzeughalter sich zwar den Weg zur Werkstatt, jedoch ist die drahtlose Kommunikation auch ein Eintrittstor für potenzielle Straftaten. [Nillson et al., 2009]

Bei der Betrachtung von FlexRay ist man speziell auf Aspekte der Daten: Vertraulichkeit, Integrität, Verfügbarkeit, Authentizität und Aktualität eingegangen. In Deutschland sind diese Kriterien auch als Schutzziele vom Bundesamt für Sicherheit in der Informationstechnik (BSI) bekannt. Ergebnis der Untersuchung ist, dass das Protokoll lediglich mit dem Cyclic-Redundancy-Check (CRC) in den Nachrichten die Datenintegrität schützt und das Kommunikationsschema die Datenverfügbarkeit bewahrt. Diese Faktoren betreffen aber nur die Safety und nicht die Security von FlexRay. Bezüglich der Vertraulichkeit, Authentifizierung und Aktualität gibt die Spezifikation keine Aussagen her. Daraufhin haben die Autoren FlexRay bezüglich der Security evaluiert und das FlexRay-Netzwerk mithilfe von CANoe über ein Wireless Gateway angegriffen. [Nillson et al., 2009]

Gemeinsamkeit des Papers zu dieser Ausarbeitung war, dass deren Grundlage ebenfalls auf einem FlexRay-Netzwerk beruht, welches sie mithilfe des Softwaretools CANoe angegriffen haben. Der größte Unterschied ist jedoch gewesen, dass die Forscher für die Analyse ein **Wireless Gateway** anstelle eines normalen zentralen Gateway-Moduls genutzt haben (siehe Abbildung 5.2).

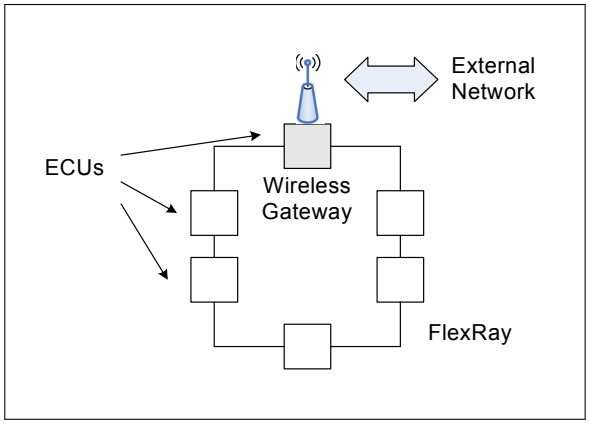


Abbildung 5.2: FlexRay-Netzwerk mit ECUs und Wireless Gateway [Nillson et al., 2009]

Ein weiterer Gegensatz zu den Arbeiten am ZGM von BMW ist, dass Nillson et.al explizit Zugang zu einem **ganzen** Netzwerk hatten. Dadurch waren sie befähigt, den Datenaustausch zu: lesen, täuschen, wegzulassen, stehlen und zu wiederholen. Demzufolge

konnten sie direkt ins Netzwerk Nachrichten senden. Diese Botschaften hatten dieselben Eigenschaften, wie bei der FlexRay-Anleitung in diesem Anhang A. In deren Fall war das Ziel, die Bremslichter einzuschalten, ohne die Bremse zu betätigen, was auch funktionierte. [Nillson et al., 2009]

Das Resultat des Papers ist, dass das FlexRay-Protokoll einfach angegriffen werden kann und es nicht ausreichend Schutz, bezüglich der Sicherheitsaspekte gibt. Außerdem sind die Resultate der Autoren mit denen der Bachelorarbeit einig, dass nicht nur Anforderungen bezüglich der Zuverlässigkeit wichtig sind. Unbedingt sind auch die Aspekte zur Sicherheit des Systems zu betrachten, da die Security genauso mit der Safety zum Schutz des Fahrers beiträgt. [Nillson et al., 2009]

5.2.4 Sicherheitsaspekte eines fahrzeuginternen Netzwerks

Ebenfalls mit den Security-Aspekten eines internen Fahrzeug-Netzwerks haben sich Kleberger et al. auseinandergesetzt. Die Zielstellung der Autoren war herauszufinden, mithilfe der zu diesem Zeitpunkt vorhandenen Forschungsergebnisse festzuhalten, welche Probleme es bezüglich der Thematik gibt und was für Lösungsansätze bisher dazu empfohlen werden. [Kleberger et al., 2011] Die Voraussetzungen für die Analyse war ein “Connected Car”, welches sich zusammensetzt aus drei Domänen:

1. dem Fahrzeug mit dem internen Netzwerk und den ECUs,
2. dem Portal des Fahrzeugherstellers sowie Lieferdiensten zum Fahrzeug und
3. dem Kommunikationskanal zwischen Fahrzeug und dem Portal

Nach der Recherche der Autoren ergaben sich für sie gewisse Sicherheitsprobleme, unter anderem bezüglich der nicht ausreichenden Bus-Absicherung sowie dem Missbrauch der Busprotokolle. Zum Thema Bus-Absicherung bezogen sie sich zm Teil auf das bisher schon vorgestellte Paper von Nillson et al. ([Nillson et al., 2009]), dass es nicht ausreichend Schutz der Datenauthentifizierung, Datenvertraulichkeit sowie Datenaktualität, seitens CAN und FlexRay gibt. [Kleberger et al., 2011]

Besonders interessant für dieser Arbeit ist die Ermittlung der Zweckentfremdung der verschiedenen Busprotokolle, inklusive FlexRay gewesen. Hierzu fanden sie heraus, dass mit Hilfe von schädlichen Error Messages den Fehlererkennungsmechanismus, der im FlexRay (und CAN) implementiert ist, angegriffen werden könnte. Das Resultat wäre, dass sich der Controller vom Netzwerk trennen würde. [Kleberger et al., 2011] Über diesen Weg hatte bisher kein anderes erarbeitetes Paper berichtet, was Anlass für weitere Recherchen in diese Richtung gibt.

5.2.5 Schwachstellen bei BMW-Fahrzeugen aufgedeckt

Erst im Mai 2018 haben chinesische Wissenschaftler vom Keen Security Lab eine Studie über mehrere Schwachstellen bei BMW herausgebracht. Darin ist enthalten, wie sie Zugang durch die OBD-II-Buchse erhalten, indem sie das Ethernet (E-Net) kontaktiert haben. Das E-Net ist ebenfalls ein Netzwerk und kann zum Auslesen von Diagnosenachrichten aus einem Gateway eingesetzt werden. Mithilfe von Reverse Engineering des Protokolls, konnte ein Speicherfehler ausgenutzt werden und so Root-Rechte erlangt werden. Aus diesen Erkenntnissen konnten die Wissenschaftler auch Lücken im Bluetooth-System eines Steuergerätes entdecken und ein Abstürzen sowie Neubooten der Haupteinheit bewirken. Der Aufbau für einen Angriff über Ethernet ist in Abbildung 5.3 dargelegt. [Tencent Keen Security Lab, 2018]

