IT 09 047
Examensarbete 30 hp
Oktober 2009

Teknologier för fordonsdiagnostik

Andrée Bylund

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Technologies for Vehicle Diagnostics

Andrée Bylund

The capacity to extract information from vehicles has the potential to be very beneficial. Performing analysis on information about fuel usage, emission values and other driving properties of a vehicle can lead to great economic and environmental benefits.

This report contains descriptions of two prominent systems for retrieving information of this nature from a vehicle, On-Board Diagnostics and FMS-Standard, and of an implementation of FMS-Standard.

The conclusion drawn after this investigation is that these systems do indeed offer access to the benefits mentioned earlier, although to varying degrees and with differing prioritisation.

Handledare: Anders Svedberg
Ämnesgranskare: Olle Gällmo
Examinator: Anders Jansson
IT 09 047
Tryckt av: Reprocentralen ITC

SAMMANFATTNING

Ett fordon har en stor mängd egenskaper som kan vara av intresse för människor i dess närhet. Hastighet och bränslenivå tillhör de av egenskaperna med mest uppenbara användningsområden men det finns många fler egenskaper som kan vara av värde att känna till.

Att kunna veta när det är fel på ett fordon är till stor nytta för dess ägare. Det finns naturligtvis flera tillvägagångssätt att upptäcka detta, men det sätt som har störst potential är att använda en dator för att övervaka fordonet. Sensorer finns utplacerade på många platser i fordonets system och rapporterar värden som man kan använda för att avgöra om någonting är fel med fordonet, och ofta även vad. Men inte bara det, informationen kan användas till även annat; att jämföra olika fordons eller förare mot varandra kan leda till värdefulla slutsatser.

Felsökning och övervakning av ett fordon kallas gemensamt för fordonsdiagnostik. Detta examensarbete har undersökt två ledande tekniker för detta: On-Board Diagnostics och FMS-standard, som används i personbilar respektive lastbilsklassade fordon.

On-Board Diagnostics, OBD, används automatiskt i personbilar för att kontrollera att dess utsläpp inte överstiger de lagstadgade gränserna och för att övervaka att fordonet fungerar som det bör. OBD har även potential att rapportera realtidsvärden som kan användas till ytterligare analys av fordonet och även dess förare.

FMS-Standard rapporterar värden för ett antal standardiserade egenskaper som främst är menade att ge en möjlighet att analysera fordonets och dess förares beteenden.

I examensarbetet har även ingått en implementation av FMS-standard. Ett programbibliotek, FMSlib, har konstruerats tillsammans med en simuleringsmiljö för att kunna testa det.

Denna rapport innehåller en detaljerad presentation av de två systemen och den programvara som utvecklats under arbetets gång.

Kortfattat är resultatet av analysen av dessa system att det finns system för fordonsdiagnostik, det går att använda dem och det finns anledningar att göra det.

Innehållsförteckning

1 INLEDNING

Detta är ett 30 högskolepoängs (20 veckors) examensarbete inom Datavetenskapligt program som utförts vid Uppsala Universitet i samarbete med CC Systems.

1.1 CC Systems

CC Systems [64] är ett företag som utvecklar och levererar avancerade styrsystem för mobila maskiner som verkar i tuffa miljöer. CC Systems har en produktserie med robusta fordonsdatorer. Dessa använder populära operativsystem som Windows XP, Windows CE och Linux. CC Systems ser ett ökat intresse av att kunna läsa ut information från fordonen för att sedan göra analyser på detta.

1.2 Problemformulering

Företag som arbetar inom områden som involverar fordon har ett intresse av att kunna övervaka detaljer i fordonets interna system. Dels för att kunna följa upp t.ex. bränsleförbrukning och dels för att övervaka hur komponenter slits. Samma intresse kan även finnas hos privatpersoner.

I dagsläget finns teknik som tillåter sådan övervakning. Det finns sensorer som kan mäta tryck, temperatur och annan information av intresse. Dessa sensorer används egentligen internt av fordonets egna system. Ett sådant system får in värden från sina sensorer som den anpassar sig efter. Exempel på sådana system är bränsleinsprutning, ABS-bromsar och klimatanläggning.

Intresse finns alltså att även utifrån få tillgång till en del av den information som de interna systemen samlar in. Med hjälp av den kan man hitta fel eller andra intressanta beteenden hos fordonet som de interna systemen inte är programmerade att söka efter.

Problemet kan sammanfattas till en grupp frågor:

1. Varför vill man kommunicera med personbilar och andra fordon?
2. Vad behöver man för att kunna göra detta?
3. Vilka tekniker finns inom detta område?
4. Hur ser dessa tekniker ut?
5. Hur använder man dem?
6. Vad kan förmedlas av varje teknik?
7. Vad har teknikerna för positiva egenskaper man vill komma åt?
8. Vad har teknikerna för negativa egenskaper man gärna undviker?

Dessa frågor försöker examensarbetet som helhet att finna svar på, en kort sammanfattning av svaren på de frågorna finns i stycke 6.4.3.

1.3 Mål och syfte

Det som vill uppnås med detta examensarbete är:

- att ge svar på de frågor som ställs i 1.2.
- att skapa mjukvara som kan vara användbar även utanför examensarbetets ramar.
- att ”ge den studerande träning i att planera, genomföra och redovisa ett vetenskapligt arbete samt att förmedla kontakt med forskning och utvecklingsarbete”[68].

Den information som examensarbetet avser sammanställa finns vid examensarbetets start inte att tillgå i en lättillgänglig form.

1.4 Tillvägagångssätt

Två metoder används för att uppfylla målen för examensarbetet.

Den första metoden är en teoretisk undersökning inom området datorassisterad fordonsdiagnostik. Det finns två teknologier som kommer att ha fokus: FMS-Standard [34] och On-Board Diagnostics (OBD)[6]. Dessa två kommer att undersökas, beskrivas och analyseras. Denna analys innefattar att jämföra systemen mot varandra. Eventuella andra teknologier inom samma område kommer också att undersökas, om tiden räcker till.

Den andra metoden är att implementera ett system som använder sig av en av dessa teknologier. Den teknologi som skall implementeras är FMS-Standard. Implementationen kommer att vara i form av ett programbibliotek med den kringmiljö som krävs för att kunna testa det.

1.5 Avgränsning

En implementation av OBD [6] utförs inte eftersom tiden inte kommer att räcka till.
Andra standarder än FMS-standard [34] och OBD [6] undersöks inte annat än mycket ytligt.

2 BAKGRUND

Här presenteras kort en del av de tekniker som finns i anslutning till datorassisterad fordonsdiagnostik.

2.1 Datorkontrollerade fordon

Numera finns det datorer nästan överallt, inklusive i fordon. De datorer som avses här är små, integrerade datorer som kontrollerar ett delsystem i fordonet. En sådan dator kallas ”Electronic Control Unit”, ECU, till skillnad från ”fordonsdator”. Med fordonsdator menas en ”vanlig” dator som finns fysiskt placerat i eller på fordonet och som interagerar med den. Den ursprungliga anledningen till att man började använda ECU:er i fordon var att dessa kan kontrollera undersystem i ett fordon med en responstid och precision som är bättre än både ”oövervakade” elektriska eller mekaniska system och system som kräver interaktion med föraren.

Ett exempel är bränsleinsprutning: För att få optimal förbränning skall förhållandet av luft och bensin vara ett visst bestämt värde. En ECU kan med sensorer mäta om det är exakt så, eller om det av någon anledning är fel fördelning, och i sådant fall kompensera för detta. Ett mindre avancerat system skulle inte kunna kompensera för små fel och mänsklig interaktion skulle vara för långsamt för att ha någon inverkan.

När en ECU samlat in all denna information kan informationen utan större problem användas till även annat än att kontrollera fordonet. Den kan analyseras och presenteras för förare och andra människor i kontakt med fordonet. Detta är vad som menas med ”fordonsdiagnostik”.

2.2 Kommunikationsnätverk i fordon

Det system som kontrollerar ett fordon består av flera olika komponenter (sensorer och ECU:er) som är anslutna till varandra i ett kommunikationsnätverk. Liksom andra sorters kommunikationsnätverk (som till exempel Internet eller telekom-nätverk) så har fordonsnätverk speciella krav på sig.

De krav som finns på fordonsnätverk är att de ska vara billiga att producera och att de ska vara tåliga mot yttre påverkan, både från elektromagnetisk störning och från de mer fysiska bekymmer som ett fordon kan råka ut för såsom vatten, värme och salt.

Av mindre vikt är bandbredd och arkitektur. De flesta ECU:er skickar en ganska liten mängd information, men den information som sänds förväntas komma fram. Ett fordonsnätverk består av ett fast antal noder. Det är normalt inte så att noder ansluter till eller lämnar nätverket medan det används, vilket leder till att nätverksarkitekturen kan vara enkel och ha låg utökbarhet.

2.3 Fleet management system

Fleet management system, FMS, är ett samlingsnamn för system som används för att styra och övervaka en vagnpark, en ”fordonsflotta”. System av den typen är uppbyggda av flera delar: en

central kontrollpunkt och ett flertal fordon som är utrustade med teknik för att kommunicera med denna centrala punkt samt med teknik för att samla in informationen som man vill övervaka.

