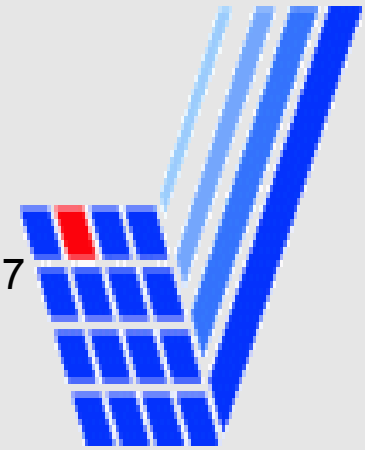
CAN

Sebastian Kosch

sebastian.kosch@googlemail.com

PG AutoLab
Seminarwochenende 21.-23. Oktober 2007



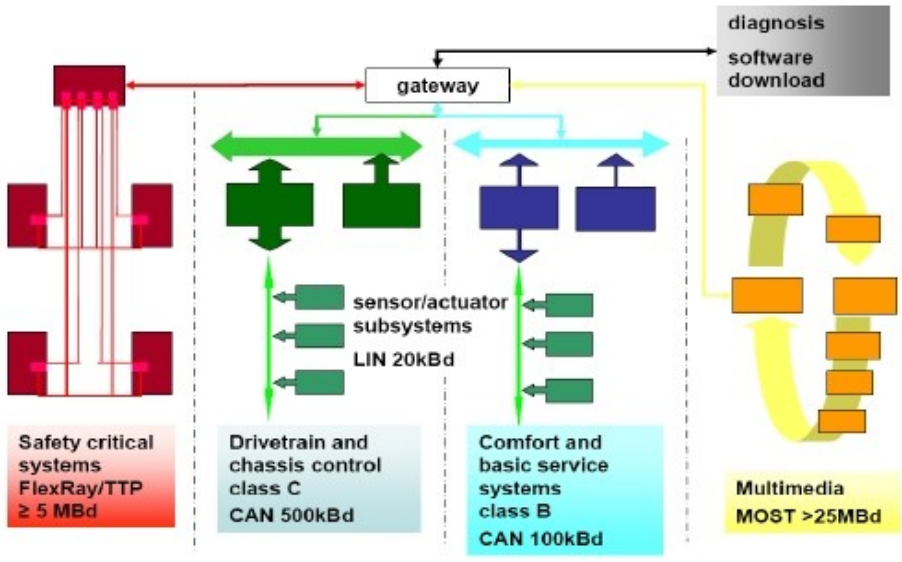
Inhaltsverzeichnis

Der CAN-Bus

Der CAN (Controller-Area-Network)-Bus wurde 1983 von der Firma Bosch entwickelt. Die Länge der Kabelbäume in den Autos zu dieser Zeit, betrug bis zu 2 km pro Fahrzeug. Durch die Entwicklung des neuen Bussystems sollten die Kabelbäume wesentlich verkürzt werden, wodurch neben dem Material auch das Gewicht der Autos reduziert wurde.
Der Can-Bus findet seine Anwendung nicht nur im Automobil Bereich. Er wird auch bei der Entwicklung medizinischer Geräte, Fahrstuhl-Steuerungen und in der Automatisierungstechnik (z.B. bei Kuka-Robotern) eingesetzt.
Durch seine Eigenschaft der mehrfachen Sicherung bei der Datenübertragung und seiner Echtzeitfähigkeit, ist der CAN-Bus besonders für Umfelder geeignet, in denen schwierige Bedingungen herrschen (z.B. elektromagnetische Felder).

