„Grundlegende Informationen zum CAN-Bus“
von Thomas Wedemeyer

1. Vorwort

Dieser Text ist ein Ausschnitt aus meiner Studienarbeit, die sich mit der Entwicklung eines CAN auf RS232 / SSI-Interfaces beschäftigt. Der Text soll die grundlegenden Eigenschaften des CAN-Busses zusammenfassen und einen Überblick über die Einsatzmöglichkeiten geben.

2. Der CAN-Bus

2.1 Grundlagen des CAN-Bus

Beim CAN-Protokoll handelt es sich um ein Bus-Protokoll, das seit Anfang der 80'er Jahre auf betreiben der Automobilindustrie entwickelt wurde. Ziel der Entwicklung war es durch die Einführung eines Bussystems im Kraftfahrzeug die Verkabelung der einzelnen Systeme, Sensoren und Aktoren zu vereinfachen und Kosten zu sparen. Für den Einsatz im KFZ-Bereich wurden deshalb die folgenden Anforderungen an das Kommunikationsprotokoll formuliert und später im CAN-Protokoll realisiert:

Der CAN-Bus muss für so genannte Klasse-C Applikationen geeignet sein. D.h., der Bus muss in der Lage sein zeitkritische Informationen mit Zykluszeiten von 1 bis 10ms und Botschafts-Latenzzeiten von unter 1ms zu übertragen. Die Länge der einzelnen Botschaften umfasst dabei nur wenige Byte. Die zu erwartende Datenrate bei dieser Anwendung liegt im Bereich von 250kBit/sec bis 1MBit/sec. Beispiele für typische Klasse-C Applikationen sind der Bereich des Motormanagements und der Stabilitätskontrolle.

Darüber hinaus sollte der CAN-Bus aber auch für so genannte Klasse-A Applikationen geeignet sein. In dieser Klasse sind Applikationen zusammengefasst, die nur geringe Datenraten benötigen und eine große Zykluszeit zwischen den Datentransfers besitzen. Die Datenrate liegt bei dieser Applikation unterhalb von 10kBit/sec. Im KFZ-Bereich ist ein Beispiel für diese Applikationsklasse die gesamte Chassis-Elektronik & Elektrik im Auto, also Zentralverriegelung, Bremsleuchten, Blinker usw.

Eine Anforderung, die sich aus Sicherheitsgründen ergibt, ist die Multimaster-Fähigkeit des CAN-Busses. Bei einem Multimaster-System ist jeder einzelne Knoten in der Lage eine Kommunikation einzuleiten. Dadurch wird vermieden, das z.B. durch einen Defekt in einem Knoten die Funktion des Gesamtsystems gefährdet ist. Für den Fall, das mehrere Knoten gleichzeitig versuchen auf dem Bus Daten zu übertragen, müssen entsprechende Vorkehrungen getroffen werden, um diese Zugriffskonflikte zu erkennen und aufzulösen. Der CAN-Bus bedient sich der Zugriffstechnik CSMA/CD+CR (Carrier Sense, Multiple Access/ Collision Detection + Collision Resolution). Einzelheiten zur Kollisionserkennung und Busarbitierung können dem Kapitel 2.3 entnommen werden.

Ein weiterer Vorteil ist, daß durch die Multimaster-Fähigkeit eine ereignisgesteuerte Kommunikation ermöglicht wird. D.h., jeder Teilnehmer ist in der Lage die Übertragung von Informationen, die durch ein Ereignis erzeugt wurden, selbstständig auszulösen. Dabei werden die Informationen mit einer so genannten

Broadcast-Kommunikation übertragen. Broadcast-Kommunikation heißt, dass die gesendete Information von allen Teilnehmern empfangen wird. Die Auswertung der empfangenen Daten in den Teilnehmern erfolgt anhand des gesendeten inhaltsbezogenen Identifiers der Nachricht. Aus diesem Grund wird jedem Ereignis ein eindeutiger Identifier zugeordnet.