Anledningen att FMS tas upp är att det är en användare av fordonsdiagnostik. I ett FMS är de tjänster som erbjuds av fordonsdiagnostik till mycket stor nytta.

Det man vill åstadkomma med ett FMS är att få en överblicksbild av sin vagnpark. Man vill kunna jämföra exempelvis bränsleförbrukning mellan olika fordon för att kunna hitta avvikande beteenden. Om ett visst fordon använder avsevärt mer bränsle än de övriga så vill man känna till detta så att man kan undersöka orsaken.

Ett FMS erbjuder vanligtvis även andra fördelar. Att kunna skicka textmeddelanden direkt till föraren över mobiltelefonnätet istället för att ringa är billigare och dessutom kan meddelandet automatiskt lagras i förarens fordonsdator. Via ett sådant system kan även automatisk orderhantering ske.
Rent praktiskt så sker informationsöverföringen från fordon till kontrollpunkt oftast antingen trådlöst eller vid regelbundna servicetillfällen.

Ett FMS kan också ha sina fordon utrustade med GPS (Global Positioning System)[2], som gör att systemet kan hålla sig uppdaterat med var fordonen befinner sig någonstans. Med den informationen kan man då göra bra val när man skall välja hur man delar ut uppdrag mellan de olika fordonen. Exempel: En taxifirma som hela tiden vet var alla bilar är, vart de ska och om de är lediga kan effektivt fördela körningar mellan bilarna.

2.4 Om standarder och varför de är bra

Ett kommunikationsnätverk som är avsett att användas på insidan av ett fordon är väldigt annorlunda från ett “vanligt” nätverk. Det är skapat med en specifik tillämpning i åtanke, vilket betyder att man kan i högre grad skräddarsy systemet för att ge väldigt god prestanda inom ett specifikt område. Men det leder också till bekymmer. Att bygga ett eget system för att använda i sina fordon är ett visserligen ett sätt att maximera prestandan i produkten, men det kräver också en stor mängd resurser. Att designa, konstruera och verifiera någonting så komplicerat som ett kommunikationssystem är ingen lätt uppgift [1]. Detta är en anledning till att fordonstillverkare inte vill göra detta själva. En annan potentiell nackdel med att skapa sin egen standard är att de produkter man skapar blir bundna till att bara kunna användas i tillverkarens egen miljö. Då behöver man själv konstruera och underhålla varje del av systemet och dess kringmiljö. Att göra på det sättet är väldigt riskfyllt, eftersom marknaden mycket väl kan tänka sig ignorera produkten helt, vilket leder till att resurserna spenderade på att utveckla systemet blir bortkastade. Risken att detta händer är större om produktutvecklaren inte redan är väl etablerad inom området.

Ett sätt att lösa detta på är att flera företag inom branschen tillsammans utvecklar systemet. Kostnaderna för utveckling och underhåll av systemet kan då för den enskilda parten bli bara en bråkdel av vad den annars skulle vara. En ytterligare fördel med att delta i ett sådant samarbete är att om den utförs av tillräckligt stor del av det aktuella marknadsområdet så stängs effektivt de aktörer som inte medverkar ute från att ta del i den framtida markanden.

2.5 Kollisionshantering

I ett nätverk händer det ibland att flera noder försöker sända ett meddelande samtidigt. Om nätverket består av en buss (som är en nätverksarkitektur där alla noder i nätverket kommunicerar över ett gemensamt medium) kommer deras meddelanden då att blandas, och mottagande noder kan inte veta hur meddelandena var menade att se ut. Det finns flera olika metoder att hantera detta problem, med olika fördelar och krav, några av dem presenteras kort här:

Anledningen att detta område tas upp är att de diagnostiksystem som har undersökts använder system där kollisioner kan förekomma.

2.5.1 Ingen kollisionshantering

Om kollisioner på nätverksnivå är ovanliga och/eller har låg betydelse kan man välja att inte hantera dem på denna nivå. Det betyder att risken för kollision finns, och att någon annan del av systemet måste kunna kompensera för detta.

2.5.2 Token Ring / Bus Master

Token ring är en metod för att undvika kollisioner och bus master är en variant av token ring. I token ring har en nod i taget rätt att sända meddelanden, och den rättigheten flyttas runt mellan noderna enligt en cirkulär lista.. Om en bus master används så är en av nätverksnoderna chef över de andra, det är denna nod som delar ut sändrättigheten till de andra noderna och så fort ett meddelande är sänt återgår sändrättigheten till den ursprungliga chefnoden.

2.5.3 Carrier Sense / Multiple Access (CSMA)

Denna metod går ut på att noderna lyssnar efter inkommande meddelande innan de själva börjar sända [3]. Ifall två noder börjar sända samtidigt uppstår dock fortfarande kollisioner. Det finns dock mer sofistikerade varianter av detta där noderna lyssnar på inkommande meddelanden samtidigt som de sänder, och för varje bit jämförs dessa två. Om dessa skiljer sig har man upptäckt ett fel, som i sin tur kan hanteras på olika sätt:

2.5.3.1 Carrier Sense/Multiple Access with Collision Detection (CSMA/CD)

När en nod upptäcker en kollision skickar den omedelbart ut ett felmeddelande på nätverket som informerar alla noder om att en kollision har upptäckts. Alla slutar då sända. Därefter väntar alla noder som försökte sända en slumpmässigt lång tid och försöker sedan igen.

2.5.3.2 Carrier Sense/Multiple Access with Collision Resolution(CSMA/CR)

Kallas även Carrier Sense/Multiple Access with Bitwise Arbitration (CSMA/BA). Denna metod innebär att noder slutar sända så fort de upptäcker att någon annan sänder. Om detta kan göras tillräckligt fort, dvs. om propageringstiden i förhållande till bitfrekvensen är tillräckligt låg, kommer inte den andra nodens meddelande att riskera att förstöras. I kombination med att skicka med ett prioritetsvärde och sändarens (unika) adress leder detta till att viktiga meddelanden alltid kommer fram i tid.

2.5.3.2.1 Non-Return-to-Zero (NRZ)

NRZ är ett exempel på en teknik som använder CSMA/CR. NRZ använder två spänningsnivåer, varav ingen är noll (att spänningen är noll och att spänningen representerar bitvärdet noll är olika saker). Det betyder att varje nod i systemet alltid sänder någonting, antingen den högre spänningen eller den lägre.

NRZ är ett system med ”lossless arbitration”, dvs. om två noder börjar sända samtidigt så blir resultatet att ett av meddelandena kommer fram, till skillnad från exempelvis Ethernet där bägge meddelandena förstörs och inget av dem kommer fram i läsbart skick. Detta fungerar genom att varje nod som sänder på bussen samtidigt lyssnar på den. Om den sänder ett värde och bussen visar ett annat värde så kan noden sluta sig till att någon annan sänder samtidigt.

Eftersom en låg spänning ”skrivs över” av en hög kommer den höga spänningen att vinna vid en konflikt. En nod som förlorar en sändningstvist kommer att övergå till att endast lyssna och när meddelandet sänts klart försöka skicka sitt meddelande igen. När en nod inte har ett meddelande att sända sänder noden den låga spänningen.

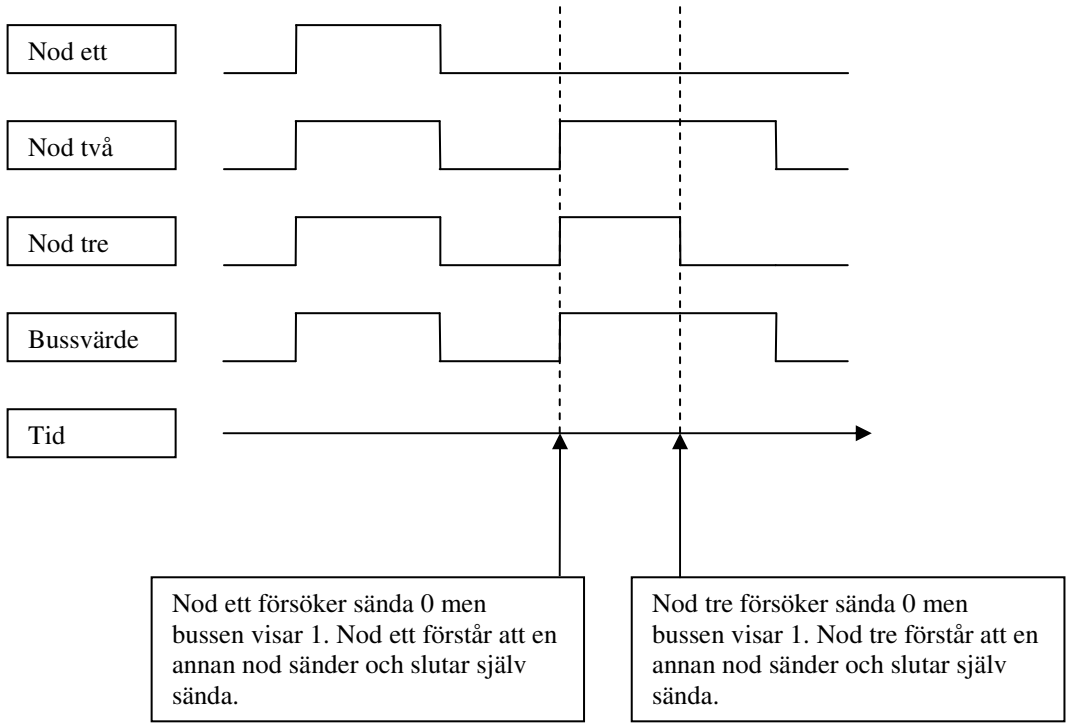

Bild 1. NRZ-arbitration. Tre noder försöker sända men två avbryter när de upptäcker konkurrerande meddelanden med högre prioritet.