Der CAN-Bus im Auto

Der CAN Bus wird in den so genannten Klasse B und C Systemen in Automobilen eingesetzt. Die Klasse C Systeme umfassen unter anderem den Antrieb und die Fahrwerkkontrolle und die Klasse B Systeme beinhalten die Komfort Systeme.
Da der CAN-Bus in verschiedenen Gebieten, welche unterschiedliche Anforderungen an den Bus haben, eingesetzt wird, kommt je nach Gebiet ein entsprechender CAN-Bus zur Anwendung. Hier ist besonders zwischen dem High-Speed- und dem Low-Speed-CAN zu unterscheiden.
Der High-Speed-CAN hat eine Übertragungsrate von 125 KBit bis zu 1 MBit. Im Automobil wird der High-Speed-Can z.B. bei der Vernetzung der Motorelektronik, der Getriebesteuerung, dem ABS oder dem Airbag benutzt. Gerade beim Airbag und dem ABS, kann man sehr gut erkennen, dass sein Einsatz eine hohe Anforderung an eine korrekte Datenübertragung hat. Ein Fehler könnte gerade im Auto schwere Unfälle hervorrufen. Insgesamt können mit dem High-Speed-CAN maximal 20 Knoten vernetzt werden.
Der Low-Speed-CAN wird z.B. bei der Vernetzung der Klimaanlage, dem Scheibenwischer, dem Schiebedach oder der Zentralverriegelung benutzt. Dieser Bus besitzt eine Datenrate von maximal 125 KBit und es können maximal 30 Knoten mit ihm vernetzt werden.



Beide Busse werden mit einer zwei adrigen Leitung realisiert. Die so genannten CAN-High und CAN-Low Leitungen. Über diese Leitungen werden zwei komplementäre Signale gesendet, wodurch auch bei einer Unterbrechung einer Leitung, ein Signal übertragen wird. Allerdings wird der Low-Speed-CAN nicht mit Widerständen terminiert. Es gibt auch noch einen Single-Wire-CAN, der nur mit einer Leitung vernetzt wird.

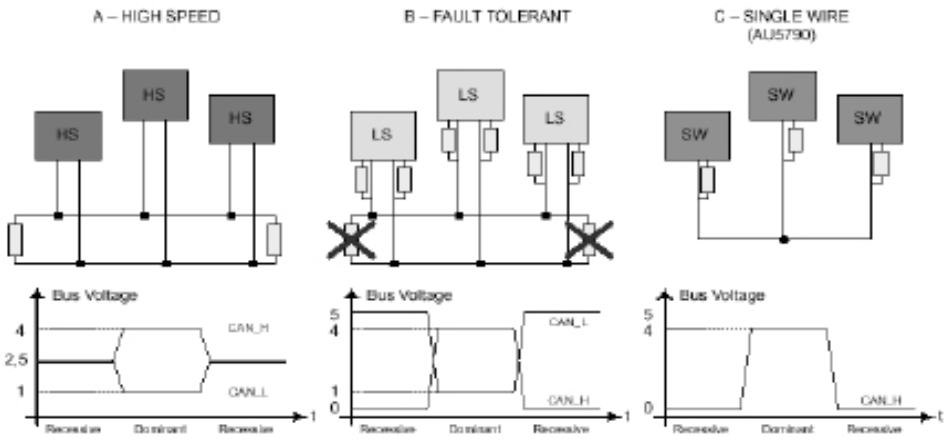


Abb. 6-15 : Physikalische Medien für A: Standard-CAN, B: Low-Speed, Fault-tolerant, C: Single-Wire [2]

Auf der Abbildung oben sind auch die rezessiven und dominanten Buspegel zu sehen. Beim High-Speed-CAN liegen der rezessive Pegel bei 4V und der dominante bei 1V. Dies hat den Vorteil, dass die Knoten den Unterschied zwischen einer durchgetrennten Leitung und einem dominanten Signal sofort registrieren.

10 kbits/s	6,7 km
20 kbits/s	3,3 km
50 kbits/s	1,3 km
125 kbits/s	530 m
250 kbits/s	270 m
500 kbits/s	130 m
1 Mbits/s	40 m

Der CAN-Bus hat zwar eine maximale Datenübertragungsrate von 1 MBit/sec, allerdings kann diese Rate nur auf einer Kabellänge von max. 40 m garantiert werden. Der Grund für diese Einschränkung liegt bei der gleichzeitigen Verarbeitung der Daten von den einzelnen Knoten. Diese kann bei längeren Kabeln nicht mehr realisiert werden.

