Hauptseminar

SS 2006

Ausgewählte Kapitel eingebetteter Systeme

(AKES)

OSEK COM und CAN

Markus Walter

28.06.06

Inhaltsverzeichnis

1. CAN

1.1. Entstehung und Eigenschaften

Der CAN (Control Area Network) - Bus wurde 1983 von Bosch entwickelt. Dabei handelt es sich um ein serielles und asynchrones Bussystem, dass für die Vernetzung von Steuergeräten im Automobilbereich entwickelt wurde. 1985 wurde der Bus dann zusammen mit Intel vorgestellt. Mittlerweile wird CAN auch in der Automatisierungstechnik verwendet[1].

Ein Entwurfsziel des Busses war, die Kabellängen (bis zu 2km) in den Fahrzeugen zu reduzieren und dadurch Gewicht zu sparen.[2] Des Weiteren wurde von dem Bus gefordert, die angeschlossene Elektronik miteinander störungssicher zu verbinden.

Der CAN-Bus gehört zu den sogenannten Feldbussen. Durch diese werden eine Vielzahl von Feldgeräten wie Messfühler (Sensoren), Stellglieder und Antriebe (Aktoren) mit einem Steuerungsgerät verbunden.

Beim CAN-Bus werden alle Nachrichten grundsätzlich als Broadcast versendet. Die einzelnen Teilnehmer können die Nachrichten jedoch filtern, so dass sie nur die für sie interessanten Datagramme empfangen.

Der CAN-Bus ist mittlerweile weit verbreitet und wird neben der Automobilindustrie, wo er als de facto Standard gilt noch in Zügen, der Luftfahrt und in Fahrstühlen eingesetzt[3]. Die Verfügbarkeit von kostengünstigen Protokollbausteinen für praktisch alle wichtigen Mikrokontrollerfamilien trägt mit zu der weiten Verbreitung bei.

Die Vorteile von CAN liegen in dem Verlustfreien gleichzeitigen Buszugriffsverfahren und der kurzen Blocklänge. Da die Wahrscheinlichkeit eines defekten Datentelegramms proportional mit der Blocklänge wächst, ist hier der CAN-Bus mit seinen max. ca. 8 Byte langen Nachrichten klar im Vorteil. Somit muss bei einem elektromagnetisch stark gestörten Umfeld keine teuren Kabel eingesetzt werden.

Ein weiterer Punkt für den CAN-Bus ist die geringe Latenzzeit für hochpriore Nachrichten. Bei der Höchstprioren Nachricht liegt diese bei maximal 130 Bitzeiten[4].

---

[1] Scherff (u.a.), Feldbusysteme in der Praxis, S. 41

[2] http://de.wikipedia.org/wiki/Can-bus

[3] http://www.can-wiki.info/

[4] Etschberger, CAN, S. 37.

1.2. Übertragungsverfahren

Der CAN-Bus arbeitet nach dem CSMA/CA (Carrier Sense Multiple Access/ Collision Avoidance) Verfahren. Anders als bei Ethernet (CSMA/CD Carrier Sense Multiple Access/ Collision Detection), werden bei gleichzeitigem Buszugriff die kollidierenden Pakete nicht zerstört, sondern durch die Arbitrierung vermieden.

Die Buszugriffsstrategie arbeitet nach dem priorisierten „Multi-Master-Prinzip“. Das Zugriffsrecht auf den Bus wird also nicht von einem übergeordneten Steuergerät geregelt, sondern jeder Teilnehmer kann gleichberechtigt mit dem Senden einer Nachricht beginnen, sofern der Bus frei ist. Wollen mehrere Teilnehmer gleichzeitig senden, so erhält im Rahmen des Arbitrierungsprozesses derjenige mit der höchstprioren Nachricht den Buszugriff[5].

Der Bus wird entweder über Glasfasern oder über Kupferleitungen vernetzt. Bei der Verwendung von Kupferleitungen werden normalerweise 2 Leitungen geführt, die den Bus unempfindlich gegenüber Gleichtaktstörungen machen. Wird ein entsprechendes Busfehlermanagement eingesetzt, kann sogar bei einem Bruch oder Kurzschluss einer Leitung die Kommunikation weitergeführt werden. Desweiteren ist auch der Betrieb mit einer Eindrahtleitung möglich, wenn allen Busteilnehmern eine gemeinsame Masse zur Verfügung steht[6].

1.2.1. NRZ

Daten werden nach dem NRZ-L (Non-Return-to-Zero) Verfahren kodiert. Dabei wird eine logische „1“ als ein High-Pegel und eine logische „0“ als ein Low-Pegel übertragen. Da die übertragenen Daten ausschließlich aus den Nutzdaten bestehen und keine Bandbreite für Synchronisation verwendet wird, ist dieses Verfahren z.B. gegenüber der Manchestercodierung bandbreitenschonender.