2.6 OSI-modellen

OSI-modellen är skapad av internationella standardiseringsorganisationen**,** ISO, och finns fullständigt beskriven i [4]. Den är till för att tydliggöra hur kommunikation mellan program i olika datorer anslutna genom ett nätverk är strukturerad.

Modellen delar upp ansvaret för de olika uppgifterna inom kommunikation mellan olika ”lager”. Det finns 7 lager totalt, se bild 2. Varje lager anses logiskt kommunicera med motsvarande lager i en annan dator (eller flera andra datorer), genom att använda sig av lagret under och för att den fått instruktioner från lagret ovanför. Ett lager kan anta att alla egenskaper

som erbjuds av de undre lagren finns och att de egenskaper som hanteras av de övre lagren inte behövs. Ett visst lager behöver därför enbart fokusera på de uppgifter det har ansvar för.

Vinsten av att dela upp uppgifterna på detta sätt är att man får en bättre överblick hur ett kommunikationssystem ser ut. Detta leder bl.a. till att färre problem uppkommer under utveckling och drift av ett kommunikationssystem och att när de väl gör det är det lättare att hitta orsaken och lösningen till dem.

Ett kommunikationsprotokoll är en uppsättning regler för hur kommunikation mellan två lager på samma nivå skall gå till. Om två datorer använder samma protokoll för varje lager (och är ihopkopplade) kan program som kör på dem kommunicera med varandra.

En fördelaktig följd av OSI-modellen är att det finns publikt tillgängliga protokoll för varje av dessa lager. Att använda ett sådant är ofta ekonomiskt och tidsmässigt fördelaktigt jämfört mot att utveckla ett eget.

Lager 7	Applikationslager
Lager 6	Presentationslager
Lager 5	Sessionslager
Lager 4	Transportlager
Lager 3	Nätverkslager
Lager 2	Datalänklager
Lager 1	Fysiskt lager

Bild 2: OSI-modellen

1. Fysiskt lager. För att kommunicera mellan två datorer krävs en fysisk anslutning mellan dem. Ett enkelt exempel på en fysisk anslutning är en elektriskt ledande metalltråd som har egenskapen att om en dator applicerar en spänning på metalltråden kan en annan ansluten dator datorn känna det. Det fysiska lagret beskriver den anslutningen och vilka gränsvärden och andra parametrar som skall gälla för den.

2. Datalänklager använder det fysiska lagret för att skicka data. Den bestämmer när en bit skall sändas och har därmed kontroll över överföringshastighet och kollisionshantering. Data finns här hoppackade i ett kort meddelande, kallat ram, som kan innehålla kontolldata för prioritet och felhantering.

3. Nätverkslagrets uppgift är att veta hur de olika datorerna på nätverket är ihopkopplade, och vart viss data skall skickas för att komma till rätt mottagare.

4. Transportlagret hanterar pålitligheten hos protokollet. Om garanterad leverans av meddelanden är del av protokollet så hanteras det här. Transportlagret bokför i sådant fall vilka meddelanden den skickat och förväntar sig att meddelas när de kommer fram

oskadda. Transportlagret kan även begränsa sin arbetshastighet för att andra lager skall hinna med.

5. Sessionslagret hanterar ”anslutningar”. Även om två enheter är ihopkopplade med varandra är det inte alltid de vill vara tillgängliga. För att kommunicera behöver de först ha en anslutning mellan varandra. Sessionslagret upprättar, avslutar och övervakar sådana anslutningar. Ett viktigt specialfall är synkronisering av parallella anslutningar.

6. Presentationslagret hanterar formatering av data. Om applikationslagret använder data på en ”lättförståelig” men minnesineffektiv form kan presentationslagret omvandla det till något effektivare för transport. Det är här eventuell kryptering sker.

7. Applikationslagert kommunicerar med applikationen. Den erbjuder en specialiserad tjänst (som till exempel skicka en fil, överföra videodata eller skicka ett mail) med en hög nivå av abstraktion.

En nackdel med denna modell är att varje lager endast talar med sina omedelbara grannar. Data som skickas mellan två applikationer passerar därför genom varje lager på sändarsidan, och varje lager igen på mottagarsidan. Detta kan leda till dålig prestanda. Av denna anledning följs ofta inte denna modell exakt i praktiken, även om den använts i teorin.

3 TEKNOLOGIER FÖR FORDONSDIAGNOSTIK

Inom området fordonsdiagnostik finns flera teknologier. Fokus här kommer att ligga på OBD [6] och FMS-standard [34] samt de tekniker som dessa två bygger på. FMS-standard och dess underliggande teknologi kommer vara mer detaljerad eftersom den även kommer att implementeras.

On-Board Diagnostics
On-Board Diagnostics, OBD, används i personbilar för att övervaka att bilen fungerar väl och att bilens utsläpp av föroreningar ligger under bestämda gränser.

FMS-Standard
FMS-Standard används i lastbilar för att samla in information från lastbilens interna system. Den bygger på en standard från Society of Automotive Engineers (SAE) som kallas J1939.

J1939
J1939 [36] är ett kommunikationsprotokoll som är designat för att utgöra lastbilars interna system. Denna standard använder CAN.

CAN
Ett system som används för intern kommunikation i lastbilar delsystem heter Controller Area Network (CAN) och finns definierat i ISO-11898 [29][30][31][32][33]. Det är ett lågnivåsystem som är till för att användas i andra system med högre abstraktionsnivå.

3.1 On-Board Diagnostics

On-Board Diagnostics, OBD, är ett system för att övervaka personbilars utsläpp och korrekta funktion. Systemet har skapats och används för att myndigheter önskar reducera mängden utsläpp, emission, som personbilar står för. Detta görs genom att man monterar övervakningssystem på fordonen, som varnar föraren om fordonets utsläppsvärden är högre än de gränsvärden som finns inprogrammerade i systemet. Nya fordon släpper inte ut så pass mycket föroreningar att systemet ger utslag men när fordon åldras ökar utsläppen. Bilar utsätts för ganska stort slitage av att användas och det finns många delar i systemet som potentiellt kan gå sönder och orsaka att utsläppen ökar. OBD varnar när ett fordon har degraderats så långt att utsläppsnivåerna är högre än vad som är tillåtet.

Egentligen heter inte systemet exakt OBD, utan detta namn används som ett generellt samlingsnamn för en grupp system. De faktiska system som används har olika namn och lite olika funktionalitet i olika områden eftersom dessa har olika lagar. Det system som faktiskt används i USA heter OBD-II[6]och det som används i Europa heter EOBD[8].

Den tillåtna utsläppsnivån har sjunkit stadigt under en tidsperiod fram till nu, och kommer sannolikt att fortsätta sjunka. Detta har gjort att fordonen har behövts byggas mer miljövänligt och att betydelsen av emmisions-övervakande system som OBD har ökat.

OBD-systemet rapporterar sin status på två sätt:

- Till föraren via en fel-indikator; en lampa på instrumentbrädan som kallas Malfunction Indicator Lamp, MIL. Om OBD-systemet har upptäckt och verifierat ett fel kommer MIL att tändas, annars är MIL släckt.
- Via en diagnostikport som ger tillgång ett flertal ”tjänster”; bland annat att rapportera vilka fel som systemet har upptäckt.

Med jämna mellanrum genomförs inspektioner av alla personbilar, bland annat genom att fråga OBD-systemet om dess status. I Sverige är det Bilprovningen [65] som sköter detta.

En sidoeffekt av att övervaka utsläpp är att det kan vara ekonomiskt fördelaktigt för bilägare att veta när någonting i bilen fungerar sämre än det borde. En del fel som är relativt billiga att reparera kan orsaka att mer kostsamma delar av fordonet också går sönder om felet lämnas orört. Fel som får emmisionen att öka får ofta även bilens bränsleförbrukning att öka, något som direkt påverkar bilens driftkostnad.

3.1.1 Historia

Den första versionen av OBD, som senare har kommit att kallas OBD-1, skapades för att en myndighet i Kalifornien, California Air Research Board (CARB)[66], ville kunna övervaka hur mycket bilarna i delstaten förorenade luften. Alla bilar som såldes från år 1988 skulle vara utrustade med detta system. Problemet med OBD-1 var att det inte var tillräckligt väl specificerat. Bestämmelserna sade att vissa egenskaper måste övervakas, men de angav inte hur information skulle vara formaterad. Det ledde till att utseendet på informationen som kunde hämtas ut varierade mellan olika biltillverkare och ibland t.o.m. mellan olika bilmodeller.