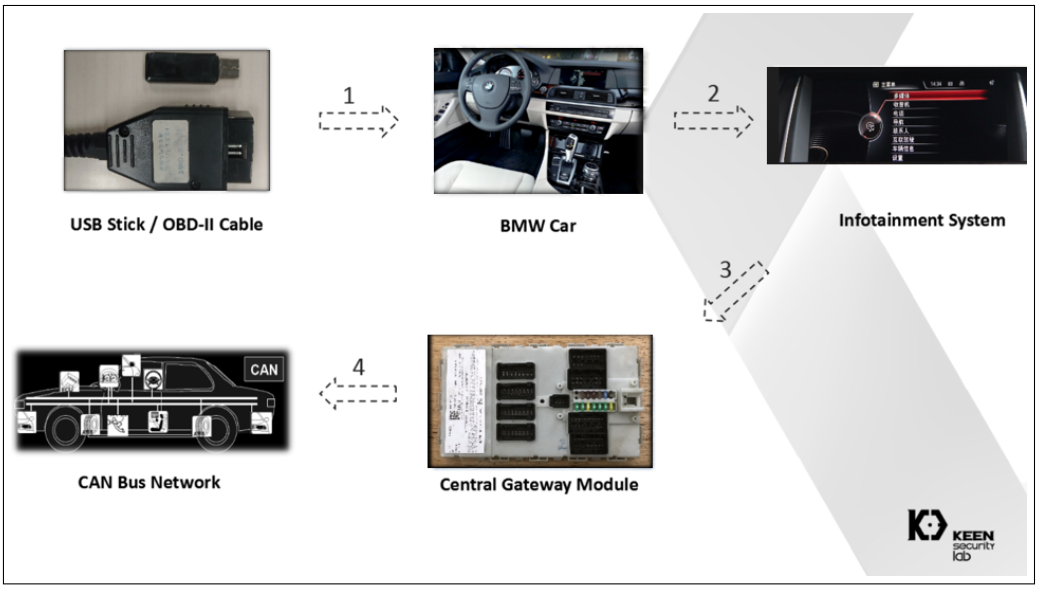


Abbildung 5.3: Angriff über OBD-II-Buchse [Tencent Keen Security Lab, 2018]

Im Vergleich zu dieser Arbeit wurde die OBD-II-Buchse ebenfalls als Eintrittsmöglichkeit ins Fahrzeugnetz betrachtet. Der Unterschied bei diesem Beispiel ist jedoch, dass anstelle des CAN-Busses, das Ethernet verwendet wurde, um einzudringen.

Eine weitere Sicherheitslücke hatte das Keen Security Lab beim MOST-Bus und der Telematic Control Unit (TCU) gefunden. Die TCU ist für Kommunikationsprozesse am Gateway verantwortlich, in die besonders der CAN- und MOST-Bus involviert sind. Mithilfe aufwendiger Verfahren konnten die Experten aus der Ferne Zugang ins Fahrzeugnetz erlangen. Eine der Methoden war der Aufbau einer Global System for Mobile Communications (GSM)-Basisstation in Form eines Imsi-Catchers, um über einen Man-in-the-Middle-Angriff [5] Befehle ans Auto zu senden. Um den Aufbau leichter nachvollziehen zu können, dient Abbildung 5.4.

[5] Angreifer steht zwischen beiden Kommunikationssegmenten

Außerdem ist es dem Team gelungen, erneut über Reverse Engineering einen Speicherfehler auszunutzen. Danach war es einfach für sie, über das Gateway Diagnosenachrichten an PT- und K-CAN zu schicken. [Tencent Keen Security Lab, 2018]

Um die detaillierte Vorgehensweise der Forscher nachzuvollziehen, kann die Quelle [Tencent Keen Security Lab, 2018] herangezogen werden.

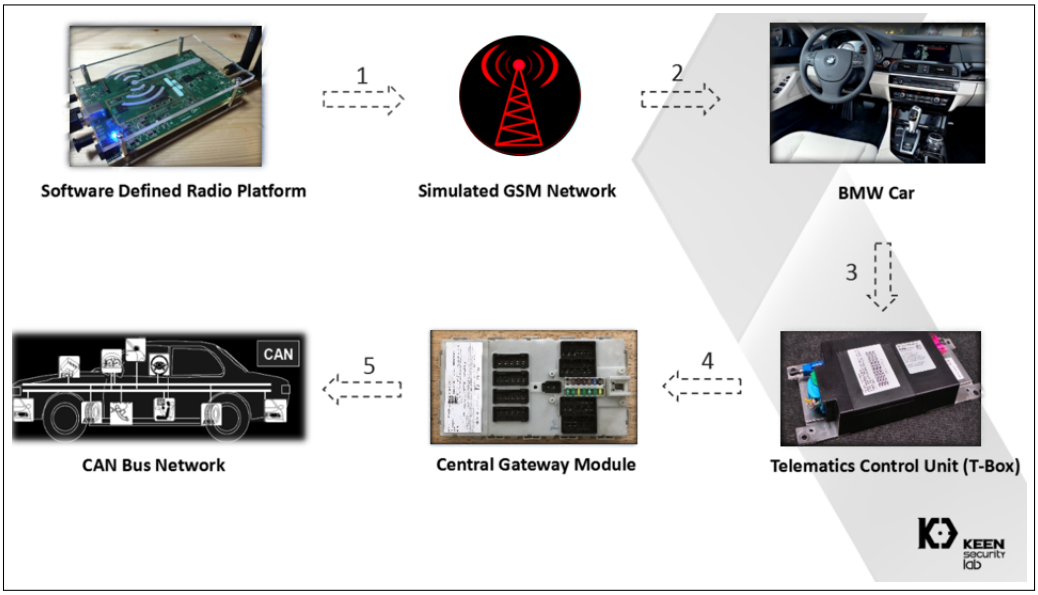


Abbildung 5.4: Angriff aus der Ferne [Tencent Keen Security Lab, 2018]

Eine Gemeinsamkeit zu dieser Sicherheitslücke ist, dass man bei der Recherche dieser Arbeit ebenfalls vom D-CAN Nachrichten an PT- und K-CAN schicken konnte (siehe Kapitel 3.2.3.2). Außerdem legen die Recherchen der chinesischen Wissenschaftler dar, wie viele Informationen mithilfe von Reverse Engeneering erlangt werden können. Ebenso wurde in dieser Bachelorarbeit diese Vorgehensweise nachgegangen, um Informationen aus der Firmware der Chips zu gelangen.

Zusammenfassend ist jedoch durch die Recherche des Wissenschaftler-Teams deutlich geworden, dass es einige Sicherheitslücken bei BMW-Fahrzeugen gibt. Allein die Zusammenfassung der Studie bietet großes Potenzial. Es wurde dahingehend in viele verschiedene Richtungen geforscht. Hierbei sind weitreichende Sicherheitslücken entdeckt worden, mit denen man sich viel intensiver beschäftigen könnte. Die Veröffentlichung des kompletten Papers soll zum Schutz für alle Verkehrsteilnehmer erst 2019 vorgenommen werden. Dadurch hat BMW die Möglichkeit, gegen die entdeckten Sicherheitsrisikos vorzugehen. Spätestens wenn das komplette Dokument veröffentlicht wird, könnte man damit noch weitere interessante Tests vornehmen. [Tencent Keen Security Lab, 2018]

5.3 Einschätzung der Arbeit

Die Betrachtung mit anderen wissenschaftlichen Arbeiten ist sehr erkenntnisreich gewesen. Dadurch konnte ein erweitertes Spektrum zu Versuchen in dieser Thematik in Erfahrung gebracht und mit denen aus dieser Bachelorarbeit verglichen werden.

Positiv in dieser Arbeit ist die Verwendung der Softwareumgebung CANoe. Diese diente als Basis für viele Tests und ermöglichte eine gute Protokollierung sowie Evaluierung der Versuche. Ebenso erfolgreich ist das Auslesen über die bekannte OBD-II-Buchse im Auto gewesen. Hierbei war es von Vorteil, ein Produkt zu besitzen, das herstellerübergreifend genutzt werden kann.

Die Autoren des Papers [Nillson et al., 2009] hatten im Vergleich zu dieser Arbeit ein komplettes Netzwerk für die Analyse zur Verfügung. Dadurch war bei ihrem Angriff auf FlexRay eine andere Voraussetzung vorhanden gewesen. Ein großer Vorteil liegt darin, dass sie die Sicherheit nach den Schutzzielen des BSI nachgegangen sind. Daran ist gut zu erkennen, welche Aspekte beachtet oder wiederum missachtet wurden. Das bietet auch die Möglichkeit, die Schutzziele bei anderen Bussystemen zu ermitteln und untereinander zu vergleichen.