Werden z.B. eine Folge von Einsen übertragen, bleibt das Signal auf dem High-Pegel. Hier erkennt man, dass bei der NRZ-Kodierung keinerlei Informationen über den Bittakt übertragen werden, so dass ein Verfahren zur Synchronisation der Busteilnehmer erforderlich ist. Dazu wird das „Bitstuffing“ verwendet.

---

[5] Etschberger, CAN, S. 35/36.

[6] Etschberger,CAN, S. 97.

1.2.2. Bitstuffing

Nach fünf gleichen Bits wird ein inverses Bit eingefügt und die monotone Bitfolge somit unterbrochen. Jeder Teilnehmer der dann eine Folge von fünf gleichen Bits auf der Leitung erkennt, nimmt das sechste Bit, überprüft danach, ob es gegen die Regeln des Bitstuffing verstößt und entfernt es dann. Wird ein „falsches“ Bit gefunden, so liegt ein Übertragungsfehler vor.

Beispiel:

Es soll die Bitfolge 10011111110 übertragen werden. Durch Bitstuffing kommt dann die Folge 10011111**0**110 auf den Bus. Die **0** wurde zusätzlich eingefügt. Der Empfänger löscht dann die zusätzlich eingefügte „0“ und erhält somit die ursprüngliche Nachricht.

1.2.3. CRC

Um Fehler bei der Übertragung oder Duplizierung von Daten erkennen zu können wird das „cyclic redundancy check“-Verfahren genutzt. Dabei werden die zu übertragenden Bits als dyadisches Polynom betrachtet und durch ein Generatorpolynom modulo-dividiert. Der sich ergebende Rest ist der CRC-Wert[7].

Beispiel:

Zu übertragende Bits: 11001

Dyadisches Poylnom: $1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0$

Das CAN-Bus Generatorpolynom lautet[8]:

$$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$$

Das Polynom hat eine Hamming-Distanz von 6, dh. es können bis zu 6 Einzelbitfehler oder bis zu 15 direkt aufeinanderfolgende Bitfehler pro Datenpaket erkannt werden[9]. Der Prüfwert ist 15-Bit lang.

1.2.4. Arbitrierung

Unter der Arbitrierung versteht man das Aushandeln des Medienzugriffs.

---

[7] http://de.wikipedia.org/wiki/Cyclic_Redundancy_Check

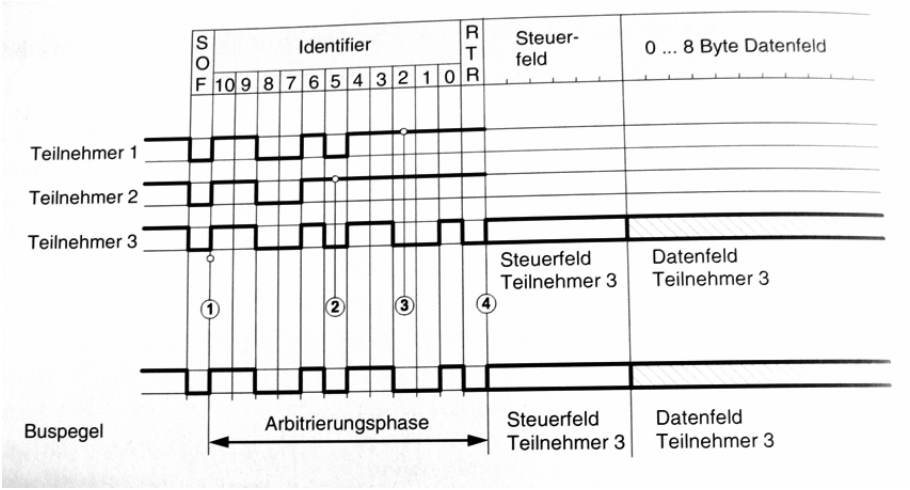
[8] Etschberger, CAN, S. 72.

[9] Lawrenz, CAN Grundlagen und Praxis, S. 88.

Voraussetzung für dieses Übertragungsprotokoll ist ein physikalisches Übertragungsverfahren, welches dominante Pegel (Low) besitzt, das rezessive Pegel (High) überschreiben kann.

Beim Senden überwacht jeder Teilnehmer gleichzeitig den Bus und vergleicht die anliegenden Daten mit den von ihm gesendeten. Stellt er eine Abweichung fest, so stoppt er augenblicklich die Übertragung.

Folgendes Beispiel erklärt die Arbitrierung:



[10]

Teilnehmer 1, 2 und 3 beginnen gleichzeitig mit einem Arbitrierungsversuch (1). Teilnehmer 2 verliert zum Zeitpunk (2), Teilnehmer 1 zum Zeitpunkt (3) das Buszugriffsrecht. Beide Teilnehmer gehen damit in den Empfangszustand; am Ende der Arbitrierungsphase (4) besitzt nur noch Teilnehmer 3 das Buszugriffsrecht. Nur dieser Teilnehmer sendet seine Nachricht über den Bus[11].