Lösningen på detta var att skapa en ny version av OBD, kallad OBD-2. Denna version hade som mål att lappa ihop de hål som lämnats i den ursprungliga definitionen. Den lyckades mestadels med detta. OBD-2 definierar standardiserade koder för de fel som kan uppstå, vilka funktioner som skall finnas tillgängliga hos ett OBD-system, utseendet på diagnostikkontakten och applikationsnivå-protokollet. Däremot definierades inte hur den informationen skulle överföras (datalänk/fysiskt lager). En vidare utökning av funktionaliteten var att OBD-2 klarade att känna av komponenter som fungerade sämre än de borde, medan OBD-1 bara kunde känna av komponenter som inte fungerade alls. Detta system skulle finnas tillgängligt i alla bilar som såldes i Kalifornien år 1996. OBD-2 blev 1996 obligatoriskt i alla bilar som skulle säljas i USA. [7]

I Europa har inte intresset av att övervaka utsläpp varit lika starkt. EU har skapat en egen version av OBD-2 som kallas EOBD som trädde i kraft 2001 för bensindrivna bilar och 2003 för dieseldrivna bilar. EOBD är i princip identisk med OBD-2, men avviker på enstaka punkter.

En av nackdelarna med OBD-2 är att den har tillåtit flera olika lågnivåprotokoll. Detta har lett till att standarden blivit större och krångligare än vad den behöver bara. Från 2008 är dock det problemet mindre, åtminstone i USA och Europa, då nya bilar endast tillåts använda ISO 15765 [22].

Det finns diskussioner och förslag på hur en framtida standard, OBD-3, skulle kunna se ut. Den främsta utökningen i detta protokoll vore att låta OBD-systemen att rapportera upptäckta fel

direkt till den övervakande myndigheten via någon slags trådlös kommunikationsteknik. Detta skulle minska tiden som onödigt förorenande bilar används innan de repareras, och skulle ta bort behovet av årliga kontroller. Förslaget har mött en del motstånd eftersom många anser att det är ett intrång i personens privatliv som inte dessa myndigheter har rätt att göra.[8]

3.1.2 Metodik

OBD är ett system som är integrerat i personbilar. Det är en standardkomponent i bilar som monteras vid produktion. Ett OBD-system består av en eller flera ECU:er, sensorer kopplade till dessa ECU:er, en diagnostikkontakt samt en fel-indikations-lampa placerad på instrumentpanelen.

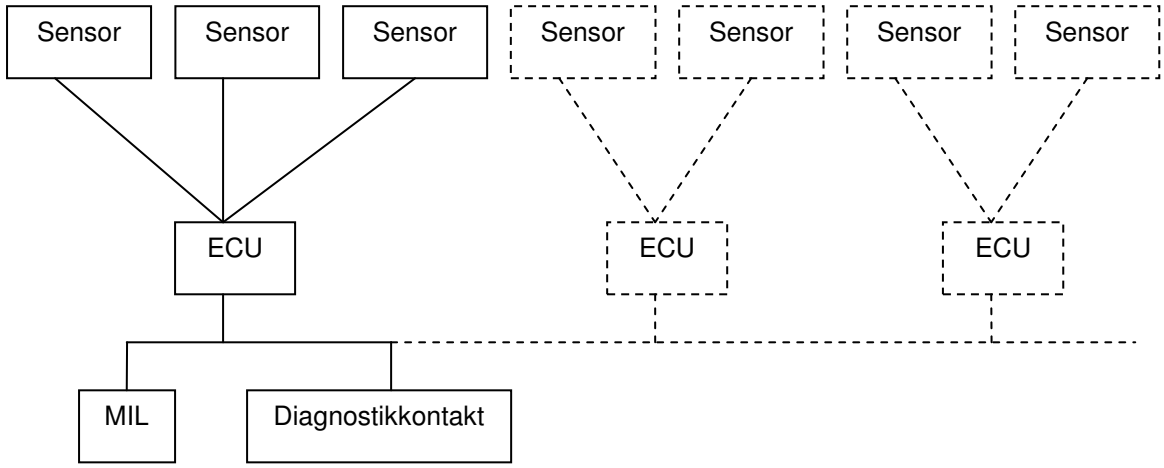

Bild 3. Komponenter i ett OBD-system

- ECU:
  En ECU samlar in information från sensorerna, informationen används (eventuellt efter viss beräkning) till att övervaka bilens utsläpp, och för att justera köregenskaper. Det kan finnas flera ECU:er på ett fordon, i sådant fall har de olika ansvarsområden. Exempelvis kan systemet bestå av en ECU som hanterar fordonets motor- och avgassystem och av en ECU som hanterar växling.

- Sensor:
  På utvalda platser i fordonet finns sensorer utplacerade. Varje sensor mäter en egenskap på en viss plats. Exempelvis finns det en sensor som mäter syrehalten som är placerad i avgassystemet framför katalysatorn. Om en ECU hanterar många egenskaper beror det på att den har många anslutna sensorer.

- Diagnostikkontakt:
  Via denna kan en användare koppla in sig i systemet för att få tillgång till den information som OBD-systemet erbjuder.

- Fel-indikations-lampa
  Denna lampa kallas MIL (Malfunction Indicator Lamp). Dess tillstånd indikerar huruvida OBD har upptäckt något fel. Den har tre lägen, släkt betyder att inget fel

hittats (och verifierats), tänd betyder att ett fel som ökar utsläpp har upptäckts och blinkande betyder att ett allvarligt fel som kan skada bilen har upptäckts.

Varje ECU övervakar den data de får in från sina sensorer. Om något värde avviker från det önskade försöker den att rätta till det. Om den inte lyckas ändra värdet kommer den att rapportera ett OBD-fel. Exempel: Det har uppstått en stor läcka mellan bränsletanken och motorn. Den ECU som kontrollerar bränsleinsprutning märker att det är för lite bränsle som når motorn, den beordrar då bränsletillförseln att öka. Ökningen ger ingen effekt; ECU:n rapporterar ett fel.

Ett fel enligt OBD är ett tillstånd hos det övervakade fordonet där utsläppen är högre än vad som är tillåtet.

Varje gång en ECU upptäcker ett fel så kommer den att tända MIL. Den kommer också att spara undan den felkod som beskriver felet. Tillsammans med felkoden sparar den även undan en ögonblicksbild (”freeze frame”) av vad vissa av fordonets köregenskaper var vid feltillfället. Ögonblicksbilden kan vara av stor nytta vid diagnostisering av felet.

Varje fel som övervakas av ett visst fordon har en kod som beskriver det. Dessa koder kallas Diagnostic Trouble Codes (DTC). Dessa är standardiserade (men det kan finnas även tillverkarspecifika) och beskrivs i [10][11]. Dessa koder är uppbyggda på ett specifikt sätt. De består av en bokstav följt av fyra siffror. Bokstaven beskriver vilket område av fordonet som felet härstammar från. Den första siffran beskriver om det är ett tillverkarspecifikt eller ett standardiserat fel, och de sista tre beskriver tillsammans vilket specifikt fel inom området det rör sig om.

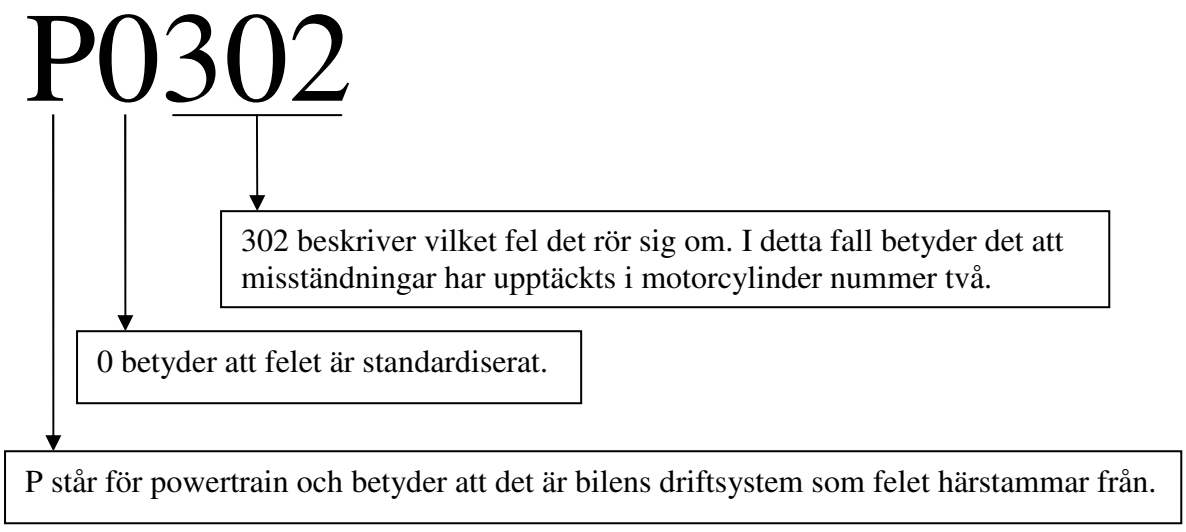

Bild 4. Exempel på OBD-felkod

OBD använder sig av s.k. ”readiness monitors”. En ”readiness monitor” används för att övervaka att fordonet har befunnit sig i ett visst tillstånd. En del av de värden som OBD övervakar kan bara observeras i vissa tillstånd. Därför behövs readiness monitors för att försäkra att systemet har haft en chans att kontrollera allt. En del av dessa monitorer beror på

andra monitorer, exempelvis kan bilen inte undersöka katalysatorns funktionsnivå om inte syresensorerna före och efter katalysatorn fungerar korrekt.