Das Prinzip des Datenaustausches

Um einen Datenaustausch zwischen den einzelnen Knoten zu ermöglichen, benutzt das CAN-Protokoll keine Geräte- oder Systemadressen, sondern Nachrichten-Identifier. Dieser Identifier ist eindeutig, woraus folgt, dass jede Nachricht ihren eigenen Identifier hat. Durch den Identifier werden zwar einzelne Systemadressen unnötig, allerdings beschränkt er auch die Anzahl der Nachrichten. Im Standard-CAN (2.0 A) beträgt die Länge des Identifiers 11 Bit, womit man 2048 Nachrichten einen eindeutigen Identifier zuordnen kann. Wenn diese 11 Bit nicht ausreichen um die Anzahl der vorhandenen Nachrichten einen Identifier zu zuweisen, steht auch noch der Extended-CAN (2.0 B) zur Verfügung. Das Extended-CAN Format besitzt einen 29 Bit Identifier und kann wesentlich mehr Nachrichten einen Identifier zuweisen als der Standard-CAN. Auf den genauen Unterschied der beiden Nachrichtenformate wird weiter unten eingegangen.

Neben der Eigenschaft der Eindeutigkeit des Identifiers ist es auch wichtig, dass der Identifier (zusammen mit dem Remote-Transmission-Bit) die Priorität einer Nachricht festlegt. Diese Prioritätenvergabe des Identifiers an die Nachrichten wird während der Arbitrationsphase beim Nachrichtenaustausch benutzt, welche noch beschrieben wird.

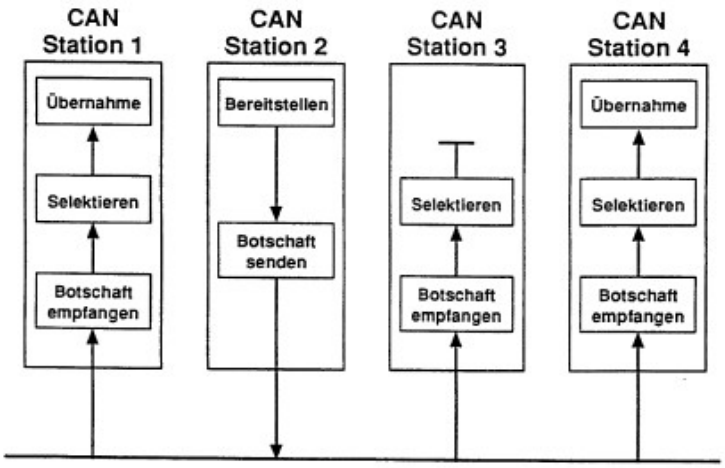
Das Sende- und Empfangsprinzip beim Nachrichtenaustausch

Wenn die CPU einer Station eine Nachricht über den CAN-Bus senden will, gibt diese die Nachricht, den Identifier und die Übertragungsaufforderung an den CAN-Baustein, der ihr zugeordnet ist. Der CAN-Baustein bildet dann die Nachricht und überträgt sie zum CAN-Bus. Nun kann es vorkommen, dass mehrere CAN-Bausteine gleichzeitig auf den CAN-Bus zugreifen wollen. In diesem Fall kommt es zur Arbitrierungsphase. Während dieser Phase wird herausgefunden,

welche die Nachricht mit der höchsten Priorisierung ist und diese erhält dann den Vorrang gegenüber niedriger priorisierten Nachrichten.

Während der Arbitrierungsphase werden die Identifier und das RTR-Bit (Remote-Transmission-Bit) aller Nachrichten betrachtet die gleichzeitig Zugriff zum Bus haben wollen. Das RTR-Bit ist im Nachrichtenformat genau hinter dem Identifier und gibt an ob es sich um eine zu sendende Nachricht handelt oder ob eine Nachricht von einem anderen Knoten angefordert wird. Das RTR-Bit ist besonders wichtig, wenn ein Knoten eine Nachricht senden und ein anderer gleichzeitig diese Nachricht anfordern will. In beiden Fällen ist der Identifier gleich. Durch das RTR-Bit (welches bei der Anforderung auf eins gesetzt wird) wird die Anforderung nun niedriger priorisiert als die Nachricht, die gesendet werden soll. Also wird die Nachricht gesendet, wodurch der Knoten der die Nachricht angefordert hat, die Anforderung kein zweites mal senden muss.