1.2.5. Leitungslänge & Übertragungsgeschwindigkeit

Die Übertragungsrate ist abhängig von der Leitungslänge[12].

Maßgeblich für die Berechnung der Leitungslänge, bzw. der Übertragungsgeschwindigkeit ist die Zeit, die ein Signal von einen Ende des Busses zum anderen braucht. Innerhalb dieser Zeit müssen die Teilnehmer auch auf den Fehlerfall reagieren können.

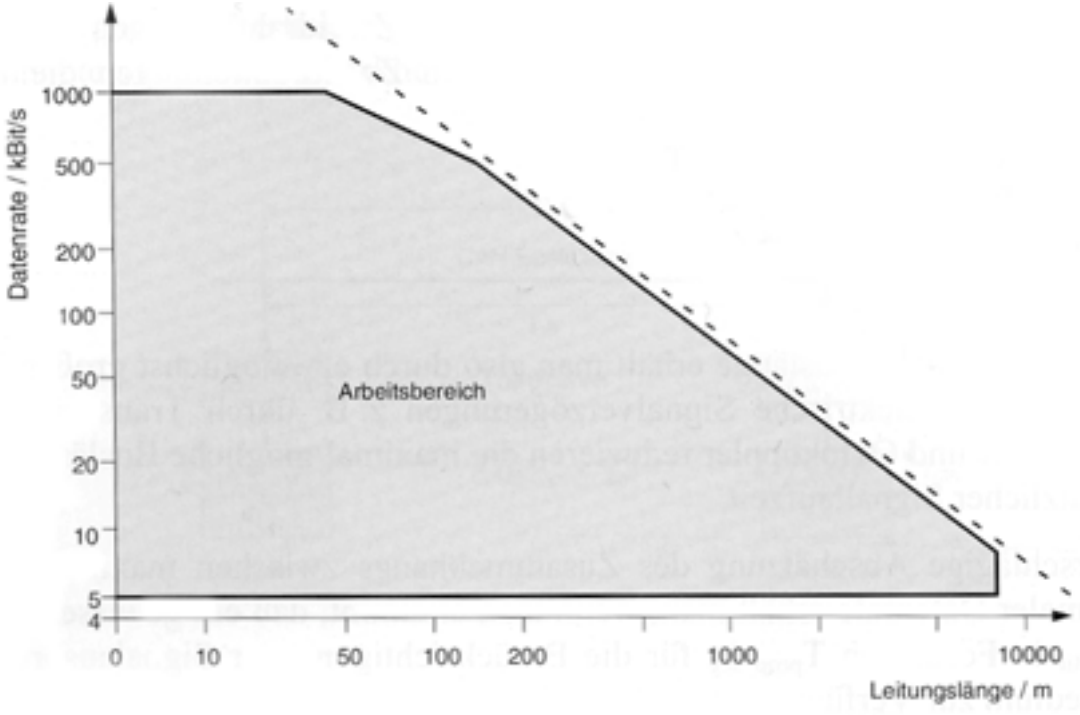
[10] Etschberger, CAN, S. 56.

[11] Etschberger, CAN, S. 56.

[12] Etschberger, CAN, S. 93.

Folgende Formel schätzt die Maximale Signalausdehnung ab. Dabei wird von einer Zweidrahtleitung mit der spezifischen Signalausbreitung vom 5 ns/m ausgegangen:

$$L_{\max} = \frac{\frac{x}{2 * Baudrate} - T_{el}}{5\frac{ns}{m}}$$ [13]



[14]

1.2.6. Objektidentifier

Die Adressierung der Datenpakete erfolgt nicht geräteorientiert, sondern kennzeichnet den Inhalt des Datenpaketes. Solche Inhalte können z.B. Temperatur, Druck oder Spannung sein, die jeweils einen eigenen Identifier haben. Die Empfänger entscheiden dann mit Hilfe des Identifiers, ob die Nachricht für sie relevant ist. Ein Teilnehmer kann beliebig viele Nachrichten senden und empfangen. Beim Senden ist es jedoch wichtig, dass kein anderer Teilnehmer Pakete mit dem gleichen Identifier sendet, da sonst die Arbitrierung nicht mehr funktioniert.
Je kleiner der Identifier ist, desto wichtiger ist die Nachricht.