När man sedan skall kontrollera att bilen inte avger mer utsläpp än tillåtet så kontrollerar man ifall det finns några felkoder lagrade, och dessutom att alla readiness monitors är satta. Om systemet inte har kört alla tester kan man inte vara säker på att fordonet inte avger för höga utsläppshalter.

3.1.3 Händelseförlopp

När man vill hämta ut information från OBD-systemet måste man följa ett visst tillvägagångssätt. [12]

1. Användaren kopplar in sig mot diagnostikkontakten.
2. Användaren initierar en handskakningssekvens.
3. När den är klar kan användaren skicka en förfrågan om vilka egenskaper systemet stödjer för angiven tjänst.
4. OBD-systemet svarar med ett värde som indikerar vilka egenskaper den stödjer egenskaper.
5. Användaren skickar en begäran om att få värdet för en av de egenskaperna.
6. OBD-systemet svarar med värdet för den egenskapen.
7. Nu kan användaren välja att skicka en ny förfrågan eller begäran (dvs. hoppa till steg tre eller fem).
8. Om det steg 7 inte utförs inom en viss tid stängs anslutningen ner automatiskt.

3.1.4 Intern struktur

OBD-2 är beskrivet i ett flertal dokument utfärdade av SAE och ISO. Olika dokument beskriver olika delar av systemet och länkas samman genom referenser till varandra.

Hierarkin beskrivs i Bild 5 Förklaring finns i den efterföljande texten.

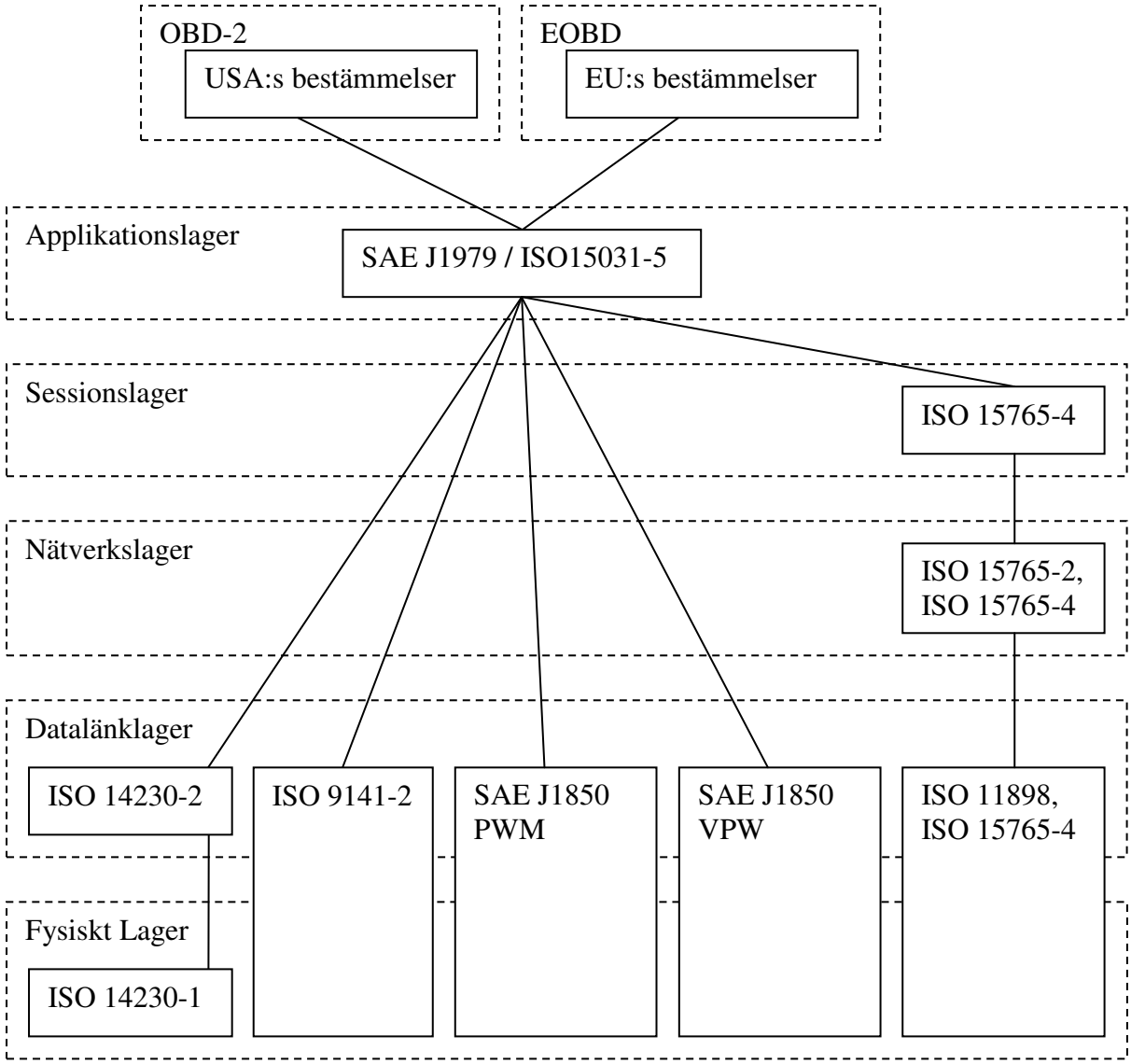

Bild 5. Dokumentationshierarkin för OBD

Högst upp i hierarkin finns de lagar och bestämmelser som är anledningen till att OBD skapats och används. Här ställs det krav som de underliggande dokumenten behöver möta.

I dokumenten J1979 [12] och ISO 15031-5 [9] beskrivs applikationsnivån för systemet. Dessa två dokument är olika men innehåller samma information. Här beskrivs vilka kommandon som en användarapplikation skall skicka för att interagera med OBD-systemet.

Under applikationsnivån finns det fem olika möjligheter. Detta kommer sig av att specifikationen ursprungligen inte definierade dessa lager, så olika biltillverkarna använde olika protokoll. De fem protokoll som idag erkänns har lagts till i standarden alltefter de blev populära. ISO 9141-2 [14], PWM [16][26] och VPW [16][26] är de som först lades till, ISO

14230 [17][18] har tillkommit i efterhand som en vidareutveckling på ISO 9141-2 och ISO 15765 [22][24] är ett ännu senare tillägg.

Själva diagnostikkontakten, som en användare kan ansluta till för att interagera med OBD, är standardiserad och beskriven i J1978 [25].

3.1.4.1 Applikationslager

När man skall interagera med OBD behöver man dels vara korrekt ansluten och dels skicka rätt frågor. I applikationslagret hanteras det senare. Det är i applikationslagret som man faktiskt får tillgång till den funktionalitet som OBD erbjuder. Applikationslagret beskrivs i J1979 [12] och ISO 15031-5 [9].

OBD-standarden erbjuder totalt tio stycken ”tjänster”(services). Tjänsterna har namn på formen $XY där X och Y är hexadecimala siffror. Denna namnkonvention används även för ID-nummer inom tjänsterna. Det kan utöver de standardiserade finnas tillverkarspecifika tjänster med nummer upp till $7F.

Vilka tjänster som finns standardiserade och deras funktionalitet varierar lite beroende på ifall ISO 15765, ofta benämnt CAN-OBD, används som protokoll för datalänk/fysiskt lager.

Exakt vilka tjänster, och vilka värden inom varje tjänst, som stöds av ett visst fordon är upp till tillverkaren. Tillverkare brukar dock se till att OBD-systemet i ett visst fordon är tillräckligt avancerat för att kunna möta de krav som finns i landet där fordonet avses säljas.

3.1.4.1.1 Service $01 – Request Current Powertrain Diagnostic Data

Denna tjänst används för att övervaka realtidsvärden hos fordonet. Man kan till exempel få reda på fordonets hastighet, bränslenivå och avgastemperatur.

Varje övervakad egenskap har tilldelats ett identifikationsnummer, Parameter ID (PID). PID går från $00 till $FF. För att hämta ut ett värde skall man först fråga vilka egenskaper som stöds genom att skicka en begäran efter PID $01. Om det PID man är intressead av stöds kan man sedan skicka en begäran efter det.

Det finns över 100 standardiserade värden som kan övervakas, men en ECU behöver inte stöda alla. Alla emmisionsrelaterade ECU:er behöver dock stödja PID $00, eftersom en begäran efter detta värde även fungerar som ett generellt kontrollmeddelande. Detta behövs eftersom kommunikationslänken avslutas om inget meddelande skickas under en viss tid. En ECU kan även ha stöd för tillverkarspecifika egenskaper.

3.1.4.1.2 Service $02 - Request Powertrain Freeze Frame Data

När en ECU upptäcker ett emmisionsrelaterat fel så kommer den att tända MIL, lagra en felkod och spara undan de körtidsdata som gäller för tillfället i en ögonblicksbild (”freeze frame”). Den information som finns lagrad i en ”freeze frame” kan man få tillgång till genom att använda den här tjänsten..