Zu beachten ist, dass zum Beispiel selbst die ausgebildeten Fachleute des Tencent keen Security Lab für die Ergebnisse mehrere Monate gebraucht haben. Zudem besitzen sie die benötigten finanziellen Möglichkeiten, um in ihren Laboren einen Neuwagen, mit all seinen Komponenten zu beschaffen und auseinanderzunehmen.
Weiterhin haben sich die Wissenschaftler intensiver mit dem MOST-Bus beziehungsweise dem Ethernet-Zugang über die OBD-II-Buchse beschäftigt und somit weniger mit FlexRay. [Tencent Keen Security Lab, 2018] Trotzdem ist durch ihre Studie deutlich geworden, wie über unterschiedliche Steuergeräte der Zugang ins interne Netzwerk möglich ist. Das zeigt für nachfolgende Forschungen, dass es wichtig ist, nicht nur ein einzelnes Bussystem zu analysieren, sondern das ganze zusammenhängende System der Busse zu betrachten. Deswegen sollte man für weitere Recherchen zum Beispiel noch den MOST-Bus und das Ethernet intensiver mit einbeziehen.

Insgesamt ist in dem begrenzten Zeitraum versucht worden, jede Möglichkeit nach bestmöglichem Wissen auszuführen, um damit an neue Ergebnisse zu gelangen. So konnte in Erfahrung gebracht werden, was FlexRay besser als CAN macht, zum einen von dem theoretischen Aspekt und zum anderen auch in praktischen Versuchen. Selbst nach Aussage der Wissenschaftler aus den Papern steht die Security in Bezug auf FlexRay noch ziemlich am Anfang, was die Forschung betrifft [Nillson et al., 2009]. Daher bietet diese Bachelorarbeit eine Grundlage zur Analyse von FlexRay und stellt damit eine gute Basis für weitere Recherchen dar.

6 Fazit und Ausblick

6.1 Fazit

Die Leistung und Entwicklung der Technologie im Automobilbereich ist immens. Nur die Absicherung im Bereich der Security (des Systems) ist viel weniger erforscht, als im Bezug zur Safety. Prinzipiell hat sich herausgestellt, dass FlexRay von den theoretischen Voraussetzungen besser geeignet ist als CAN. Besonders der Aufbau und der Kommunikationszyklus von FlexRay sind deutlich nützlicher für eine sicherheitskritische Kommunikation.

Bei vielen Tests war es zum Beispiel möglich, mit diversen Nachrichten den CAN-Bus anzusprechen, wobei die Botschaften mit unterschiedlichen Inhalten versehen wurden. Anhand weiterer Versuche konnten an den Pins des Gateways FlexRay-Signale nachgewiesen werden. Ein Nachrichtenaustausch vom CAN- zum FlexRay-Bus war jedoch nicht erfolgreich.

Eine andere Alternative ist gewesen, durch Reverse Engineering an weitere Erkenntnisse zu gelangen. Dafür sind in der für die Erstellung der Bachelorarbeit vorgesehenen Zeit alle Voraussetzungen zum Extrahieren der Daten geschaffen worden. Dadurch hat sich ergeben, dass der EEPROM und der Mikrocontroller zugänglich waren und man die Daten der Mikrochips extrahieren konnte, da diese nicht verschlüsselt waren.

Zusammenfassend kann man sagen, dass es möglich ist, über die OBD-II-Buchse oder das Multimedia-Segment über den CAN- oder den MOST-Bus in das Kommunikationsnetz vorzudringen. [Tencent Keen Security Lab, 2018] Dadurch besteht die Annahme, dass durch intensives Reverse Engineering der extrahierten Dateien aus den Mikrochips auch der Zugang zum FlexRay-Bus ermittelt werden kann. Wenn dieser Schritt gelingt, ist es möglich, den Bus anzusprechen und später auch anzugreifen.

6.2 Ausblick

Das Thema FlexRay befindet sich in der Forschungsgruppe Car Forensic noch in der Entwicklungsphase, dadurch sind einige Tests noch erweiterbar. Gleichzeitig ist auch jeder Misserfolg des einen Tests eine neue Erkenntnis für das weitere Vorgehen. Durch den Umfang dieser Thematik konnten in dem gegebenen Zeitraum nicht alle Ideen, die noch weitere Informationen erbringen könnten, umgesetzt werden. Deswegen folgt nun ein Überblick über mögliche Vorgehensweisen, die noch folgen könnten.

Reverse Engineering

Besonders durch andere Forschungsarbeiten ist deutlich geworden, dass die meisten Informationen durch Reverse Engineering in Erfahrung gebracht werden konnten [Le-Khac et al., 2018], [Tencent Keen Security Lab, 2018]. Deswegen wäre eine Informationsquelle, die extrahierten Dateien aus dem EEPROM komplett auszuwerten. In Zukunft könnte man ebenfalls daran weiterarbeiten, den Prozessor auszulesen. Dieser Mikrochip ist definitiv ein Bestandteil für die Kommunikation der Busse untereinander [Kurt Veit - Mixed Mode, 2007]. Dafür könnte man neben den bereits versuchten Vorgehensweisen noch andere Methoden anwenden, wie zum Beispiel die Anschaffung eines höherwertigen Auslesegerätes oder den Chip von der Platine abzulösen und versuchen auszulesen.

Anschaffung weiterer Fahrzeugelemente

Interessant wäre es außerdem, ein zweites Gateway oder weitere ECUs [6] zur Verfügung zu haben. Voraussetzung dafür wäre, dass sichergestellt werden kann, dass die Bauelemente problemlos im Fahrzeug funktioniert haben. Damit hätte man zum einen die Möglichkeit, die Testergebnisse zu vergleichen und zum anderen auch die extrahierten Daten gegenüberzustellen und daraus Erkenntnisse zu ziehen.

Intrusion Detection System

Zur Absicherung von CAN-Bussen wurden bereits Intrusion Detection Systeme (IDS) angewendet. Diese sollen den Bus vor Attacken in Form von schädlichen Nachrichten schützen. Die Arbeitsweisen der IDS basieren auf jeweils unterschiedlichen Detektions-Methoden. Bisher sind solche Systeme nur bei CAN-Bussen implementiert worden, daher wäre es möglich, in Zukunft auch an einem IDS für das FlexRay-Protokoll zu arbeiten. [Kleberger et al., 2011] Mithilfe des Systems könnte eine Voraussetzung geschaffen werden, zu bestimmen welche Nachrichten unter welchen Bedingungen gesendet werden dürfen.

Mögliche Trennung der Bussysteme?

Die Auswertung der Paper zeigt, wie an vielerlei Stellen Zugang ins fahrzeuginterne Netzwerk erlangt werden kann. Über diese Schwachstellen ist es möglich, die Sicherheit des Fahrers zu beeinflussen. Somit wäre ein Lösungsansatz, ob man die Bussysteme für die sicherheitskritische Kommunikation wie FlexRay nicht von sicherheitsirrelevanten Systemen wie MOST trennen könnte. Dadurch wäre zumindest eine physische Trennung gegeben und ein direkter Zugang nicht ohne weiteres möglich. Eine andere Alternative wäre, über das Gateway die Verbindung nur aus der Richtung des sicherheitskritischen Bussystems zuzulassen und nicht umgekehrt.

[6] neben den durch CANoe bereits simulierten Modulen

Anhang A: Anhang

Anleitung zur Erstellung einer FlexRay-Simulation in CANoe

Aufgabe: Über ein FlexRay-Bussystem sollen **2 Steuergeräte** miteinander kommunizieren. Zur Veranschaulichung der Funktionsweise sollen nach dem Starten der Simulation durch das Betätigen des Bremspedals die Bremslichter angehen.

1. **Projekt anlegen:**

   a) Starten des Tools CANoe 10.0 SP2.
   b) Neues Projekt anlegen mit “Datei” –> “Neu” –> “FlexRay”
   c) Unter “Hardware” –> “Verwendete Kanäle” –> “CANoe Optionen Allgemein” das Projekt auf “Simulierter Bus” umstellen.
   d) Speichern der Konfiguration in einem geeigneten Ordner.

2. **Simulationsaufbau:**
   2 Knoten einfügen:

   a) Rechtsklick auf Knoten –> “Netzknoten einfügen” –> ECU 1 + ECU 2 wird erstellt.
   b) Bei “Interaktiven Generatoren” über Rechtsklick ein PDU Panel hinzufügen.