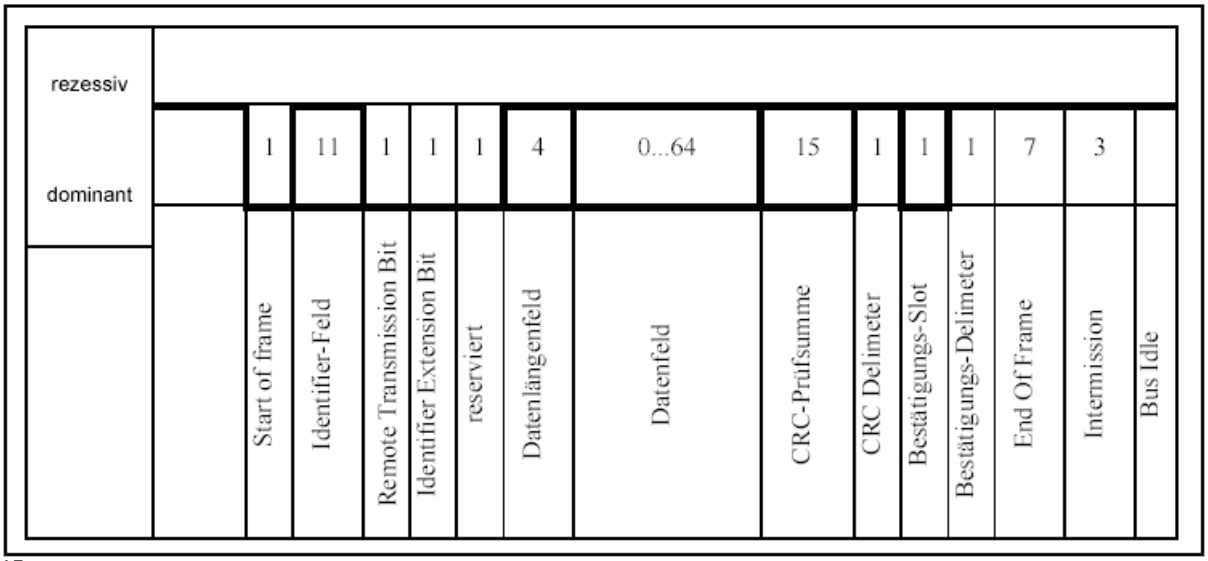
[13] Etschberger, CAN, S. 93.

[14] Etschberger, CAN, S. 94.

1.2.7. Nachrichtenformate

1.2.7.1. Datentelegramm (Data Frame) und Datenanforderungstelegramm (Remote Frame)

Im folgenden wird das „Base frame format“ beschrieben. Im Gegensatz zum „Extended frame format“ ist der Identifier 29 Bit lang und kann somit bis zu 512 Mio. Nachrichten unterscheiden. Die beiden Formate sind zueinander kompatibel und können sogar gleichzeitig in einem Netz existieren.



[15]

Start of frame: Dient zur Synchronisation der Busteilnehmer und besteht aus einem Dominaten Bit.

Identifier-Feld: Enthält den oben Beschriebenen Objektidentifier.

RTR: Benötigt ein Teilnehmer ein Objekt eines anderen Teilnehmers, das er zur Weiterverarbeitung benötigt, so setzt er das RTR auf rezessiv. Da dieses Feld bei der Arbitrierung mit einbezogen wird, hat ein Datentelegramm immer den Vorrang vor einem Datenanfoderungstelegramm.

IDB: Ist das Bit dominant, so folgen weitere Daten.

Datenängefeld: Hier wird die Länge der Nutzdaten angegeben. Da pro Telegram max. 8 Byte übertragen werden können, sind 4 Bits (1000 = 8) ausreichend, um die Länge der Daten anzugeben.

[15] http://de.wikipedia.org/wiki/Can-bus

Datenfeld: Enthält die eigentlichen Nutzdaten. Der Inhalt dieses Feldes ist nicht standardisiert.

CRC-Prüfsumme: Die nach obigen Verfahren berechnete CRC-Prüfsumme

CRC-Delimiter: Auf die Prüfsumme folgt ein rezessives Begrenzungsbit, das ihrerseits wieder als Prüfbit dient.

Bestätigungsslot: Wird das Paket korrekt empfangen wird hier ein dominantes Bit gesendet. Danach folgt wieder ein rezessives Begrenzungsbit.