Weiterhin ist beim CAN-Protokoll auch die Möglichkeit vorgesehen durch Anforderung das Versenden von Informationen auszulösen. Diese Abfrage von Informationen wird Remote Request genannt. Bei einem Remote Request wird ein spezieller Remote Frame gesendet, der keine Daten sondern nur den Identifier der gewünschten Nachricht enthält. Dieser Remote Frame wird dann von einem Teilnehmer mit einem normalen Datenframe beantwortet. Die Nachricht wird wiederum von allen Teilnehmer empfangen und die Datenkonsistenz im ganzen System ist sichergestellt.

Bei der Entwicklung des CAN-Protokolls wurde auch darauf geachtet, dass Möglichkeiten zur Erkennung von Übertragungsfehlern vorhanden sind. Dazu wurde eine Fehlererkennung in mehreren Ebenen vorgesehen. Auf Botschaftsebene ist eine Fehlererkennung mittels einer im Telegramm übertragenen 15-Bit CRC-Prüfsumme (CRC:=Cyclic Redundancy Check) implementiert. Darüber hinaus ist auch eine Fehlererkennung auf der physikalischen Übertragungsebene vorgesehen. Weitere Einzelheiten zur Fehlererkennung können den nachfolgenden Kapiteln entnommen werden.

Durch die oben beschriebenen Eigenschaften des Protokolls setzt sich der CAN-Bus in der Autoindustrie seit 1990 immer mehr durch. Inzwischen wird von fast allen Herstellern dieses Protokoll in ihren Fahrzeugen verwendet. Darüber hinaus setzt sich der CAN-Bus auch im Bereich der Industriesteuerungen und der Automatisierungstechnik durch. Die weitere Verbreitung des CAN-Busses auch außerhalb der Automobilindustrie führte dazu, dass der Wunsch nach dem CAN-Protokoll als »offenes« Kommunikationsprotokoll immer größer wurde. 1992 wurde dann von der CiA (CAN in Automation) damit begonnen die OSI Layer 1,2 & 7 zu definieren.

Layer		***Aufgabe***
7	*Application*	Funktionen zum Zugriff auf die Daten bereitstellen
6	*Presentation*	Konvertierung & Formatanpassung von Daten
5	*Session*	Dienste zum Auf- & Abbau von Sitzungen
4	*Transport*	Verbindung zwischen Prozessen herstellen
3	*Network*	Vermittlung der Daten zwischen verschiedenen Quellen & Zielen
2	*Data Link*	Sichere Übertragung der Telegramme sicherstellen
1	*Physical*	Definiert die Übertragung der einzelnen Bits auf dem Übertragungsmedium

Abbildung 1: Aufbau des OSI-Modells

Im OSI Layer 1 wurden detaillierte Spezifikationen für die physikalische Ebene der Kommunikation festgelegt. In diesem Layer sind z.B. Empfehlungen für Kabel, Stecker und Leistungstreiber zusammengefasst Der OSI Layer 2, der Data Link Layer, steuert und schützt die Kommunikation auf der Ebene eines Frames. Die OSI Layer 3 bis 6 sind für CAN nicht notwendig. Die Spezifikation auf der Ebene des Application-Layers (Layer 7)

ist nicht festgelegt. Für diesen Layer sind aber verschiedene Protokolle definiert worden (z.B. SDS, DeviceNET, OpenCAN, usw.), aber es gibt keine bindende Spezifikation.