3. **Erstellung einer Datenbank:**

   a) “Werkzeuge –> AUTOSAR EXPLORER –> File –> New File –> FlexRay (Channel A) –> OK” –> abspeichern“*dateiname*.arxml”
   b) Optional: “New_FlexrayCluster” umbennen bei ShortName manuelle Eingabe möglich (im Bsp.: Bussystem)
   c) “FlexRayChannelA” markieren wie in in Abbildung A.1
   d) Steuergeräte (ECU) erstellen mit “Create ECU” (6.Tool von links anklicken) Namen geben: “ECU”
   e) Weiteres ECU erstellen mit dem Namen “BSC” .
   f) Benötigtes **Frame** erstellen, welches die Informationen über das Rücklicht beinhaltet –> “Create Frame” (Briefumschlag Symbol)–>
   Name: Frame_51_0_2
   Length[Byte]: 26
   Slot: 51
   Cycle Repetition: 2

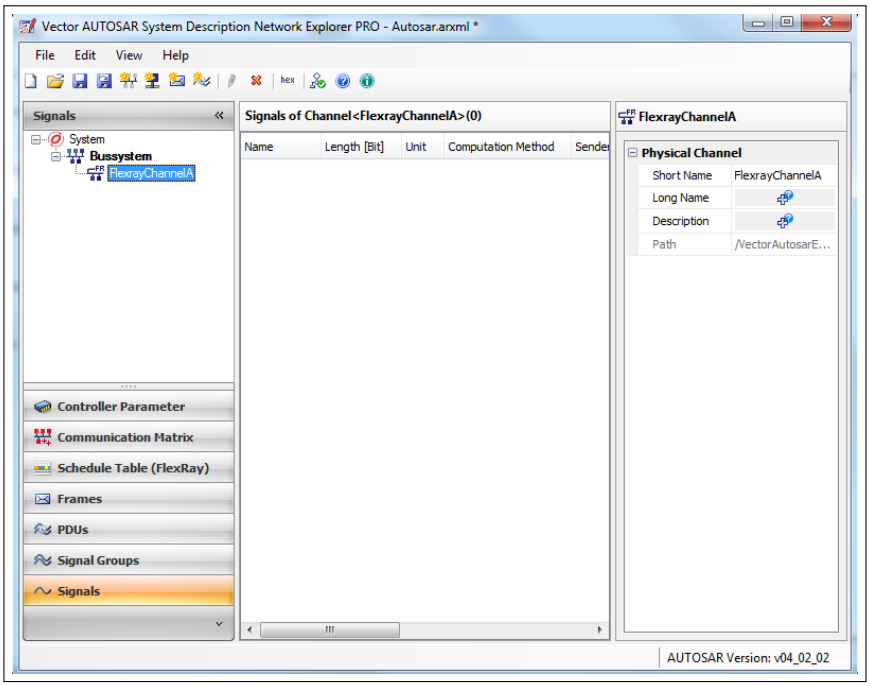


Abbildung A.1: Datenbankausschnitt aus CANoe-Anleitung [Eigene Abbildung]

Sender: BSC
Receiver: BLU
(Siehe Abbildung: A.2) –> OK –> Doppelklick auf das Frame:

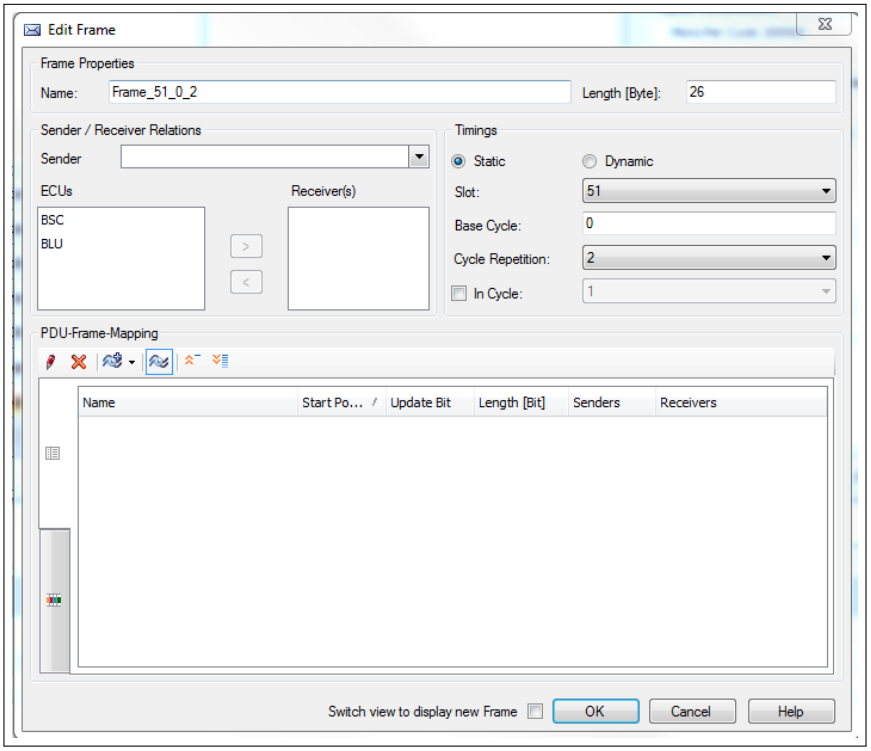


Abbildung A.2: Datenbankausschnitt des Frames aus CANoe-Anleitung [Eigene Abbildung]

g) **Hinzufügen einer Protocol Data Unit (PDU)** über das“PDU-Frame-Mapping” Feld im Frame Fenster –> 3. Symbol von links “Add New/Existing PDU” –> Signal-I-PDU.

h) Neues Fenster öffnet sich –> Name: “BackLightInfo”; Length: 8; Update Bit Position: 200; Byte Order: Motorola; Sender: BSC; Receiver: BLU –> OK –> Wechseln auf PDU Reiter (unten links) und Doppelklick auf das erstellte PDU.

i) **Hinzufügen** der entsprechenden **Signale** über das “Mapped Signales” Feld im Edit Signal-I-PDU Fenster –> 3. Symbol von Links “Add New/Existing Signal”

- 1. Signal: Name: BackUpLight; Length [Bit]:1; Position [Bit]: 6; Byte Order: Motorola –> OK
- 2. Signal: Name: BrakeLight; Length [Bit]:1; Position [Bit]: 7; Byte Order: Motorola –> OK (Siehe Abbildung A.3)

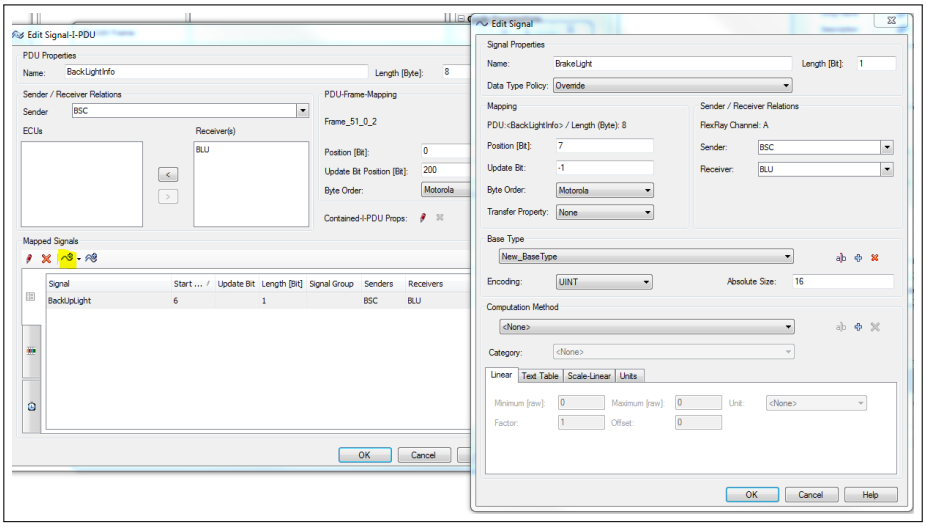


Abbildung A.3: Datenbankausschnitt der PDU und des Signals aus CANoe-Anleitung [Eigene Abbildung]

j) “Edit Frame Fenster” bei BackLightInfo Start Position: “71” manuell eintragen –> OK

k) Am Ende sollte die Darstellung wie in Abbildung A.4 aussehen -> abspeichern.

l) Auf CANoe Simulationsaufbau Darstellung wieder wechseln: –> einfügen der Datenbank über den Simulationsaufbau, Rechtsklick auf “Datenbasen –> Hinzufügen” –> arxml Datei auswählen.

m) Rechtsklick auf die erstellten Netzknoten –> “Konfiguration -> DB-Knoten”: (je nach Name aus der Datenbank) „Bussystem::BLU“

4. **Erstellen von Systemvariablen:**
“Umgebung –> Systemvariablen” –> 1. Symbol von links “Neue Systemvariable anlegen...” wie in Tabelle A.1 aufgeführt ist. Das Ergebnis zeigt Abbildung A.5.