Sobald ein CAN-Baustein die Buszuteilung hat, werden alle anderen Busteilnehmer zu Empfängern. Alle Busteilnehmer nehmen nach dem korrektem Empfang der Nachricht eine Akzeptanzprüfung vor. Wenn die Nachricht für den Empfänger ist, registriert er dies an dem Identifier und verarbeitet die Nachricht. Ist die Nachricht nicht für den Empfänger gedacht, registriert er es wiederum an dem Identifier und verwirft die Nachricht.



Broadcast-Übertragung und Akzeptanzprüfung durch die CAN-Knoten

Standard- vs. Extended-CAN

Das Standard-CAN-Format besteht aus einem Start-of-Frame, dem Arbitration Field, dem Control Field, dem Data Field, dem Ack Field und dem End of Frame. Es ist zu beachten, dass das Standard-Format bei gleicher Priorität dem Extended-Format vorgezogen wird.

Im Start-of -Frame wird ein dominantes Bit auf den Bus gelegt, damit alle Knoten wissen das nun eine Nachricht gesendet wird.

In dem Arbitration Field befinden sich, wie weiter oben schon beschrieben, der Identifier und das

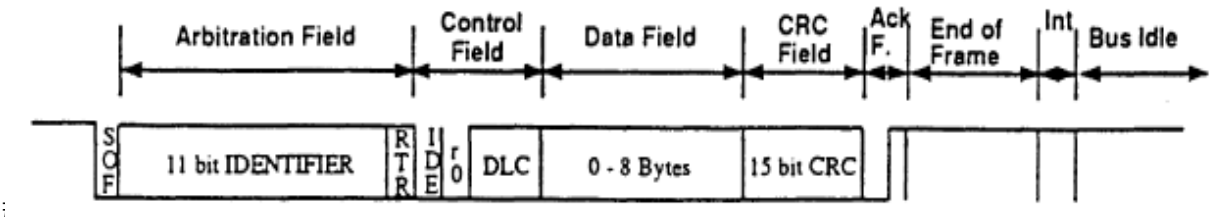
RTR-Bit (Zur Erinnerung: Identifier= eindeutige Identifikationsnummer der Nachricht, RTR-Bit= wird gesetzt wenn eine Nachricht angefordert wird). Mit dem Arbitration Field wird herausgefunden, welche die am höchsten priorisierte Nachricht ist, wenn mehrere CAN-Bausteine Zugriff auf den Bus haben wollen. Die am höchsten priorisierte Nachricht wird nach der Arbitrationsphase gesendet.

Das Control Field beinhaltet das IDE-Bit, das r0-Bit und den DLC (Data Length Code). Das IDE-Bit gibt an, ob die Nachrichten im Standard- oder im Extended-Format gesendet wird. Das r0 Bit ist reserviert für zukünftige Erweiterungen des Formats. Im Data Length Code muss angegeben werden, wie viel Bytes an Daten mit der Nachricht mitgeschickt werden.

In dem Control Field wird die CRC-Checksumme der Nachricht mitgesendet. Die CRC-Summe wird von jedem Empfänger der Nachricht berechnet und mit der mitgesendeten CRC-Summe verglichen.

Während des Ack Fields, legen Knoten, die die Nachricht nicht korrekt empfangen haben einen dominanten Pegel auf den Bus. Da die an den Bus angeschlossenen Knoten mit dem Bus eine interne Wired-AND Schaltung bilden, werden die rezessiven Pegel von den anderen Knoten überschrieben und der Sender weiß, dass ein Fehler vorliegt.

Im End-of-Frame wird eine Folge von sieben rezessiven Bits gesendet. Da sieben Bits gleicher Priorität gegen die Bit-Stuffing Regeln verstößt (siehe Fehlererkennung), werden alle Knoten wieder synchronisiert.