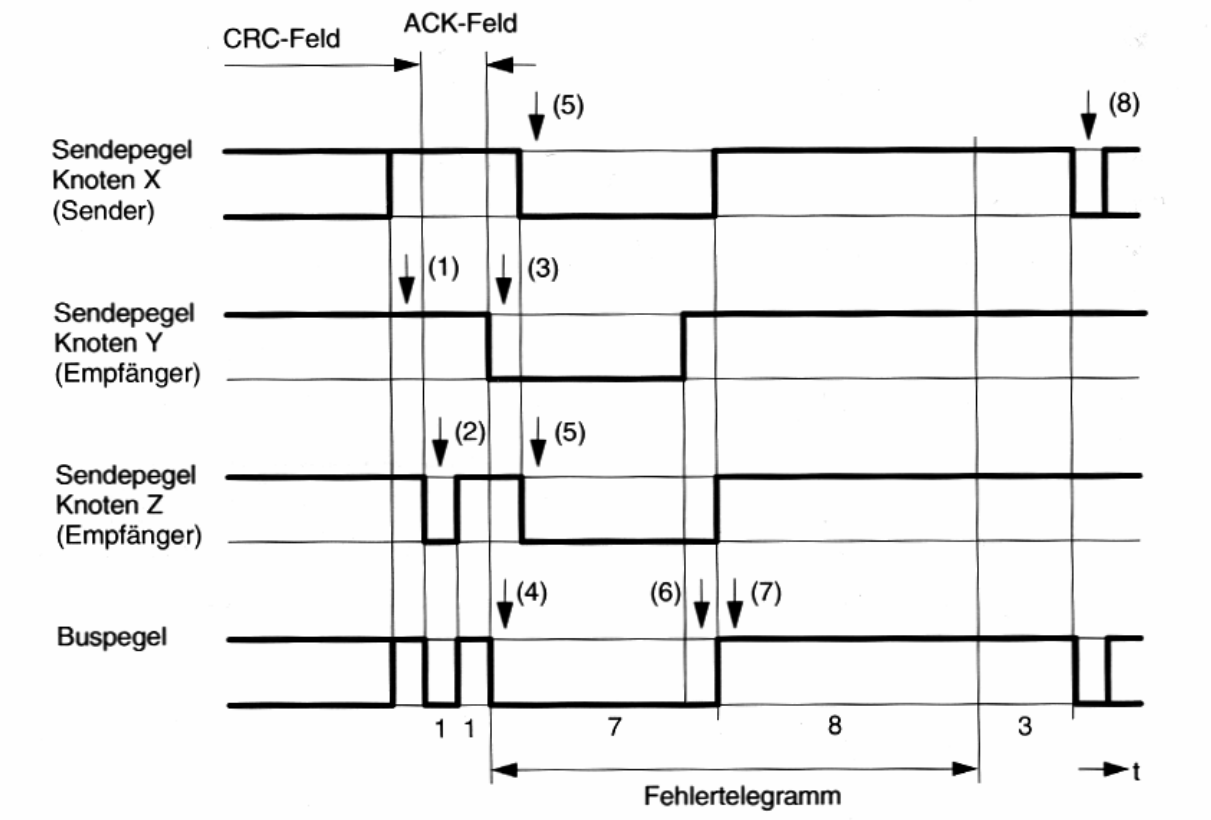
End of Frame: Das Telegramm wird durch 7 rezessive Bits abgeschlossen.

Die Unterscheidung zwischen dem „Base frame format“ und dem „Extended frame format“ erfolgt mit Hilfe des „Identifier Extension Bit“ und des darauffolgenden reservierten Bits. Im erweiterten Format ist der 29-Bit Identifier in zwei Teile aufgeteilt, nämlich den 11 Bit langen Basis-Identifier sowie einen 18 Bit langen „Extended Identifier“. Der Basis Identifier entspricht dem Identifier des Standardformates. Im Anschluss an den Basis-Identifier folgen im erweiterten Format als Teil des Arbitrierungsfeldes das „Substitute Remote Request Bit“ (SRR-Bit) und das „Identifier Extension Bit“.

Mit dem im Extended Format rezessiv übertragenen SRR-Bit wird das RTR-Bit des Standardformats ersetzt und bedeutet, daß ein im Standardformat übertragenes Datentelegramm prinzipiell höherprior ist als ein im erweiterten Format übertragenes Datentelegramm mit gleichem Basis-Identifier[16].

[16] Etschberger, CAN, S. 82.

1.2.7.2. Fehlertelegramm (Error Frame)



[17]

Knoten Y erkennt einen Fehler im Rahmen der CRC-Prüfung (1). Knoten Z hat die Nachricht fehlerfrei empfangen und bestätigt deshalb im ACK-Slot (2). Da der Bestätigungsmechanismus erhalten bleiben muß, signalisiert Knoten Y den erkannten Fehler erst nach dem ACK-Feld (3). Die Knoten X und Z erkennen zum Zeitpunkt (4) einen Formfehler, da sie das rezessive Endezeichen erwarten. Sie signalisieren den erkannten Fehler im nächsten Bit (5). Auf dem Bus entsteht somit ein insgesamt 7 Bit langes aktives Fehlertag. Knoten Y erkennt zum Zeitpunkt (6), dass er als erster den Fehler signalisiert hat. Nach Abschluss des 8-Bit langen Endekennzeichens sowie des 3 Bit langen Interframe Space kann der sendende Knoten mit einer erneuten Busarbitierung die Wiederholung der Nachrichtensendung beginnen.