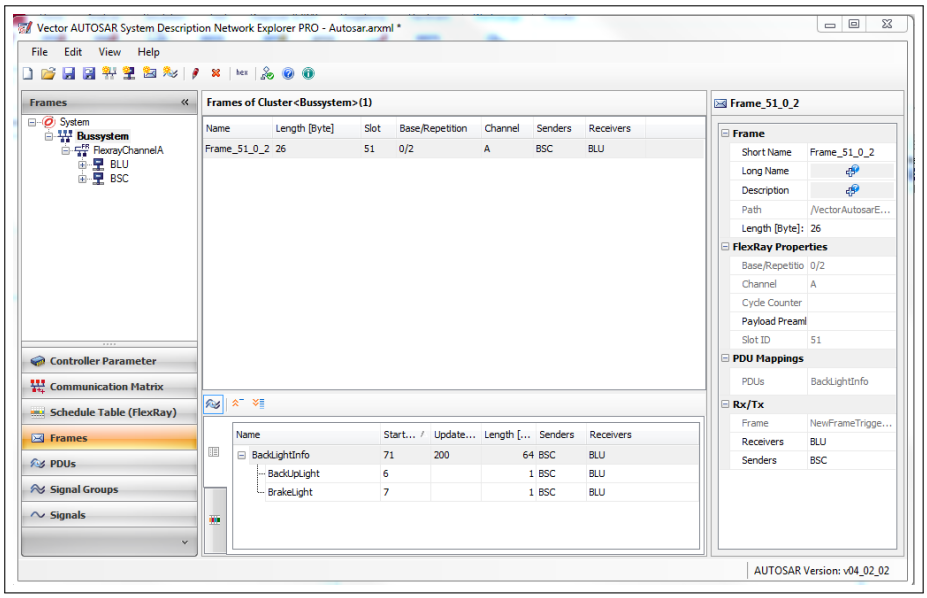


Abbildung A.4: Datenbankausschnitt aus CANoe-Anleitung [Eigene Abbildung]

Namensraum	Panel	Panel	Panel	Panel
Name:	BackLightStatus	BrakeActive	BackUpLight	AccleratorPedal
Datentyp:	Integer 32 Bit	Integer 32 Bit	Integer 32 Bit	Integer 32 Bit
Iniitalwert:	0	0	0	0
Minimum:	-	0	-	-
Maximum:	-	1	-	-

Tabelle A.1: Erstellung der Systemvariablen in CANoe [Eigene Ermittlung]

5. **Panels designen:** Mithilfe des Panel Designers erstellt man geeignete Oberflächen zur Repräsentation der Funktionsweise:

   a) Werkzeuge –> Panel Designer –> neues Fenster öffnet sich und an der rechten Seite Möglichkeit auf den Reiter “Toolbox” zu wechseln, dort sind entsprechende Tools zur Erstellung schon zur Verfügung gestellt

   b) Aus Toolbox via Drag&Drop eine “Group Box” ins Panel reinziehen –> “Eigenschaften” Text: “Pedal” ersetzen.

   c) In die Group Box ein “Switch/Indicator” hineinziehen –> Eigenschaften Image: *pedal*.bmp hochladen –> Tipp: Rechtsklick auf das Bild –> “Größe anpassen”. Weiterhin bei Eigenschaften verändern: Name: SwitchControl1; Button Behaviour: True; Symbol Filter: System Variable; Symbol: “AcceleratorPedal” einstellen (Unter Panel)

   d) In Group Box weiteren “Switch/Indicator” einfügen der über das Bremspedal der beiden Pedale gezogen wird. Eigenschaften Image: *bremse*.bmp hochladen; Name: SwitchControl2; Button Behaviour: True; Symbol Filter: System Variable; Symbol: “BrakeActive”

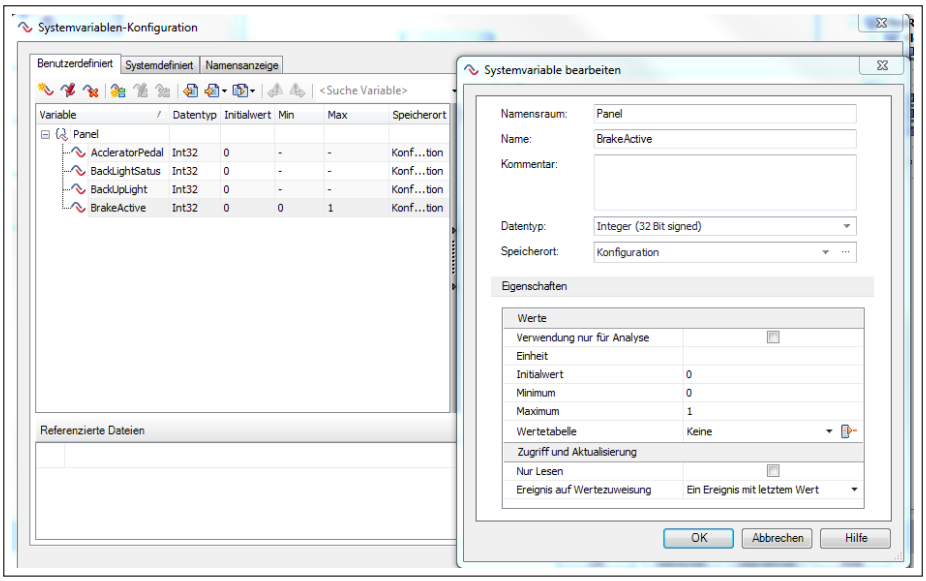


Abbildung A.5: Systemvariablendarstellung aus CANoe-Anleitung [Eigene Abbildung]

e) Eine weitere Group Box ins Panel neben die “Pedal” Group Box hinsetzen mit dem Text: “BackLight”.

f) In diese Group Box, 2 “Switch/Indikator” nebeneinander einfügen und den Namen: “SwitchControl3”, “SwichtControl4” zuweisen und unter Eigenschaften einmal das linke Rücklicht und in den anderen Switch Controller das rechte Rücklicht einfügen weitere Eigenschaften: Symbol Filter: System Variable Symbol: BackLightStatus

g) Jeweils ein “Switch/Idicator” über den bereits gesetzten SwitchControl3 sowie SwitchControl4 genau über den Bremslicht einfügen und bei bei Eigenschaften die entsprechenden Bilder einfügen Eigenschaften: Symbol Filter: System Variable Symbol: BackUpLight –> abspeichern unter “*dateiname*.xvp” die Panels könnten dann wie in Abbildung A.6 aussehen