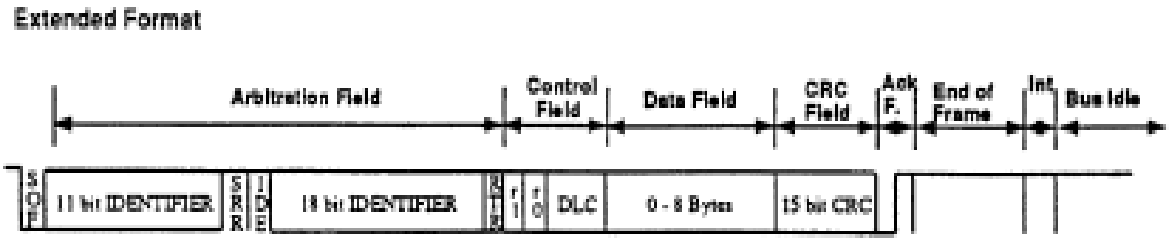
Botschaftsrahmen im Standard-Format (CAN-Spezifikation 2.0A)

Das Extended-Format unterscheidet sich nur in wenigen Feldern vom Standard-Format. Nach dem Start-of-Frame wird auch hier erst einmal ein 11 Bit Identifier gesendet, worauf dass SRR-Bit (Substitute Remote Request Bit) und das IDE-Bit folgen.

Durch das SRR-Bit hat das Extended-Format eine niedrigere Priorität als das Standard-Format, da es dem RTR-Bit im Standard-Format entspricht aber hier rezessiv gesendet wird. Im Anschluss an das SRR-Bit wird wie im Standard-Format das IDE-Bit gesendet.

Nach dem IDE-Bit werden dann die restlichen 18 Bit des Identifiers gesendet. Auf den Rest des Identifiers folgt das RTR-Bit für das Extended-Format, damit auch in diesem Format ein Telegramm angefordert werden kann. Die nächsten zwei Bits (r1,r0) sind reserviert für zukünftige Erweiterungen des Extended-Formats.

Der restliche Telegrammaufbau entspricht dem des Standard-Formats, welcher schon weiter oben beschrieben wurde.



Botschaftsrahmen im Erweiterten-Format (CAN-Spezifikation 2.0B)

Fehlererkennung

Der CAN-Bus wird, wie man schon durch die Anwendung beim ABS erkennt, auch in sicherheitskritischen Bereichen benutzt. In diesen Bereichen ist es besonders wichtig, dass ein auftretender Fehler durch verschiedene Mechanismen erkannt und die Nachricht erneut gesendet wird.

Im CAN-Protokoll werden fünf verschiedene Mechanismen zur Fehlererkennung realisiert: Die Bitüberwachung, die Überwachung des Nachrichtenformats, die zyklische Blockprüfung, das Bit-stuffing und das Senden eines Acks durch den Sender.

Bei der Bitüberwachung vergleicht der Knoten, der eine Nachricht sendet, seinen Soll-Pegel mit dem Ist-Buspegel. Wenn nun der Bus einen anderen Pegel aufweist, als der vom Knoten gesendete Pegel, wird der Sendevorgang wiederholt.

Jeder Knoten überwacht unter anderem auch das Nachrichtenformat. Die Formate sind als Standard- oder Extended-Format vorgegeben (siehe oben). Wenn nun ein Knoten die Verletzung des Formats (z.B. wenn 10 Byte Daten gesendet werden) erkennt, fordert er den Sender zu einer erneuten Übertragung der Nachricht auf. Die Aufforderung des Empfängers an den Sender zur Wiederholung des Sendens findet immer statt, wenn ein Empfänger einen Fehler erkennt.

Die CRC-Checksumme, die mit jeder Nachricht mitgesendet wird, dient auch der Fehlererkennung. Jeder Knoten, der eine Nachricht empfängt berechnet seinerseits auch die CRC-Checksumme und vergleicht sie mit der mitgesendeten. Die Checksumme wird berechnet, indem man ein ausgewähltes Polynom durch die Nachricht dividiert. Der Rest der Division bildet die CRC-Checksumme.