2.2 Aufbau des CAN-Frames

Für den inhaltlichen Aufbau eines CAN-Frames, also eines CAN-Telegramms, existieren im Augenblick zwei Spezifikationen, die sich hauptsächlich in der Anzahl der Identifierbits unterscheiden. In einem Standard CAN-Frame gemäß der CAN-Spezifikation 2.0A wird der Identifier durch 11-Bit festgelegt. Damit lassen sich 2032 verschiedene logische Adressen kodieren. Das heißt, dass in einem Netzwerk maximal 2048 verschiedene Informationen verschickt werden können. Da dies für viele Anwendungen nicht ausreicht ist die Anzahl der Identifier in der Spezifikation 2.0B auf 29 Bit erweitert worden. Durch diese Erweiterung ist es möglich 536.870.912 verschiedene Informationen zu übertragen. Telegramme, die gemäß der Spezifikation 2.0B aufgebaut sind, werden als Extended-CAN-Frames bezeichnet. Der genaue Aufbau der Frames ist in der Abbildung 2 & Abbildung 3 dargestellt.

Dataframe CAN 2.0A (11 Bit Identifier)

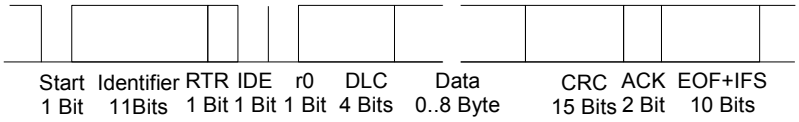

Abbildung 2: Aufbau des Standard-CAN-Telegramms

Dataframe CAN 2.0B (29 Bit Identifier)

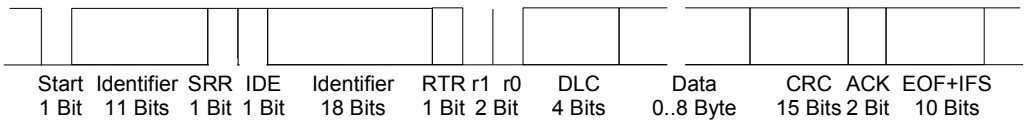

Abbildung 3: Aufbau des Extended-CAN-Telegramms

Eingeleitet wird ein CAN-Frame durch ein Startbit, das Low-Pegel führt (dominantes Bit), durch dessen fallende Flanke die einzelnen Teilnehmer synchronisiert werden. Nach dem Startbit folgen 11 Bit, die den Identifier bilden. Der Identifier kennzeichnet die logische Adresse und die Priorität der Nachricht. Je kleiner die logische Adresse der Nachricht ist, desto höher ist die Priorität der Nachricht. Bei der Übertragung des Identifiers wird das MSB zuerst übertragen.

Nun folgt bei einem Standard CAN-Frame die Übertragung des Remote Transmission Request Bits (RTR-Bit). Mit diesem Bit wird ein Remote-Frame gekennzeichnet. Dieser Frame-Typ enthält keine Daten, sondern fordert eine Nachricht von einem anderen Teilnehmer an. Die Übertragung der angeforderten Nachricht erfolgt direkt im Anschluss an die Übertragung des Remote Frames, vorausgesetzt der Bus wird nicht vorher durch eine Nachricht mit höherer Priorität belegt.

Als nächstes wird das Control Feld übertragen, das aus 6 Bits besteht. Bei einem Standard-Frame ist das erste Bit immer low. Dieses Bit wird IDE (Identifier Extension) genannt. Mit diesem Bit wird zwischen Standard und Extended-Frames unterschieden. Das Bit »r0« ist für zukünftige Erweiterungen reserviert. In den nächsten 4 Bit des Controlfeldes ist die Anzahl der Datenbytes in dem Telegramm kodiert.

In dem folgenden Datenfeld werden die Nutzdaten übertragen. Es können bis zu 8 Datenbyte pro Telegramm übertragen werden. Um Übertragungsfehler in den vorangegangenen Feldern zu erkennen, wird dann eine

15 Bit lange CRC-Prüfsumme übertragen. Zur Bestätigung, dass die übertragene Nachricht fehlerfrei empfangen wurde, stehen zwei ACK-Bits (Acknowledge) zur Verfügung. Mit diesen Bits zeigen die anderen Teilnehmer dem Sender, dass die Nachricht richtig empfangen wurde. Dazu legt der Sender einen rezessiven Pegel (1) auf den Bus und wartet darauf, dass die anderen Teilnehmer diesen Pegel mit einem dominanten Pegel (0) überschreiben. Durch dieses Acknowledge-Verfahren ist sichergestellt, dass mindestens ein Teilnehmer des Netzwerks die Nachricht richtig empfangen hat.