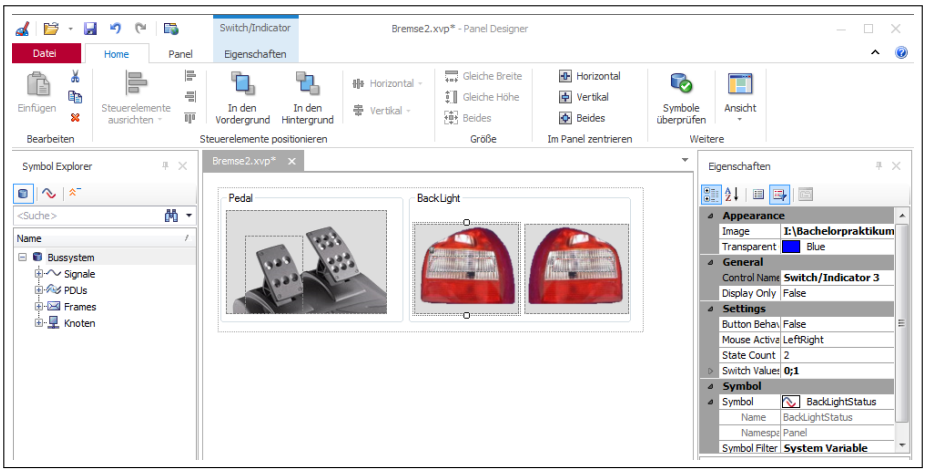


Abbildung A.6: Panels aus CANoe-Anleitung [Eigene Abbildung]

h) Datei -> Zur Konfiguration hinzufügen

6. **Steuerelementen Funktionen übergeben**

   a) Rechtsklick auf BLU –> bearbeiten -> geeigneten Namen vergeben und speichern -> dadurch wird CAPL Browser geöffnet mit dem man den Knoten über den Quellcode Funktionen übergeben kann.
   Am Ende folgenden Quellcode hinzufügen:

```
on signal_update BrakeLight
{
    // The signal will switch the Brake light on/off
    @sysvar::Panel::BackLightStatus = this;
}
```

Listing A.1: BrakeLight

   b) Rechtsklick auf BSC –> bearbeiten (öffnet sich wieder im CAPL Browser):

```
variables
{
    int gBrakePedalStatus = 0;
    const int PRESS = 1;
    const Int NOPRESS = 0;
}

on sysvar sysvar::Panel::BrakeActive
{
    if(@this)
    gBrakePedalStatus = PRESS;
    else
    gBrakePedalStatus = NOPRESS;
    $BrakeLight = @this;
}
```

Listing A.2: BrakeActive

   –> beide Dateien kompilieren über “Compiler –> kompilieren”

7. **Programm starten**

   a) Zurück zum Hauptbildschirm des Vector CANoe –> abspeichern, Starten, Oberflächen Fenster vom Panel Designer wenn nötig wieder in Vordergrund klicken.
   b) Betätigen der Bremse mittels Mausklick.
   c) Ergebnis: Bremslichter gehen an. (Die Enddarstellung der Simulation ist zu Anfang im Kapitel Ergebnisteil in Abbildung 4.1 zu sehen.

Pinbelegung Stecker A51*1B

Pin	Art	Bezeichnung/Signalart	Anschluss/Messhinweise
1	E/A	ohne Parkmanöverassistent ab 2011_09 FlexRay Bus-Signal	Spurwechselwarnung
1	E/A	FlexRay Bus-Signal	Parkmanöverassistent
2	E/A	ohne Parkmanöverassistent ab 2011_09 FlexRay Bus-Signal	Spurwechselwarnung
2	E/A	FlexRay Bus-Signal	Parkmanöverassistent
3	–	nicht belegt	
4	E/A	FlexRay Bus-Signal	Dämpfersatellit hinten rechts
5	–	nicht belegt	
6	E/A	FlexRay Bus-Signal	Vertikaldynamikmanagement
7	–	nicht belegt	
8	E/A	FlexRay Bus-Signal	Dämpfersatellit vorn links
9	E/A	FlexRay Bus-Signal	Dämpfersatellit vorn links
10	–	nicht belegt	
11	E/A	FlexRay Bus-Signal	Verteilergetriebe
12	E/A	FlexRay Bus-Signal	Verteilergetriebe
13	–	nicht belegt	
14	E/A	FlexRay Bus-Signal	Dynamische Stabilitäts-Control (DSC)
15	E/A	FlexRay Bus-Signal	Dynamische Stabilitäts-Control (DSC)
16	E/A	FlexRay Bus-Signal	Elektromechanische Servolenkung
17	–	nicht belegt	
18	E/A	Ethernet Datenleitung	Diagnosesteckdose
19	M	Masse	Massepunkt
20	–	nicht belegt	
21	–	nicht belegt	
22	E/A	FlexRay Bus-Signal	Dämpfersatellit hinten rechts
23	–	nicht belegt	
24	E/A	FlexRay Bus-Signal	Vertikaldynamikmanagement
25	–	nicht belegt	
26	E/A	FlexRay Bus-Signal	
27	E/A	FlexRay Bus-Signal	
28	–	nicht belegt	
29	E/A	bis 2011_08 FlexRay Bus-Signal	Spurwechselwarnung
30	E/A	bis 2011_08 FlexRay Bus-Signal	Spurwechselwarnung
31	–	nicht belegt	

Pin	Art	Bezeichnung/Signalart	Anschluss/Messhinweise
32	E/A	FlexRay Bus-Signal	Integrated Chassis Management
33	E/A	FlexRay Bus-Signal	Integrated Chassis Management
34	E/A	FlexRay Bus-Signal	Elektromechanische Servolenkung
35	–	nicht belegt	
36	E/A	Ethernet Datenleitung	Diagnosesteckdose
37	–	nicht belegt	
38	–	nicht belegt	
39	E	Versorgung Klemme 30F	Sicherung F3
40	–	nicht belegt	
41	E/A	Ethernet Datenleitung	Diagnosesteckdose
42	E	Wecksignal Klemme 15	Car Access System
43	–	nicht belegt	
44	E/A	Diagnose Bus-Signal	Diagnosesteckdose
45	E/A	Diagnose Bus-Signal	Diagnosesteckdose
46	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung
47	E/A	PT-CAN Bus-Signal	PT-CAN Bus-Verbindung
48	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung
49	E/A	K-CAN Bus-Signal	K-CAN2 Bus-Verbindung
50	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung
51	E/A	K-CAN Bus-Signal	K-CAN Bus-Verbindung
52	–	nicht belegt	
53	E/A	Ethernet Datenleitung	Diagnosesteckdose
54	E/A	Ethernet Datenleitung	Diagnosesteckdose

CAPL-Programm

```
variables
{
    int ID, DLC;
    message * msg;
    msTimer cycTimer;
}

on start
{
    msg.id = 0;
    msg.dlc = 0;
    output(msg);
    setTimer(cycTimer,100);
}

on Timer cycTimer
{
    msg.id ++;
    DLC==0;
    output(msg);
    write("msg.ID %f",msg.id);

    if (msg.id < 2049)
    {
        setTimer(cycTimer,100);
    }

    else
    {
        msg.id = 0;
        msg.dlc ++;

        if (msg.dlc < 9)
        {
            setTimer(cycTimer,100);
        }
    }
}
```

Listing A.3: CAPL-Programm

Nachrichtenaustausch

K-CAN –> PT-CAN

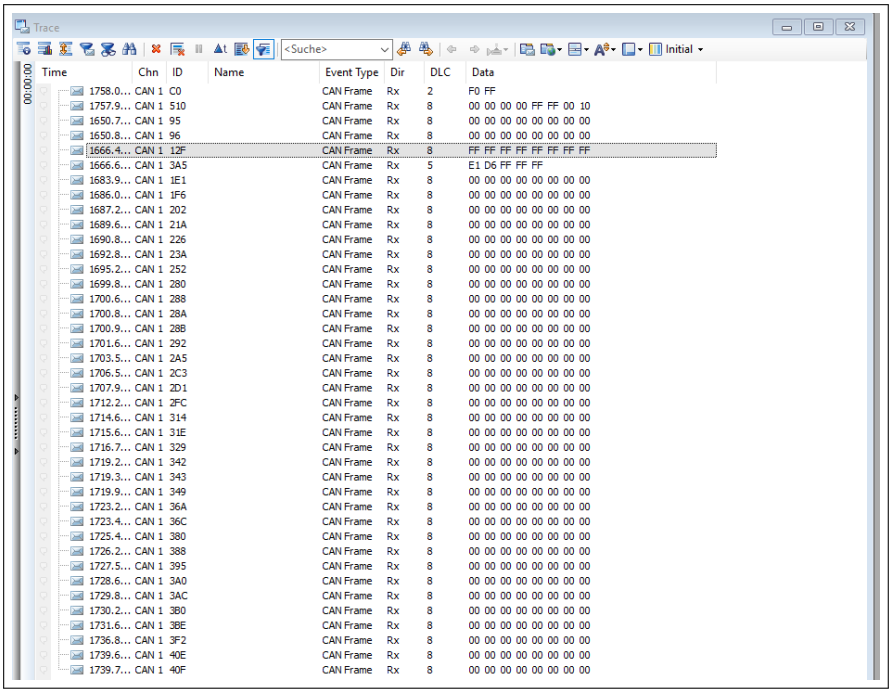


Abbildung A.7: Protokollierung der empfangenden Nachrichten des PT-CANs [Vector Informatik GmbH, 2018]