1.2.7.3. Überlasttelegramm (Overload Frame)

Ein Überlasttelegramm kann als ein spezielles Fehlertelegramm aufgefasst werden. Das Überlastflag besteht aus 6 dominanten Bits und zerstört die festgelegte Form

[17] Etschberger, CAN, S. 66.

des Telegrammzwischenraums. Als Folge des Verzögerunsüberlastflags erkennen auch alle anderen Netzknoten eine Überlastbedingung (dominantes Bit im Telegrammzwischenraum) und senden ihrerseits ein Überlastflag. Ein Überlastflag endet wie ein Fehlertelegramm mit 8 rezessiven Bits[18].

[18] Etschberger, CAN, S. 67.

2. OSEK-COM[19]

2.1. Einleitung

OSEK steht für ein industrielles Standardisierungsgremium und bedeutet ausgeschrieben „Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug“.

Das 1993 gegründete Gremium besteht aus verschiedenen Kfz-Herstellern, deren Zulieferern und Software-Häusern. Gründungsmitglieder waren BMW AG, Daimler-Benz AG (heute DaimlerChrysler AG), Adam Opel AG, Volkswagen, Robert Bosch GmbH, Siemens AG und das Institut für industrielle Informationstechnik der Universität Karlsruhe (TH).

Im Jahr 1994 schloss man sich mit der 1988 gegründeten französischen VDX-Initiative bestehend aus PSA (Peugeot, Citroën) und Renault zusammen. Seitdem lautet die offizielle Bezeichnung OSEK/VDX. Die Gründungsmitglieder bilden heute das Steering Committee[20].

OSEK COM ist primär auf die Zusammenarbeit mit einem Betriebssystem ausgelegt, dass den OSEK OS Spezifikationen genügt. OSEK COM stellt Dienste bereit, um Nachrichten zwischen Prozessen (entweder Tasks und/ oder Interrupt-Service-Routinen) des OSEK OS auszutauschen. Diese Nachrichten werden an Sending-Message-Objects geschickt und von Receiving-Message-Objects empfangen.

Liegen die beteiligten Prozesse auf derselben Electronic Control Unit (ECU) spricht man von interner Kommunikation, ansonsten von externer. Die Übertragung erfolgt für die Prozesse in der Art, dass die Prozesse interner von externer Kommunikation nicht unterscheiden können.

OSEK COM bietet m:n Kommunikation an: 0-m Sender können Nachrichten an das selbe Message Object schicken, welches bei interner Kommunikation an 0-mehrere Receiving-Message-Objects und bei externer Kommunikation an 0-1 Receiving-Message-Object übermittelt werden.

Durch die Verwendung von OSEK COM in Verbindung mit den CAN-Bus, ist es möglich, größere Nachrichten, als die durch CAN maximal möglichen 8 Bytes zu senden, da von OSEK COM eine Nachrichtensegmentierung vorgesehen ist.

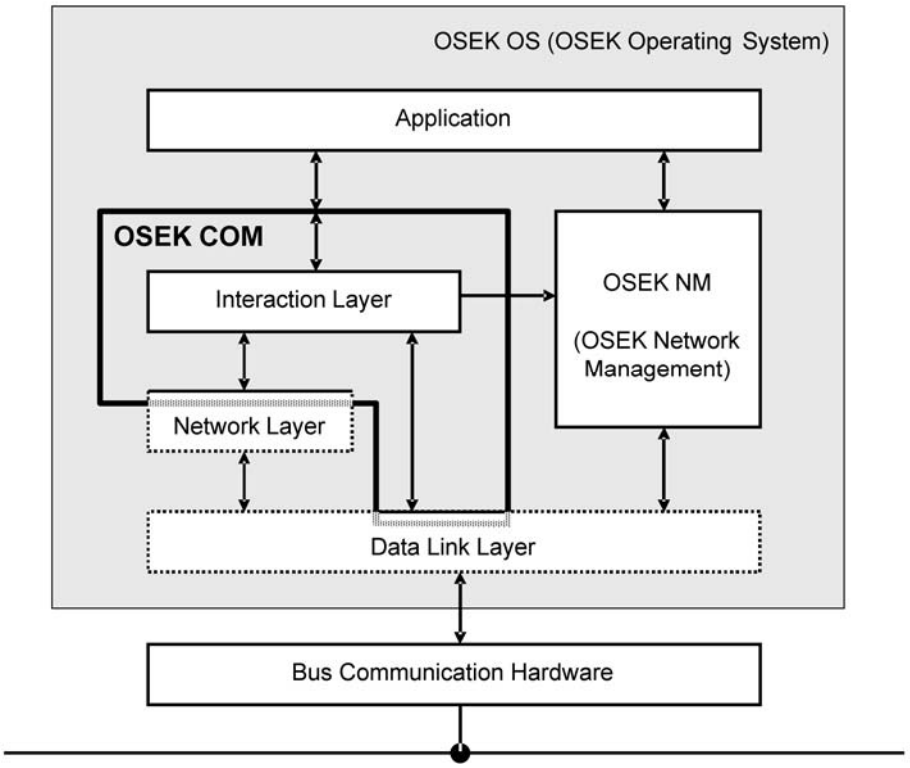
---

[19] Zu Kapitel 3 vgl. http://portal.osek-vdx.org/files/pdf/specs/osekcom303.pdf.