Beendet wird die Übertragung eines Frames durch sieben rezessive Bits, die die Bit Stuffing Kodierung verletzt (Siehe auch Kap. 2.3). Dieses Kennzeichen wird EOF (=End Of Frame) genannt. Nach dem EOF werden noch drei weitere rezessive Bits (IFS=Inter Frame Space) eingefügt, mit denen sichergestellt werden soll, dass die Teilnehmer genügend Zeit haben, um die empfangene Nachricht zu verarbeiten.

Die Unterscheidung zwischen einem Standard und einem Extended CAN-Frame erfolgt anhand des IDE-Bits. Führt das IDE-Bit einen rezessiven Pegel handelt es sich bei dem übertragenen Telegramm um einen Extended Frame. In diesem Fall hat das bei einem Standard Frame vor dem IDE-Bit gesendete RTR-Bit keine Bedeutung. Deshalb wird dieser »Platzhalter« bei Extended Frames als Substitute Remote Request Bit bezeichnet. Das IDE-Bit wird gefolgt von den restlichen 18 Bit des Identifiers, dem RTR-Bit und dem 6.Bit langen Control Feld. Die Bits r1 & r0 in dem Control Feld sind für zukünftige Entwicklungen reserviert.

Wie man an dem Aufbau der CAN-Frames gemäß der Spezifikation 2.0A & 2.0B sieht, ist es für einen CAN-Controller nicht schwierig das Format der einzelnen Nachrichten-Telegramme zu unterscheiden. Dadurch ist es möglich gleichzeitig Standard & Extended CAN-Frames über ein Netzwerk zu übertragen. Voraussetzung dafür ist allerdings, dass die Controller die beiden Standards erkennen können. Es gibt viele CAN-Controller, die nur die CAN Spezifikation 2.0A aktiv unterstützen. Im Allgemeinen verhalten sich die neueren dieser Typen aber gegenüber CAN 2.0B passiv,d.h. sie ignorieren Telegramme, die nach CAN 2.0B Spezifikation übertragen werden.

2.3 Bit-Kodierung und Arbitrierung

Die einzelnen Bits eines Frames werden über das Netzwerk im NRZ-Verfahren (Non Return to Zero) mit Bitstuffing übertragen. Bei NRZ-Verfahren werden die Daten digital übertragen, wobei die Dauer jedes Bits gleich lang ist und die Bits direkt aufeinander folgen. Der High-Pegel wird dabei als logisch Null und der Low-Pegel als logisch Eins interpretiert. Durch das NRZ-Verfahren wird die zur Verfügung stehende Bandbreite optimal genutzt. Allerdings enthält dieses Verfahren keine Informationen, die zur Synchronisation bei der Übertragung benutzt werden können. Um bei längeren Datentelegrammen eine sichere Synchronisation auf die einzelnen Bits sicherzustellen, werden die Flankenwechsel im Bitstrom zur Nachsynchronisation des Bittakts benutzt.

Um sicherzustellen, dass dieses Verfahren auch funktioniert, wenn in einem Bitstrom keine Flankenwechsel auftreten, wird in den Bitstrom nach genau 5 gleichen Bits ein so genanntes komplementäres Stuff-Bit zusätzlich eingefügt (to stuff:=hinein stopfen). Auf der Empfängerseite wird dieser Stuff-Bit automatisch vom Controller wieder aus dem Bitstrom heraus gefiltert.