PT-CAN –> K-CAN

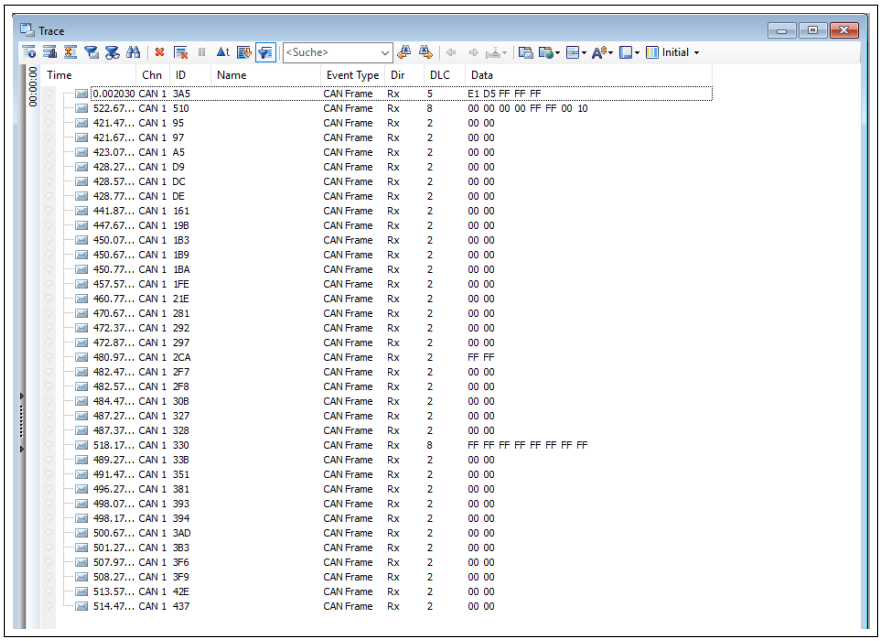


Abbildung A.8: Protokollierung der empfangen Nachrichten des K-CANs [Vector Informatik GmbH, 2018]

Chipbezeichnungen

Nummer	Bezeichnung	Information
1	ELMOS 91056B 125C 0370A	Auto ECU Chip
2	ELMOS 91056B 125C 0370A	Auto ECU Chip
3	**ATMEL MEGA48 15AT1 1035 G6711**	**8-bit Mikrocontroller - MCU** [Atmel, 2016]
4	CY7C1011CV33 – 10ZSXA 1025 708045 PHI F04 (03316)	RAM
5	**25LC256E SN e3 1025 9H4**	**Serial EEPROM** [Microchip technology Inc., 2017]
6	**MPC5567 MVR132 QMM1038 TGGUQ**	**CPU/Mikrocontroller** [Freescale Inc., 2012]
7	MICREL KSZ8893 MQL AM 1036A7K GF2F239 1105 36LWJ3	Ethernet Switch
8	O-A-S-I-S e3 OS8 1050AQ 538E 1A 1035 T14 3348A	
9	LA4032V 75TN 483 B017 RR10	CPLD - komplexe programmierbare

Tabelle A.3: Mikrcontroller Bezeichnungen Vorderseite [Eigene Ermittlung]

Nummer	Bezeichnung	Information
10	KF25 EK035	
11	TJA 104 1AT N0E0L202 Hng 10402	High-Speed CAN transceiver
12	TJA 1055/3/c CS967 403 Tng1040	Enhanced Fault-Tolerant CAN Transceiver
13	TJA 104 1AT N0E0L202 Hng 10402	High-Speed CAN transceiver
14	TJA 104 1AT N0E0L202 Hng 10402	High-Speed CAN transceiver
15	CY7C1011CV33 – 10ZSXA 1031 708129 PHI F 04 (0336)	RAM
16	B5973D EB026	Schaltspannungsregler Power Conversion
17	B5973D EB026	Schaltspannungsregler Power Conversion
18	4299G 1035	LDO Spannungs-Regulator

Tabelle A.4: Mikrcontroller Bezeichnungen Rückseite [Eigene Ermittlung]

Schaltpläne aus XPROG-Software

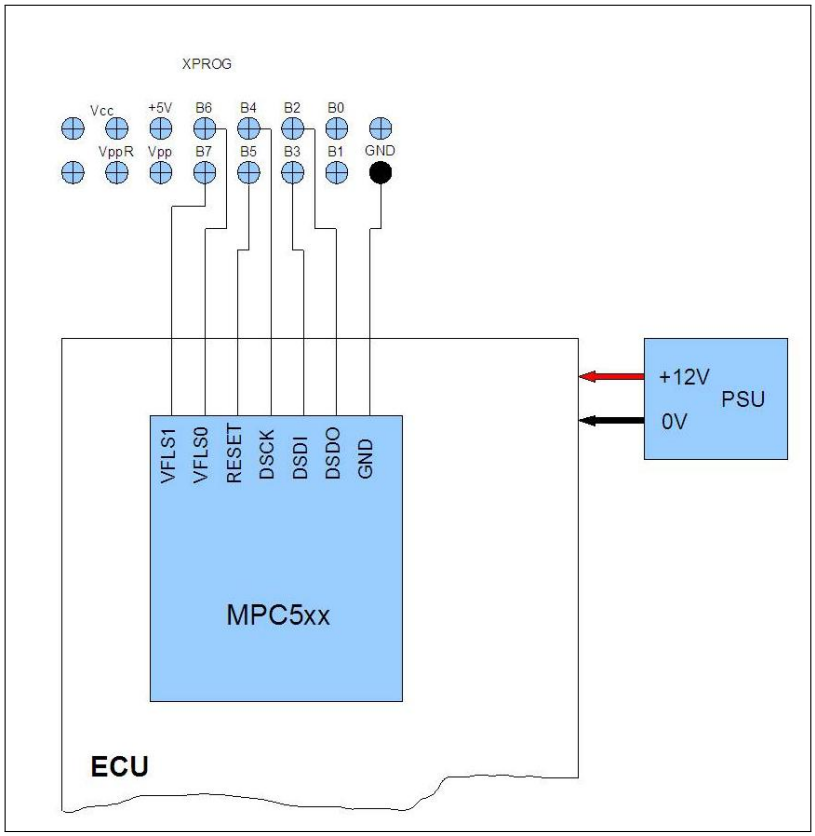


Abbildung A.9: Schaltplan für MPC5XXX [ELDB electronics store, 2011-2018]

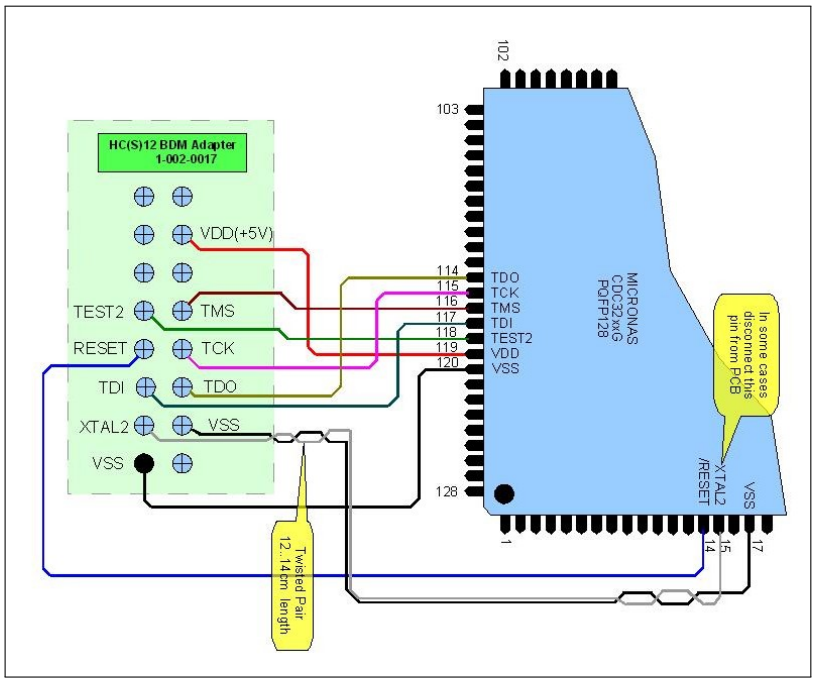


Abbildung A.10: Schaltplan für MICRONASCDC32 [ELDB electronics store, 2011-2018]

Literaturverzeichnis

Abrites LTD. Abrites diagnostics. 2014. URL https://abrites.com/home. Zuletzt besucht: 23.08.2018.