[20] http://de.wikipedia.org/wiki/OSEK

2.2. Die einzelnen Schichten

2.2.1. Schematische Übersicht



2.2.2. Interaction Layer

Der IL stellt die OSEK COM API dar, die Dienste für das Versenden und Empfangen von Nachrichten bereitstellt. Interne Kommunikation wird direkt vom IL geregelt.
Wird mit einem externen Teilnehmer kommuniziert, verpackt der IL eine oder mehrere Nachrichten in eine I-PDU und gibt sie an die darunterliegende Schicht weiter. Die IL kümmert sich auch um die Byte-Order der Pakete und konvertiert die Daten aus dem Netzwerk in das für die CPU gewünschte Format.
Message Objects können auf zwei Arten angelegt werden:

- Queued Message Objects
- Unqueued Message Objects

Der Unterschied besteht darin, das ankommenden Nachrichten bei den Queued MO in eine FIFO-Schlange gelegt werden und gelöscht werden, nachdem die Anwendung sie ausgelesen hat. Ist die Warteschlage voll, werden neu ankommenden Nachrichten gelöscht. Um die m:n Fähigkeit von OSEK COM nicht zu beeinträchtigen, wird für jeden Empfänger der Nachricht eine eigene Warteschlage angelegt, die dann jeweils separat entleert wird.

Bei den Unqueued Messages wird nur die jeweils aktuellste Nachricht gespeichert. Diese kann von den Empfängern beliebig oft gelesen werden. Trifft eine neue Nachricht ein, so überschreibt sie die alte Nachricht.
OSEK COM kümmert sich nicht um das Reservieren von Speicher für die Nachrichten. Das ist Sache der Anwendung.

2.2.3. Network Layer

Das Network Layer entspricht der Schicht 2 des ISO-OSI Referenzmodells. Es ist, abhängig vom eingesetzten Kommunikationsprotokoll für die Nachrichtensegmentierung und Zusammensetzung zuständig. Desweiteren bietet es empfangsbestätigtes Senden an und Flußkontrolle an. OSEK COM spezifiziert das Network Layer nicht näher weiter, sondern definiert nur eine Grundfunktionalität.

2.2.4. Data Link Layer

Auch das Data Link Layer wird von OSEK COM nicht spezifiziert. Es werden nur die minimalen Anforderungen an die Schicht genannt. Dazu gehört der unbestätigte Versand von einzelnen Paketen über das Netzwerk.

2.3. Deadline Monitoring

Das Deadline Monitoring ist auf die externe Kommunikation beschränkt. Der Sender kann hierbei überprüfen, ob der Versandwunsch eines Paketes durch die darunterliegende Schicht innerhalb eines bestimmen Zeitfensters geschieht.
Der Empfänger kann überprüfen, ob periodisch verschickte Nachrichten innerhalb eines festgesetzten Zeitpunktes bei ihm eintreffen.

2.4. Empfangen von Nachrichten

Beim Empfang von externen Nachrichten wird die PDU von der darunter liegenden Schicht empfangen. Ist die PDU fehlerhaft, wird je nach Konfiguration ein „Message Reception Error“ ausgeben und das Paket verworfen, ohne es an den IL auszuliefern. Ist das Paket fehlerfrei wird das PDU in ein sogenanntes I-PDU kopiert und ein „I-PDU Callout“ ausgelöst. Jede weitere Operation wird nun für jede einzelne Nachricht separat durchgeführt. I-PDUs, die leere Nachrichten enthalten, werden als letztes behandelt.

Mit Hilfe des Deadline Monitorings kann überprüft werden, ob die Nachricht rechtzeitig eingetroffen ist und im Fehlerfall einen „Message Reception Error“ weitergeben.
Danach werden die Nachrichten aus den I-PDU in ein Message-Objekt ausgepackt und gegebenenfalls die Endianess angepasst.
Wird die Nachricht nur intern Verschickt, beginnt der Versand an dieser Stelle. Die Nachricht kann noch ausgefiltert werden, um der Applikation nicht unnötige Pakete zukommen zu lassen. Leere Nachrichten werden generell nicht ausgefiltert. Die Filterung wird anhand des Objektidentifiers durchgeführt.
Im Anschluss wird eine „Message Reception“ Benachrichtigung ausgegeben. Die Anwendung kann die Nachricht dann mit Hilfe des API-Aufrufs „ReceiveMessage“ aus den Queued oder Unqueued Message Objects abholen.
Bei interner Kommunikation werden die Nachrichten direkt an die Reception Objects weitergegeben