Der Buszugriff erfolgt beim CAN-Bus mit Hilfe einer zerstörungsfreien, bitweisen Arbitrierung. Die Voraussetzung für dieses Buszugriffsverfahren sind Bustreiber, die in der Lage sind einen rezessiven 1-Pegel und

einen dominanten 0-Pegel zur Verfügung zu stellen. D.h. der rezessive 1-Pegel kann durch den dominanten 0-Pegel auf dem Bus überschrieben werden. (Siehe dazu auch Kapitel 2.5)

Bei diesen Arbitrierungsverfahren wird verhindert, dass zwei Teilnehmer gleichzeitig Nachrichten auf dem Bus übertragen nachdem beide Teilnehmer einen freien Bus erkannt haben, in dem der Identifier bitweise auf den Bus gelegt wird. Während sie dies tun lesen die Teilnehmer gleichzeitig den Bitstrom wieder ein und vergleichen die Bitwerte mit dem gesendeten Wert. Wird dabei festgestellt, dass ein rezessiver Pegel von einem anderen Teilnehmer mit einem dominanten Pegel überschrieben wurde stellt der Teilnehmer, dessen Bit überschrieben wurde, sofort seine Übertragung ein und wartet bis der Bus wieder frei ist. Durch dieses Verfahren ist sichergestellt, dass immer die Nachricht mit der höchsten Priorität den Zugriff auf den Bus erhält. Dieses Arbitrierungsverfahren wird zerstörungsfrei genannt, weil der Gewinner der Arbitrierung sein Telegramm nicht erneut von vorn zu senden beginnen muss

2.4 Ausnahmebehandlung

Durch den Aufbau der CAN-Telegramme und das Bit-Stuffing-Verfahren sind CAN-Controller in der Lage Fehler während der Übertragung zu erkennen. Festgestellt werden können Bit-Fehler, Bit-Stuffing-Fehler, CRC-Fehler, Formatfehler im Telegramm und auch Acknowledgement-Fehler.

Wird einer dieser Fehler von einem Teilnehmer erkannt informiert er sofort die anderen Teilnehmer und den Absender des Telegramms, indem er einen Error-Frame sendet. Durch den Empfang eines Error-Frames verwerfen alle Teilnehmer die empfangene Nachricht und beginnen ebenfalls einen Error-Frame zu senden. Wenn der Bus wieder frei ist, beginnt dann der Absender der Nachricht mit einer Wiederholung der Sendung. Dadurch ist sichergestellt, dass durch Übertragungsfehler keine Informationen verloren gehen.

Ein Error-Frame besteht aus einer bewussten Verletzung der Kodierungsvorschrift. In einem Error-Frame wird einfach eine Folge von 6 oder mehr aufeinander folgenden dominanten Bits gesendet. Damit ist ein Error-Frame eine Verletzung der Bit-Stuffing-Regel und der Error-Frame wird als Bit-Stuffing-Fehler von den anderen Teilnehmern erkannt.

Um sicherzustellen, dass ein defekter Teilnehmer, der nicht in der Lage ist Nachrichten richtig zu empfangen, das gesamte Netzwerk blockiert, ist in jedem Controller ein spezieller Algorithmus implementiert. Mit Hilfe dieses Algorithmus zieht sich der Controller im Fehlerfall schrittweise vom Busgeschehen zurück.

Stellt der CAN-Controller fest, dass er als erster Netzteilnehmer einen Error-Frame gesendet hat, erhöht er einen internen Fehlerzähler. Solange der Wert des Fehlerzählers unter 126 liegt sendet der Controller im Fehlerfall die oben beschriebenen Error-Frames. Dieser Fehlerbehandlungsmodus wird »Active Error« genannt. Liegt der Wert des Fehlerzählers aber über 126 schaltet der Controller in den so genannten »Error Passiv«-Modus. In diesem Modus sendet der Controller einen 6 Bit Error-Frame mit einem rezessiven Pegel. Wenn der Fehlerzähler einen Stand von 255 erreicht hat wird der Controller in den »Bus Off«-Modus geschaltet und nimmt nicht mehr an der Kommunikation auf dem Bus teil. Aus dem »Bus Off«-Modus kann der CAN-Controller nur noch durch einen Reset befreit werden. Aus dem »Error Passiv«-Modus kann sich der Controller selbst befreien, da der Fehlerzähler um eins dekrementiert wird, wenn ein anderer Teilnehmer einen Fehler zuerst erkannt hat.