För att komma åt de data som finns används samma tillvägagångssätt som för att komma åt motsvarande data i Service $01. PID $00 svarar vilka värden som stöds, även om ingen ”freeze

frame” finns lagrad. PID $02 beskriver vilken DTC som orsakade att värdena sparades eller svarar med noll om inget fel inträffat.

Standarden kräver att systemet lagrar minst en ”freeze frame”, men det är tillåtet att använda fler än en om tillverkaren så önskar.

3.1.4.1.3 Service $03 - Request Emission-Related Diagnostic Trouble Codes

För varje fel som en ECU registrerar sparar den en felkod. Denna tjänst används för att hämta ut alla felkoder från en ECU. Innan den används skall systemet frågas om den har några koder (genom att använda Service $01 PID $01).

Endast felkoder för fel som verifierats kommer att rapporteras. För att ett fel skall klassas som verifierat behöver det ha observerats under flera resor.

3.1.4.1.4 Service $04 - Clear/Reset Emission-Related Diagnostic

Denna tjänst används för att nollställa OBD-systemet.

- Den tar bort alla felkoder, både verifierade och overifierade.
- Den tar bort all ”freeze frame”-information.
- Den nollställer ”readiness monitors”.
- Den tar bort resultat från icke-kontinuerligt övervakade system
- Den nollställer värden för körtid/körsträcka/uppstarter sedan nollställning
- Den nollställer värden för körtid/körsträcka med MIL påslagen.
- Den utför eventuell tillverkarspecifik nollställning.

3.1.4.1.5 Service $05 - Request Oxygen Sensor Monitoring Test Results

Denna tjänst ger tillgång till resultatet från systemets test av syresensorerna (även kallad lambdasensorer). Information går även att få fram genom att använda Service $06, och därmed används inte denna tjänst alltid.

Anledningen till att just dessa test fått en egen kategori kan vara att de är viktigare än andra test. Det är dessa sensorer som övervakar katalysatorns effektivitet, och det är katalysatorn som är den primära metoden att minska utsläpp.

Om man använder ISO 15765 som lågnivåprotokoll så används inte Service $05. Istället framförs informationen under Service $06.

3.1.4.1.6 Service $06 - Request On-Board Monitoring Test Results for Specific Monitored Systems

Denna tjänst ger tillgång till resultaten av tester som utförts på icke-kontinuerligt övervakade s system. Syftet med dessa tester är att övervaka utsläpp, och kontrollera att OBD-systemet fungerar korrekt.

Denna tjänst är strukturerad lite olika beroende på om ISO 15765 används eller inte. Om ISO 15765 används finns en del standardiserade tester, om inte så finns inga tester standardiserade.

Det kan finnas tillverkarspecifika tester. Information om dessa förmedlas av tillverkaren till reparatörer och bilägare på något sätt (exempelvis via Internet : [27]). Det finns en metod för att får reda på vilka tester som stöds, dock är den endast obligatorisk om ISO 15765 används. Denna metod är analog med den som används i Service $01.

Systemet bevarar alla insamlade testresultat till dess att den kan ersättas av nyare testresultat.

En del tester kan endast köras om andra tester har lyckats. Exempelvis går det inte att hitta ”små luftläckor” om det finns en ”stor luftläcka”. I sådant fall markeras testerna som ofärdiga.

3.1.4.1.7 Service $07 - Request Emission-Related Diagnostic Trouble Codes Detected During Current or Last Completed Driving Cycle

För att OBD skall registrera en felkod så måste ett fel uppkomma vid flera resor i rad. Detta för att försäkra sig om att det verkligen finns ett fel. Ett fel som bara uppkommit en gång kommer inte att tända MIL eller registrera en ”riktigt” felkod. Däremot kommer den att registrera ”pending DTC”, dvs. en icke-verifierad felkod. De icke-verifierade felkoderna kan man inspektera genom att använda denna tjänst.

I funktion och beteende är den här tjänsten lika som Service $03, med undantaget att den visar icke-verifierade felkoder istället för verifierade felkoder.

Det finns två anledningar till att denna funktion finns. Den ena anledningen använda den för att hitta fel som faktiskt finns, men inte uppkommit tillräckligt ofta för att registrera en riktig felkod. Detta händer vanligtvis om den kräver vissa extrema situationer som inte uppkommit under mer än en resa. Den andra anledningen är för att kunna testa ett fordon som nyligen har fått sina koder nollställda med Service $04, exempelvis efter att ha reparerats, för att se ifall ett fel finns kvar. I sådant fall är det effektivare att bara behöva en testkörning istället för flera.

3.1.4.1.8 Service $08 - Request Control of On-Board System, Test or Component

Denna tjänst är ett kontrolläge, där man kan beordra OBD-systemet att göra olika saker. Det finns i dagsläget bara en standardiserat parametervärde, och dess uppgift är att ställa in systemet så att det är redo att köra ett ”evaporative system leak test”.
Alla andra parametervärden är reserverade.

3.1.4.1.9 Service $09 - Request Service Information

Denna tjänst används för att få ut fordonsspecifik information. Man kan få ut är värden för exempelvis ”Vehicle Identification Number” (VIN) och för kalibrationsdata.

Vilka egenskaper som stöds kan man få ut på motsvarande sätt som i Service $01.

3.1.4.1.10 Service $0A – Request Emission-Related Diagnostic Trouble Codes with Permanent Status

Denna tjänst existerar enbart om ISO 15765 används som bärarprotokoll. Den har samma beteende som Service $03, men den nollställs inte av Service $04, utan endast när OBD-systemet självt avgör att felet är korrigerat.

Anledningen till att denna tjänst lagts till är att bilägare inte skall fuska på inspektioner genom att nollställa sitt OBD-system innan.

3.1.4.2 Datalänk / fysiskt lager

OBD har fem olika officiellt erkända bärarprotokoll. Protokollen har olika namn och benämningar i olika kretsar. Ibland benämns de efter dokumentet som beskriver det, ibland efter ett formellt eller informellt namn på tekniken eller en förkortning av det. I bild 5 visas hur de här protokollen passar in i OBD-hierarkin.

3.1.4.2.1 ISO 9141-2

ISO 9141 [13][14][15] beskriver en konfiguration för överförande av diagnostikdata från vägfordon till extern diagnostikenhet. Del två är specifikt anpassad för OBD-2[14] systemet.

Överföringshastigheten hos detta system är 10,4 kbaud och den nominella spänningen som används är 12V. Systemet har stora likheter med det mer välkända RS-232 [28]. Det här protokollet förekommer främst i fordon med europeiskt och asiatiskt ursprung.

Protokollet benämns ibland ”ISO-OBD”, i synnerhet i äldre beskrivningar (innan de två andra ISO-specificerade standarderna togs i bruk).

Att gå in i detaljer på hur protokollet fungerar är inte nödvändigt här, men i kort säger standarden att det skall finnas en dubbelriktad kommunikationskanal, kallad ”K-line” och att det kan finnas en enkelriktad till fordonet utöver det, ”L-line”. Systemet är byteorienterat (använder en byte som minsta dataenhet, i motsats till ett meddelande som minsta dataenhet), bitkodningen som används är Non-Return-to-Zero och meddelandena består av 11 bytes uppdelade i 3 byte huvud, 7 byte data och 1 byte checksum. Protokollet använder en initialiseringsprocess, denna initialisering krävs för att systemet skall godkänna förfrågningar.

3.1.4.2.2 SAE J1850

Detta dokument [16] beskriver två kommunikationsprotokoll, som båda används som lågnivåprotokoll för OBD-diagnostik. Dessa två benämns Variable Pulse Width (VPW) och Pulse Width Modulation (PWM). De har vissa egenskaper gemensamt i datalänklagret, men det fysiska lagren är markant olika. Båda dessa är system som använder CSMA/CR.

Data i J1850 skickas i meddelande, varje meddelande som ser ut som bild 6 (ej skalenlig) visar.

SOF	Data	CRC	EOD	IFR	EOF

Bild 6. Meddelandestruktur J1850
SOF står för Start-Of-Frame och markerar att en ny frame sänds.
Data är det faktiska innehållet. Det kan vara upp till tio byte lång.
CRC står för Cyclic-Redundancy-Check är ett kontrollvärde för att minska risken för bitfel.
EOD står för End-of-Data och markerar att meddelandet är slut.
IFR står för In-Frame-Response och tillåter mottagande noder att direkt ge en indikation på att de är nöjda med meddelandet. Denna används inte alltid.
EOF står för End-Of-Frame och markerar att en frame är slut.

3.1.4.2.2.1 Variable Pulse Width

Variable Pulse Width, VPW, används av GM och Chrysler. Den använder en kommunikationssignal och har en genomsnittlig överföringshastighet på 10,4 kb/s. Anledningen till att denna teknik heter som den gör är att en bit inte alltid är lika lång (vilket också leder till att överföringshastigheten varierar). En bit representeras av att en viss spänning uppehålls under ett visst tidsintervall, en ”puls”. En puls har olika betydelse beroende på om det är en lång eller kort puls och om den har hög eller låg spänning den har. Lång puls av hög spänning och kort puls av låg spänning representerar noll och de andra två kombinationerna representerar ett.