Das Bit-Stuffing wird benutzt um eine längere Folge von Bits gleicher Wertigkeit zu verhindern. Wenn nun von einem Sender mehr als fünf Bits gleicher Wertigkeit gesendet werden, wird vom Sender ein Stuff Bit (mit komplementärer Wertigkeit zu den zu übertragenen Bits) eingefügt. Die Stuff Bits werden von den Empfängern wieder entfernt. Falls ein Empfänger doch eine Folge von mehr als fünf gleichwertigen Bits feststellt, ist ein Fehler erkannt und der Sender wird benachrichtigt.

Der letzte Mechanismus zur Fehlererkennung ist das Überwachen des Acks durch den Sender. Jeder Sender geht nämlich von einem Fehler aus, wenn er kein Ack von den Empfängern bekommt. Die Knoten mit dem Bus ergeben, wie schon oben erwähnt, eine Wired-And Schaltung. Um ein Ack zu senden legen die Empfänger eine 1 auf den Bus. Falls ein Empfänger einen Fehler erkennt, legt er eine 0 auf den Bus, wodurch ein dominanter Buspegel entsteht. Dieser dominante Buspegel wird vom Sender erkannt, was eine Wiederholung des Sendevorgangs zur Folge hat.

Damit ein außer Kontrolle geratener Knoten nicht immer wieder das wiederholte Senden einer Nachricht zur Folge hat, gibt es Error-Counter. Der Error-Counter setzt einen Knoten, der eine bestimmte Anzahl von Fehlermeldungen überschritten hat, in einen passive Error Mode . In diesem Modus kann der Knoten kein wiederholtes Senden der Nachricht bewirken.