Die Bearbeitung des Fehlerhandlings erfolgt vollständig im CAN-Controller. Ein an den CAN-Controller angeschlossener Mikroprozessor wird nur über Flags über den aktuellen Fehlerbehandlungsmodus informiert.

2.5 Die physikalische CAN-Schnittstelle

Im Prinzip kann die physikalische Schnittstelle des CAN-Busses auf unterschiedliche Weise realisiert werden. Wichtig ist dabei nur, dass es möglich ist auf dem Bus rezessive und dominante Bits darzustellen. Neben Konzepten zur Datenübertragung über optische Eindrahtverbindungen gibt es verschiedene Konzepte, die eine Übertragung der Daten mit Hilfe von Differenzsignalen vorsehen.

Im Allgemeinen wird die physikalische Schnittstelle des CAN-Busses gemäß der ISO 11898 realisiert. Bei dieser Verbindung handelt es sich um eine Zweidraht-Verbindung mit Differenzsignalen. Die maximale Übertragungsgeschwindigkeit beträgt 1MBit/sec bei einer Kabellänge von 40m, wobei maximal 30 Busknoten installiert sein dürfen. Bei geringeren Datenraten sind größere Kabellängen möglich, z.B. bei einer Übertragungsrate von 125kBit/sec beträgt die zulässige Kabellänge 500m.

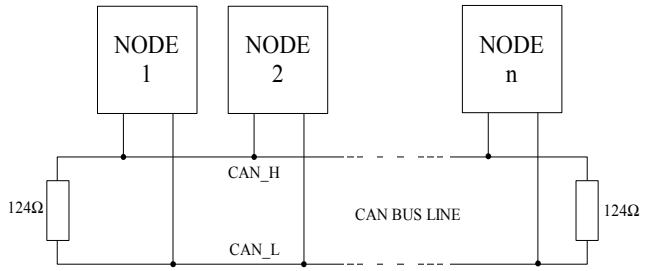

Abbildung 4: Aufbau des CAN-Netzwerks

Die Nennimpedanz des Buskabels wird mit 120Ω angegeben. An den beiden Enden der zwei CAN-Signalleitungen, die CAN_H und CAN_L genannt werden, ist ein Leitungsabschluß von jeweils 124Ω bei 1% Toleranz und 200mW Verlustleistung vorgesehen.

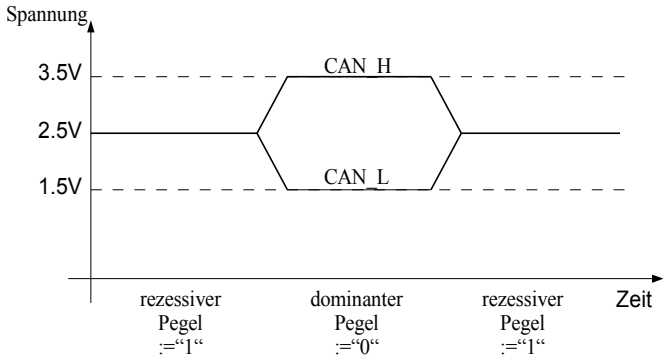

Abbildung 5: Signalpegel des CAN-Bus

Laut Spezifikation liegt auf dem Bus ein rezessiver Pegel an, wenn das Signal des CAN_H-Signals nicht höher liegt als das Signal CAN_L+0,5V. Der dominante Pegel wird durch eine Spannung an CAN_H definiert, die mindestens 0,9V höher liegt als die der CAN_L Leitung.