Kommunikationen i VPW sker ”broadcast”.Det leder till att alla noder ser meddelandet, inklusive sändaren. Liksom i NRZ har en hög spänning alltid företräde över en låg. På sådant sätt löses nätverkskonflikter i enlighet med CSMA/CR. Om man ser till att meddelanden alltid sänds i takt med varandra så kommer en bit med värdet 0 alltid ha företräde över en bit med värdet 1. Start-of-Frame(se bild 6) i detta fall består av en puls av högspänning som är längre än en ”lång puls”; på sådant sätt synkroniseras nätverket vid varje meddelande som sänds.

3.1.4.2.2.2 Pulse Width Modulation.

Pulse Width Modulation, PWM, är det andra protokollet som beskrivs i J1850. De flesta bilar som använder PWM är tillverkade av Ford. PWM använder två balanserade signaler och har en överföringshastighet på 41,6 kb/s. PWM använder en intern klocka i varje nätverksenhet som synkroniserar om sig vid varje mottagen bit. Det möjliggör högre kommunikationshastighet.

I den här varianten är tiden uppdelade i ”korta enheter”. Under en ”kort tidsenhet” kan spänningen vara antingen hög eller låg. De symboler (bitvärden och kontrolltecken) som skickas på nätverket består av en serie ”korta tidsenheter”. En ”Start of frame”-symbol består av hög spänning i fyra tidsenheter följt av låg i två. En bit med värdet 1 består av en tidsenhet hög spänning följt av två med låg. En bit med värdet 0 består av två tidsenheter med hög spänning följt av en låg.

En inkopplad nätverksnod lyssnar alltid samtidigt som den sänder. Om den under en ”kort tidsenhet” sänder en låg spänning men ser en hög kommer den att inse att någon annan nod på nätverket också sänder och själv sluta sända. Det beteendet leder till att systemet hanterar kollisioner utan förluster.

3.1.4.2.3 ISO 14320

ISO 14320 [17][18][19][20] dokument beskriver ett kommunikationsprotokoll som heter ”Keyword Protocol 2000” (KWP). KWP är en vidareutveckling av ISO 9141, och är i viss mån bakåtkompatibelt med detta. Den accepterar samma initialisering som ISO 9141-2, men även ett annat, snabbare, alternativ. KWP klarar överföringshastigheter på upp till 10,4 kbps och använder sig av meddelanden som kan innehålla upp till 255 byte data.

3.1.4.2.4 ISO 15765

Detta protokoll [21][23][22][24] bygger på en arkitektur som heter CAN (Controller Area Network). CAN beskrivs i detalj i sektion 3.4 Den har en kommunikationshastighet på upp till 1Mbps, vilket är avsevärt högre än de andra systemen. CAN-OBD är det enda alternativet som

är tillåtet i nytillverkade bilar från år 2008. ISO 11898 [29][30][31][32][33] beskriver den generella CAN-arkitekturen.

CAN är äldre än vad OBD-2/EOBD är, men började användas i personbilar först 2003. En av orsakerna till att CAN först börjat användas på senare tid har varit att den har varit dyrare än sina konkurrenter i kombination med att det inte funnits behov av den högre överföringshastigheten som CAN erbjudit.

3.1.4.3 Diagnostikkontakt

Den yttersta delen av OBD-systemet är diagnostikkontakten. Detär via den som man kan få tillgång till systemet.

OBD använder en kontakt med 16 (2x8) stift, där olika stift finns beroende på vilket lågnivåprotokoll som används, se bild 7 och tabell 1. Den fysiska placeringen av kontakten varierar en aning, men den skall sitta så att den är åtkomlig från förarsätet.

Kontakten definieras i detalj i dokumentet J1962[25].

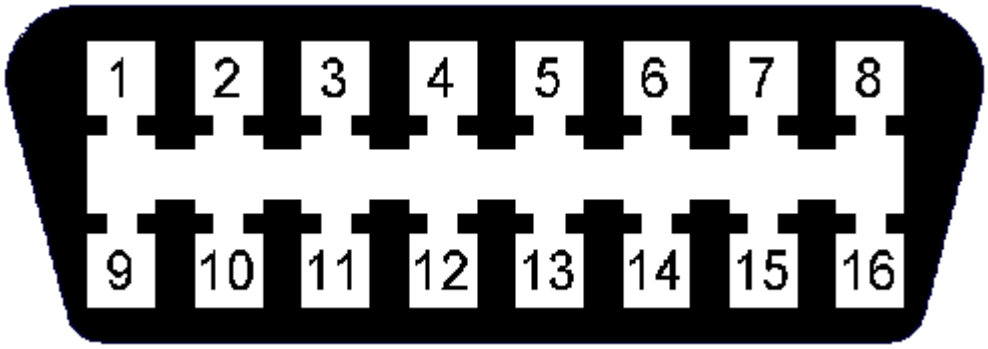

Bild 7. Diagnostikkontakten som används av OBD

Stift	Funktion	Protokoll
2	Buss +	J-1850 PWM / VPW
4	Chassiejord	Alla
5	Signaljord	Alla
6	Buss hög	ISO 15765
7	K-line	ISO 9141, ISO 14230
10	Buss -	J-1850 PWM / VPW
14	Buss låg	ISO 15765
15	L-line	ISO 9141, ISO 14230
16	Batterispänning +	Alla

Tabell 1. Standardiserade stiftfunktioner

3.1.5 Användning

OBD är ett system som i hög grad är riktat till ”vanliga människor”. Väldigt många människor har en bil, och kommer därför i kontakt med OBD. Det leder till att mycket information finns tillgänglig som beskriver systemet för en användare av det.

Det finns många tillverkare av testapparater till OBD, dvs. fristående enheter som kan interagera med ett fordons OBD-system. Fordonstillverkarna själva tillverkar ofta sådana apparater, men majoriteten av de testapparater som finns på marknaden är tillverkade av tredjepartstillverkare.

Testapparater finns i många olika versioner och nivåer. En del är väldigt simpla, riktade till den obildade hemmaanvändaren som vill veta vad som är fel på bilen, och innehåller vanligtvis möjligheten att hämta ut felkoder och förklara dem, men inte mycket mer. Andra är riktade till reparatörer/avancerade hemmanvändare och ger tillgång till allt inom OBD-standarden samt stora mängder tillverkarspecifik information.

Det finns även alternativ till testapparater, och det är att ansluta en dator till systemet. Det finns företag som säljer anslutningar mellan OBD och PC, och det finns företag som säljer programvara till PC och handdatorer som kan tolka informationen som man får tillgång till genom anslutningarna. Sådan programvara är oftast kapabel att hantera hela OBD:s utbud av information.

3.2 FMS-Standard

FMS-Standard [34] är en teknologi för att förmedla fordonsdata till en för fordonet extern övervakare. Den är avsedd att användas i lastbilar och liknande tunga fordon och riktar sig främst mot företag.

Namnet FMS-Standard kommer av att det är en standard avsett att användas tillsammans med ett FMS (se 2.3). FMS-Standard är skapad för att standardisera den del av ett FMS som hanterar informationsinsamlingen.

3.2.1 Ursprung

FMS-Standard är utvecklad av en grupp företag inom lastbilsbranschen. Gruppen kallade sig först FMS-group men har sedan dess blivit en del av ”The European Automobile Manufacturers Association”, ACEA,[33] under namnet “Heavy Truck Electronic Interface Group”.

Anledningen till att standarden är utvecklad gemensamt var att gruppmedlemmarna ville att deras kunder skulle kunna använda lastbilar från olika tillverkare utan att behöva flera olika övervaknings- och diagnostiksystem.

3.2.2 Funktion

FMS-Standard ett gränssnitt som finns till för att en övervakare av ett FMS skall kunna komma åt den information som ett fordon samlar in. Syftet med FMS-Standard är att förse FMS-användaren med en del av den information som används av fordonets egna system. Inte all information förmedlas, utan endast information som är av betydelse för FMS-användaren, exempelvis bränslemängd och hur långt fordonet har körts.

FMS-Standard rapporterar via en diagnostikkontakt med jämna mellanrum värdena för ett antal i standarden definierade egenskaper. Den skickar informationen så fort den själv får tillgång till

den, vilket leder till att informationen får mycket hög uppdateringsfrekvens. Alla tidskritiska egenskaper uppdateras med 100 ms mellanrum eller oftare.

3.2.3 Tillvägagångssätt

FMS fungerar så att en ECU(Electronic Control Unit) som kallas FMS-gateway finns inkopplad på fordonets interna kommunikationssystem (i praktiken är det systemet alltid J1939[36], se stycke 3.3). Enheten skickar sedan ut information med den typ och frekvens som FMS-standard definierar via en extern CAN-anslutning [29][30][31][32][33]. Man kan för att ta del av denna information koppla in en annan enhet, exempelvis en dator, till anslutningen och läsa av informationen.

En inkopplad FMS-gateway är alltid på och därför behöver ett diagnostikverktyg som kopplas in på dess anslutning inte använda någon initialiseringsprocess. Det är till och med så att en inkopplad diagnostikenhet inte skickar någonting alls; all information som tillhandahålls av FMS-standard skickas ut av FMS-gateway vid regelbundna intervall, vare sig det är någon som lyssnar eller inte.