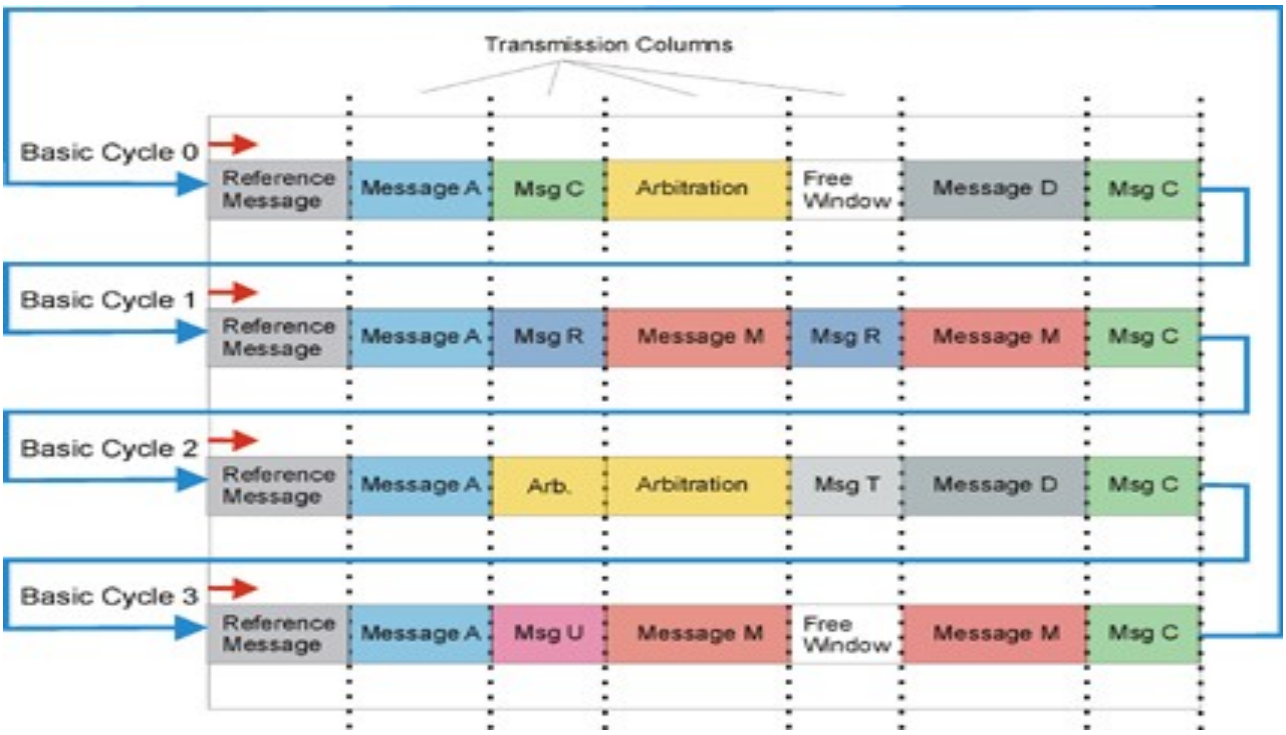
Echtzeitverhalten

Wie man in den oberen Kapiteln feststellt, ist der CAN-Bus Ereignis orientiert, wodurch eine Realisierung von Echtzeitverhalten nicht möglich ist. Um das Echtzeitverhalten dennoch zu erhalten wurde der TTCAN (Time-triggered communication on CAN) entwickelt. Wenn man den TTCAN benutzt, verliert man nicht zwangsweise das Ereignis gesteuerte Verhalten des CAN-Busses. Ein Ereignis kann immer den Zyklus des Echtzeitverhaltens unterbrechen.

Beim TTCAN werden Basiszyklen festgelegt, in denen die Nachrichtenfolgen zeitlich angeordnet werden. Am Anfang von jedem Basiszyklus wird ein reference frame gesendet um alle Knoten zu synchronisieren. Durch die zeitliche Anordnung im Basiszyklus wissen alle Knoten zu welchem

Zeitpunkt sie ihre Nachricht senden können. Sollte ein Knoten ausfallen, werden die Frames von einem anderen Knoten übernommen. Diese Frame-Übernahme kann bis zu fünf mal stattfinden. In manchen Zeitfenstern ist eine Arbitrierungsphase vorgesehen. Während diesem Zeitfenster können alle Knoten versuchen ihre Nachricht zu übertragen und die am höchsten priorisierte Nachricht wird nach der Arbitrierung gesendet. Zeitfenster können auch frei gelassen werden. Die freien Zeitfenster können dann für spätere Erweiterungen benutzt werden.

Um die Synchronisation der Knoten zu gewährleisten, wird nicht ein einziger langer Basiszyklus verwendet, sondern mehrere kurze hintereinander. Zu jedem Basiszyklus werden alle Knoten wiederholt synchronisiert. Für das Senden der reference frames ist ein aktiver Time-Master zuständig. Sollte der Time-Master ausfallen, kann die Aufgabe von bis zu acht passiven Time-Mastern übernommen werden. In den reference frames sind die aktuelle Zeit des Time-Masters , sowie die Zyklusnummer enthalten.



CAN-Bausteine

Um den verschiedenen Anforderungen der unterschiedlichen Anwendungsgebiete gerecht zu werden, gibt es zwei verschiedene Arten um Knoten an den Bus anzuschließen.

Zum einen wäre da der Stand-alone Protokollcontroller-Baustein, welcher über eine Schnittstelle

mit einer CPU verbunden wird. Üblicherweise wird diese Art Baustein direkt an den Bus angeschlossen und über eine Speicheradresse angesprochen (Memory Mapping). Der Stand-alone -Baustein wird in der Regel für niedrige Übertragungsraten benutzt.

Des weiteren gibt es die Mikrocontroller mit integriertem Protokollcontroller-Modul. Diese Mikrocontroller werden gewöhnlicher Weise für Mess-, Regel- und Steueraufgaben benötigt. Die Controller werden verbaut um hohe Übertragungsraten zu erreichen.

Um nur Ein- und Ausgabefunktionen zu realisieren werden I/O-Bausteine mit integriertem Protokollcontroller-Modul, so genannte SLIO-CAN-Controller (SLIO= Serial Linked I/O) eingesetzt. Die SLIO-Controller besitzen digitale und/oder analoge Ein- und Ausgänge und dienen der kostengünstigen Vernetzung von Sensoren und Aktoren ohne lokale Intelligenz.