In dem vereinfachten Blockschaltbild in Abbildung 6 ist die interne Beschaltung eines CAN-Transreceivers dargestellt. Soll ein rezessiver Pegel (TxD:=1) auf den Bus gelegt werden sperren die beiden Transistoren im Transreceiver. Da über die Transistoren und den Terminierungswiderstand kein Strom fließt fällt auch keine Spannung über dem Widerstand ab. D.h., die Spannung, die an den beiden Eingängen des Receiver anliegt, ist gleich. Es wird also ein rezessiver Pegel über den Receiver zurückgelesen.

Durch einen 0-Pegel am TxD-Eingang wird auf dem CAN-Bus ein dominater Pegel erzeugt. In diesem Fall schalten die beiden Transistoren im Transreceiver durch. Damit fließt ein Strom über die Transistoren und den Terminierungwiderstand. An dem Terminierungswiderstand fällt dabei eine Spannung ab. Dadurch liegt am CAN_H-Eingang des Receivers eine höhere Spannung an als an dem CAN_L-Eingang. Diese Spannungsdifferenz wird von dem Receiver als dominanter Pegel zurückgelesen.

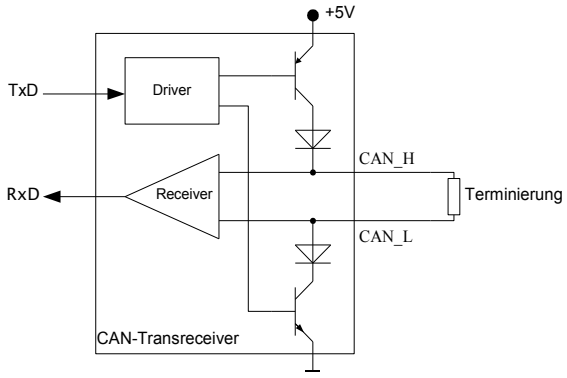

Abbildung 6: Vereinfachtes Blockschaltbild eines CAN-Transreceivers

Bisher gibt es für die Steckverbindungen zwischen einzelnen CAN-Komponenten keine Normierung im OSI 1 Layer. Allerdings gibt es eine Empfehlung, die von der Gruppe »CAN in Automation« (CiA) entwickelt wurde. Bei dem Stecker handelt es sich um eine 9-polige Sub-D Steckverbindung. Die Pinbelegung kann der Tabelle 1 entnommen werden.

PIN	***Signal***	***Beschreibung***
1	-	(reserviert)
2	CAN_L	Signal CAN_L
3	CAN_GND	Signal GND
4	-	(reserviert)
5	-	(reserviert)
6	CAN_SHLD	Signalabschirmung
7	CAN_H	Signal CAN_H
8	-	(reserviert)
9	CAN_V+	Optionales Power Supply V+

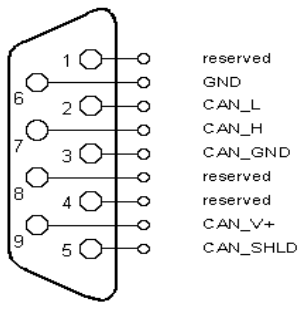

Abbildung 7: Pinbelegung des CAN-Steckers

Tabelle 1: Pinbelegung gemäß CiA Draft Standard 102 V2.0

3. Literatur

Lawrenz, W.

CAN Controller Area Network-Grundlagen und Praxis

Hüthing-Verlag 98

4. Kontakt

World Wide Web: www.thomas-wedemeyer.de

Email: mail@thomas-wedemeyer.de

5. Versionen

- Orignaltext: Studienarbeit vom September 1999
- V1.0 vom 26.01.2000
- V1.1 vom 13.04.2006
  Neues Layout, kleine Fehler korrigiert (11-Bit ID, CRC)