Atmel. *ATmega48/V 88/V 168/V [Datasheet]*, 2016. URL http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2545-8-bit-AVR-Microcontroller-ATmega48-88-168_Datasheet.pdf. Zuletzt besucht: 24.08.2018.

Atmel Corporation. *JTAGICE mkII Quick Start Guide*, 2006. URL http://ww1.microchip.com/downloads/en/DeviceDoc/doc2562.pdf.

BMW Group. *Der neue BMW 7er - Entwicklung und Technik*. Vieweg + Teubner, 2009. S.29ff.

S. Buschmann. *OMNeT++ basierte Simulation von FlexRay Netzwerken zur Analyse von Automotive Anwendungen*. B.A., Hochschule für Angewandte Wissenschaften Hamburg, 2012. S.1-7, 15ff.

CarX24 - Andrea Faltinek. E65_codes.xls. 2008. URL www.carx24.de\/E65_Codes.xls. Zuletzt besucht: 24.08.2018.

ELDB electronics store. Xprog-box hardware. 2011-2018. URL http://www.eldb.eu/index.php?route=information/infocategory&path=4_7. Zuletzt besucht: 30.07.2018.

FlexRay Consortium. Flexray communications system protocol specification version 3.0.1. 2010. URL https://svn.ipd.kit.edu/nlrp/public/FlexRay/FlexRay%E2%84%A2%20Protocol%20Specification%20Version%203.0.1.pdf. Zuletzt besucht: 24.08.2018, S.36ff.

Freescale Inc. *MPC5567 Microcontroller Data Sheet*, 2012. URL https://www.nxp.com/docs/en/data-sheet/MPC5567.pdf. Zuletzt besucht: 24.08.2018.

T. Hoppe. *Prävention, Detektion und Reaktion gegen drei Ausprägungsformen automativer Malware - Eine methodische Analyse im Spektrum von Manipulationen von Schutzkonzepten*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2014. S.55 ff.

International Organisation for Standardization. Iso 11898-1:2015 road vehicles – controller area network(can) – part 1: Data link layer and physical signalling, 2015.

International Organisation for Standardization. Iso 11898-1:2016 road vehicles – communication between vehicle and external equipment for emissions-related diagnostics – part 3: Diagnostic connector and related electrical circuits: Specification and use, 2016.

P. Kleberger, T. Olovsson, und E. Jonsson. Security aspects of the in-vehicle network in the connected car. In *IEEE Intelligent Vehicles Symposium (IV), S.528-533*. 2011.

Kurt Veit - Mixed Mode. Flexray v1.0. 2007. URL `http://www.w3service.net/DL/Vortrag_Flexray_FH_SW.pdf`. Zuletzt besucht: 24.08.2018.

N.-A. Le-Khac, D. Jacobs, J. Nijhoff, K. Bertens, und K.-K. R. Choo. Smart vehicle forensics: Challenges and case study. In *ELSEVIER Future Generation Computer Systems*. 2018.

T. Lecklider. Entertainment rides the most bus. In *EE: Evaluation Engineering. Jun2014, Vol. 53 Issuee 6, S.22-24. 3p.* 2014.

Microchip technology Inc. *25LC256 Data Sheet*, 2017. URL `http://ww1.microchip.com/downloads/en/DeviceDoc/21822E.pdf`. Zuletzt besucht: 24.08.2018.

S. Mukherjee, H. Shirazi, I. Ray, J. Daily, und R. Gamble. Practical dos attacks on embedded networks in commercial vehicles. In *Springer International Publishing AG 2016, S.23-42*. 2016.

N. Navet und F. Simonet-Lion. *Automotive Embedded Systems Handbook*. Industrial Information Technology, 2009.

NewTIS.info. A51 zentrales gateway-modul. 2018a. URL `https://www.newtis.info/tisv2/a/de/f11-530d-xdrive-tou/components-connectors/components/components-with-a/a51-central-gateway-module/BpxkP7fD`. Zuletzt besucht: 23.08.2018.

NewTIS.info. Pinbelegung(pib) a151 zentrales gateway-modul. 2018b. URL `https://www.newtis.info/tisv2/a/de/f11-530d-xdrive-tou/components-connectors/components/components-with-a/a51-central-gateway-module/EZUhUufu`. Zuletzt besucht: 23.08.2018.

D. K. Nillson, U. E. Larson, F. Picasso, und E. Jonsson. A first simulation of attacks in the automotive network communications protocol flexray, 2009.

Nutzername im Forum:chris30o0. E60 vfl can-bus codes. 2015. URL `https://www.bmw-syndikat.de/bmwsyndikatforum/topic366934_CAN-Bus_Projekt__E60__Codierung__Diagnose_und_Programmierung.html`. Zuletzt besucht: 24.08.2018.

Nutzername im Forum:l86. Fehlerspeicher ausgelesen. 2018. URL https://www.f30-forum.de/index.php?thread/8663-fehlerspeicher-ausgelesen/. Zuletzt besucht: 24.08.2018.

Nutzername im Forum:mikeb. Handling 29-bit (extended) format. 2017. URL https://community.carloop.io/t/handling-29-bit-extended-format/239/15. Zuletzt besucht: 09.08.2018.

J. Petit und S. E. Shladover. Potential cyberattacks on automated vehicles. In *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, S.547*. 2015.

K. Reif. *Bosch Autoelektrik und Autoelektronik - Bordnetze, Sensoren und elektronische Systeme, 6. Auflage*. Vieweg + Teubner, 2011. S.92-143.

K. Reif. *Automobilelektronik - Eine Einführung für Ingenieure, 4. Auflage*. Vieweg + Teubner, 2012. S.7-13.

Tencent Keen Security Lab. Experimental security assessment of bmw cars: A summary report, 2018. URL https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf. Zuletzt besucht: 09.08.2018.

Vector Informatik GmbH. *VN7610 FlexRay/CAN Interface Handbuch*, 2016. Version 6.0.

Vector Informatik GmbH. *VN1600 Interface Familie Handbuch*, 2017. Version 4.1.

Vector Informatik GmbH. *User Assistance (deutsch)*, 2018. Version 10.0 SP2.

A. Vetter. Ross-tech hex-net. 2018. URL https://vcdspro.de/produkte/ross-tech-hex-net/. Zuletzt besucht: 02.08.2018.

VIN-Info Sp. z o.o. Bestellbericht. 2001-2018. URL https://de.vin-info.com/berichte-bestellen/WBAMX31060C524214. Zuletzt besucht: 02.07.2018.

D. Vogt. *Praktikumsbericht: Entwurf und Ausbau einer Testumgebung für Sicherheitsmechanismen ausgewählter eingebetteter Systeme*. Hochschule Mittweida, 2018.

W. Zimmermann und R. Schmidgall. *Bussysteme in der Fahrzeugtechnik – Protokolle, Standards und Softwarearchitektur, 5. Auflage.* Springer Verlag, 2014. S.57-138.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 27. August 2018

Maria Juliane Clausing