Um der hohen Stör- und Spannungsfestigkeit gerecht zu werden, ist es empfehlenswert einen externen Transceiver-Baustein einzusetzen. Die externen Transceiver wandeln die Ausgänge des Controllers in physikalische Bussignale und gleichzeitig den Buspegel in einen Logikpegel um, der zum Controller weitergereicht wird.

CAN-Software

Zur Zeit haben fast alle Automobilhersteller ihre eigene OEM-spezifische Standardsoftware. Die einzelnen Softwaremodule werden von verschiedenen Softwarefirmen implementiert. Das hat zur Folge, dass ein modularer Baukasten entstanden ist, der in Verbindung mit den Steuergeräte-zulieferern in verschiedenen Fahrzeugbaureihen verwendet wird. Das Resultat ist, dass sich durch die hohe Anzahl von komplexer Standardsoftware, sehr viele Konfigurationsmöglichkeiten ergeben.

Das AUTOSAR Konsortium hat sich zum Ziel gesetzt die hohe Anzahl verschiedener Software durch eine industrieweite standardisierte Software zu ersetzen.

Die CAN-Open Software geht schon in die Richtung eines Standards. Mit CAN-OPEN lassen sich herstellerunabhängige Geräte mit definierten Telegrammen parametrisieren. Da aber nicht alle herstellerspezifischen Features abgedeckt werden können, resultiert die Anwendung von CAN-Open in einen Overhead von ca. 16 KByte.

Fazit

Der CAN-Bus ist ein Bussystem, welches sehr zuverlässig ist. Selbst in Umfeldern mit hohen

Störfaktoren kann der CAN-Bus eingesetzt werden. Je nach Anzahl der Knoten die an den CAN angeschlossen werden, kann man sich für den Standard- oder Extended-CAN entscheiden. Wenn man zudem ein Echtzeitverhalten benötigt, wird der TTCAN eingesetzt, ohne das ereignisgesteuerte Verhalten des normalen CAN-Bus zu verlieren.

Die Auswahl der CAN-Bausteine orientiert sich neben der Preisfrage auch an der Übertragungsrate, an den Störfeldern im Anwendungsgebiet und an der Art der Hardware, die an den Bus angebracht werden soll.

Bei der Software ist zu beachten, dass es immer noch eine sehr große Menge an unterschiedlicher herstellerspezifischer Software gibt. Der Versuch eine industrieweite Standardsoftware zu entwickeln, ist mit CAN-Open schon teilweise geglückt. Allerdings auf Kosten eines Overheads von ca. 16 KByte.

Letztendlich ist der CAN-Bus in heutigen Autos unverzichtbar. Der CAN wird unter anderem bei der Motorsteuerung, der Getriebesteuerung, dem ABS und dem Airbag benutzt. Allein die Anwendung in diesen Teilgebieten des Automobils sollte die Zuverlässigkeit des CAN-Bus unterstreichen.

Literaturverzeichnis

- www.can-cia.org
- www.auto.tuwien.ac.at/LVA/DA_04.pdf
- cis.cs.tu-berlin.de/Lehre/SS-03/Sonstiges/technInfo/CAN-Ausarbeitung.pdf
- ivs.cs.uni-magdeburg.de/bs/lehre/wise9900/proro/vortrag/can/CAN.pdf
- http://www7.informatik.uni-erlangen.de/~dulz/fkom/06/4.pdf
- www.rtos.de.PDF-file/CAN/Deutsch/intro-d.pdf
- www.semiconduktors.bosch.de/pdf/can.pdf
- www.canopen.org
- ls12-www.cs.uni-dortmund.de/~falk/LEHRE /SS05/Proseminar/Auto_Docs/ausarbeitung_auto_16.pdf
- svenherzfeld.de/can/candipl.pdf
- Konrad Etschberger: CAN; Grundlagen, Protokolle, Bausteine, Anwendungen, Hanser Verlag, 1994