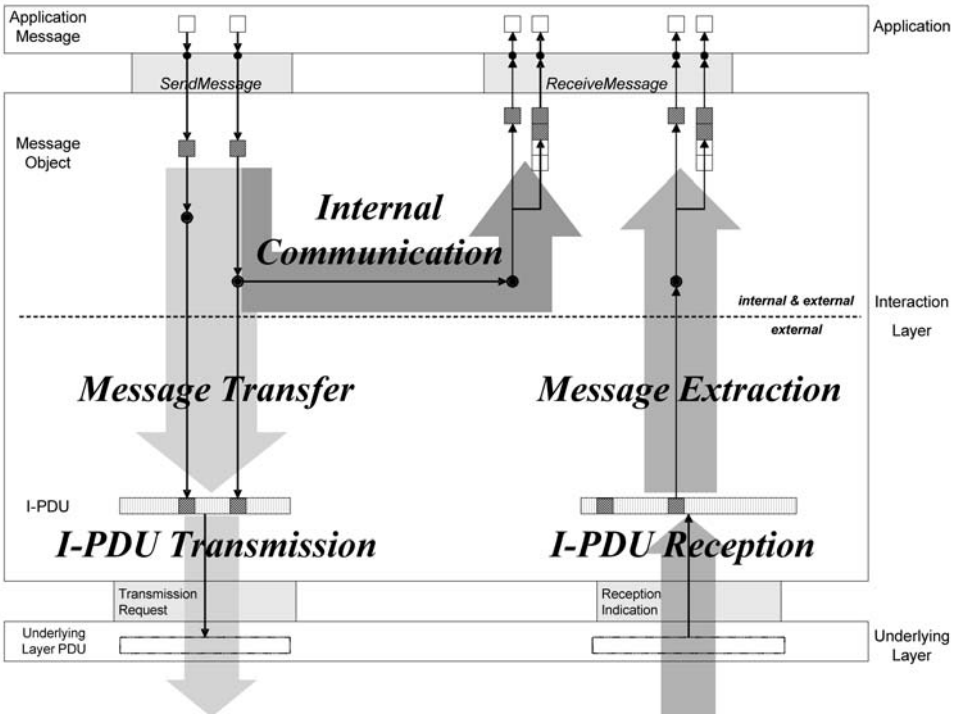
2.5. Senden von Nachrichten

Beim Senden von Nachrichten ruft die Applikation die API-Funktion „SendMessage“ auf, die daraufhin die Nachricht in ein Message Object kopiert.
Im Falle der internen Kommunikation wird dieses Message Object direkt im IL an den Empfänger weitergegeben.
Im nächsten Schritt wird eine Filterung des Pakets durchgeführt. Auch hier werden leere Nachrichten nicht ausgefiltert. Gegebenenfalls wird noch die Byte-Order angepaßt und die Nachricht dann in eine I-PDU verpackt.
Als Transfer Property wird entweder „Triggered“ oder „Pending“ gewählt. Ist „Pending“ gewählt worden, wird das Pakte nicht versandt, bei „Triggered“ wird die I-PDU an die darunterliegende Schicht zu, Versand weitergegeben. Des Weiteren stehen folgende Transmission Modes zur Verfügung:

- Direct: In Verbindung mit dem Triggered Transfer Property wird das Paket direkt versandt.
- Periodic: Beim Periodischen Versand wird das Pending Transfer Property gewählt. Die I-PDU wird in regelmäßigen Abständen verschickt. Bei den Sende-Aufrufen der Applikation wird lediglich das I-PDU aktualisiert, hat aber keinen Einfluss auf den Versendezeitpunkt.

- Mixed: Hier werden die beiden Versandarten kombiniert. Die I-PDU wird nach dem Ablaufen eines konfigurierbaren Timers versand. Schickt die Anwendung eine Nachricht mit dem Triggered Transfer Property, wird die I-PDU sofort versandt und der Timer zurückgesetzt. Um nur die Daten der I-PDU zu aktualisieren, sendet die Applikation eine Nachricht mit dem Pending Transfer Property.

Folgende Grafik zeigt den Versand und Empfang von Nachrichten schematisch:



3. Literaturverzeichnis

- Etschberger, Konrad (Hg.), *Controller-Area-Network*. *Grundlagen, Protokolle, Bausteine, Anwendungen*, München^{2}, 2000.
- Lawrenz, Wolfhard (Hg.), *CAN. Grundlagen und Praxis*, Heidelberg^{2}, 1997.
- Scherff, Birgit/ Haese, Erwin/Wenzek, Hagen R., *Feldbussysteme in der Praxis. Ein Leitfaden für den Anwender*, Berlin 1999.

<http://de.wikipedia.org/wiki/OSEK>

<http://de.wikipedia.org/wiki/Can-bus>

< http://www.can-wiki.info/>
<http://de.wikipedia.org/wiki/Cyclic_Redundancy_Check>
<http://portal.osek-vdx.org/files/pdf/specs/osekcom303.pdf>