Anledningen till att en FMS-gateway används istället för att bara koppla in ett diagnostikverktyg direkt på fordonens interna kommunikationsnätverk är säkerhet. De interna kommunikationssystemen i fordonet är väldigt beroende på att varje inkopplad ECU följer protokollet. Om man kopplar in en ECU som med hög frekvens skickar skräpmeddelanden med maximal prioritet kommer inte någon annan nod kunna skicka någonting alls och detta påverkar fordonets användbarhet negativt. Fordonstillverkarna vill därför inte att man kopplar in egna enheter [35], men för att information från dessa system ändå skall kunna utvinnas har FMS-Standard utvecklats. Den agerar som ett gränssnitt för kommunikationen från fordonets interna system till en extern enhet.

FMS-Standard är i hög grad baserat standarden SAE J1939[36]. J1939 beskriver det system som vanligtvis används internt i de fordon som FMS-Standard är avsett för. Detta yttrar sig i att FMS-Standard i sig inte beskriver i detalj hur systemet är uppbyggt, utan endast hur den skiljer sig från ett J1939-system.

3.2.4 Meddelandestruktur

De meddelanden som skickas på den externa anslutningen är strukturerade och numrerade likadant som meddelanden i J1939 (specifikt J1939-71[46]) men inte alla meddelanden skickas vidare. Av meddelandena som skickas vidare så är en del information förändrad. Dokumentationen för FMS-Standard beskriver de meddelanden som stöds.

Ett av de värden som FMS-Standard förmedlar är ”Vehicle Identification Number” (VIN). Detta värde kan vara upp till 200 byte långt. Detta får inte plats i ett vanligt meddelande. Lösningen är ett transportprotokoll som heter ”Broadcast Announce Message”, (BAM), som skickar det för stora värdet uppstyckat över flera meddelanden. I FMS-standard används ”Broadcast Announce Message” endast för att överföra fordonets ”Vehicle Identification Number”, men i J1939 används det även till annat.

3.2.5 Händelseförlopp

Interaktion med FMS-Standard följer ett visst mönster:

1. Användaren kopplar in en CAN-anslutning till fordonets FMS-gateway.
2. Denna FMS-gateway rapporterar med jämna mellanrum fordonets status för de värden som den stödjer. Dessa meddelanden kommer i ett stadigt flöde utan att den inkopplade datorn behöver begära dem
3. Om användaren känner att nog information insamlats kopplas anslutningen loss utan att någon nedstängningsprocess eller liknande behövs.

3.2.6 Dokumentationsstruktur

FMS-standard bygger på J1939[36], som presenteras i stycke 3.3. J1939 bygger i sin tur på CAN [29][30][31][32][33] (som presenteras i 3.4) men innehåller egna dokument som beskriver den variant av CAN som används.

Bild 8 visar hur standarderna FMS-Standard, J1939 och CAN hänger ihop.

- FMS-Standard i sig består av ett dokument, ”FMS-document”, som definierar den underliggande strukturen. Den strukturen finns för tydlighetens skull utplockad och beskriven i bild 8. FMS-Standard använder lager 1, 2, 3 och 7 i OSI-modellen.
- J1939 består av ett stort antal dokument, endast de viktigaste av dessa har ritats ut i bild 8. J1939 använder lager 1, 2, 3 och 7 i OSI-modellen.
- CAN finns i många olika varianter, som definieras i olika dokument från olika organisationer. Den vanligaste är ISO 11898 och dess fysiska lager används i FMS-Standard. J1939 har en egen variant av CAN vars datalänklager används i FMS-Standard. CAN täcker lager 1 och 2 i OSI-modellen.

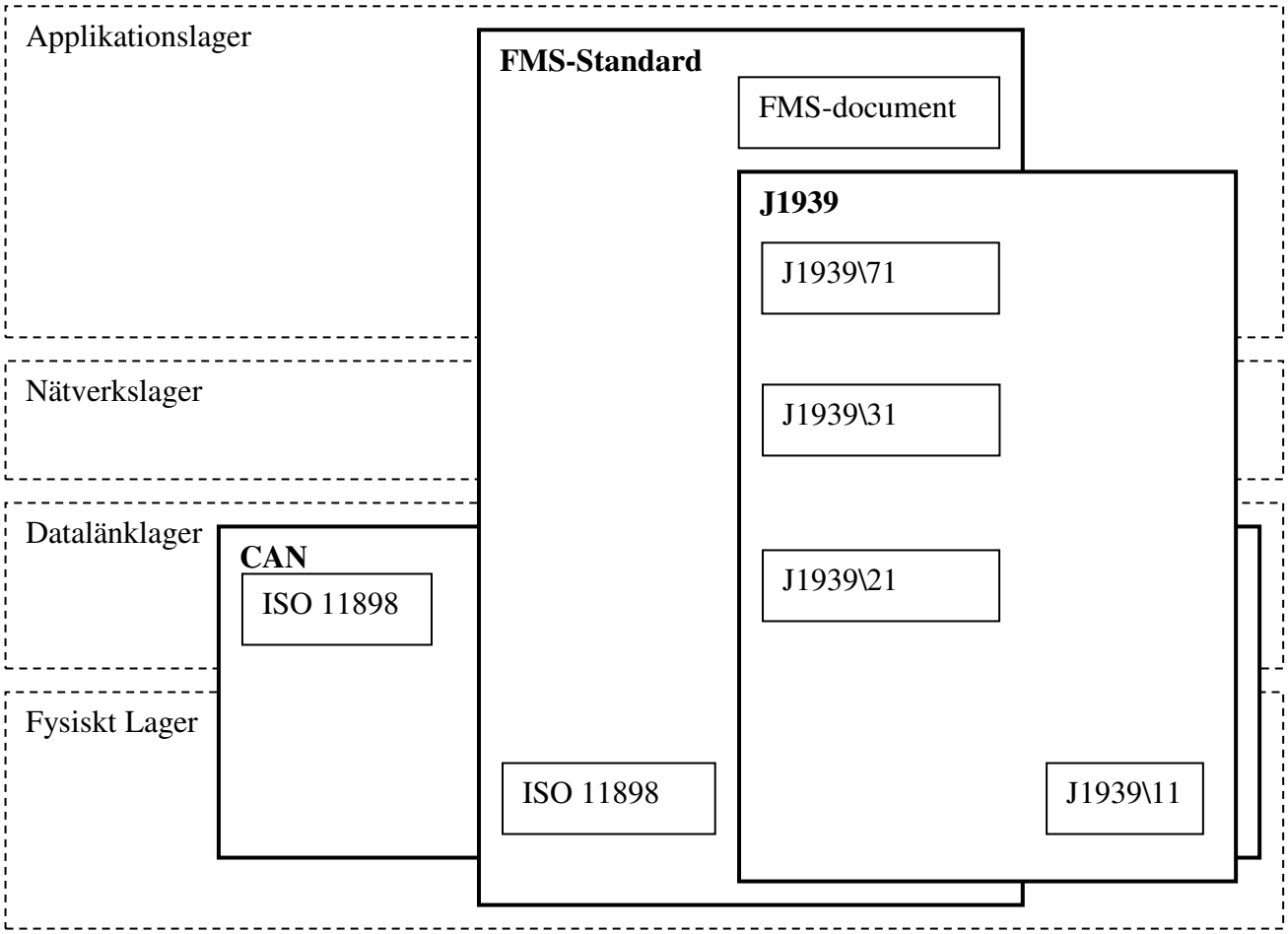

Bild 8. Dokumentationshierarki för FMS-Strandard. FMS-Standard använder sig av J1939 och CAN, men endast delar av dem.

FMS-Standard skulle kunna användas mot ett annat underliggande system än J1939; FMS-gateway:en skulle då översätta den interna kommunikationen till formatet som specificeras i FMS-standard.

3.3 SAE J1939

J1939 är en standard för kommunikationsnätverk i fordon. J1939, vars fulla namn lyder ”Recommended Practice for a Serial Control and Communications Vehicle Network”, är publicerat av Society of Automotive Engineers, SAE, och består i sin tur av ett antal underdokument [42][43][44][45][46][47][37][38][39][40][41][48][49][50][51][52].

SAE är en organisation som existerar för att främja samarbete inom områden som rör transport och mobilitet. Medlemmarna i SAE är oftast sådana som arbetar i närhet till fordon avsedda att användas antingen på land eller i vatten eller luft. Ett av SAE:s mål är att publicera standarder för användning inom dessa områden.

För att man skall bättre kunna förstå varför FMS-Standard är uppbyggt som det är bör J1939 betraktas. Vissa saker i FMS-Standard, exempelvis valet av meddelandenumrering, kan verka underliga annars.

3.3.1 Användningsområde

Det ursprungliga användningsområdet för J1939 var dieseldrivna lastbilar, men standarden används även i industrifordon, jordbruksmaskiner och andra system i miljöer med mycket störning.

J1939 är menat som en ersättning till de äldre standarderna J1587[53] och J1708[54]. J1939 täcker samtliga funktioner som erbjuds av de andra två tillsammans, och har dessutom ytterligare funktionalitet. En sak som skiljer dem åt är kommunikationshastighet, J1939 har markant högre hastighet än J1587/J1708.

3.3.2 Struktur

OSI-modellen(se 2.6) kan användas för att beskriva J1939, men eftersom nätverk inne i fordon inte har samma krav på sig som vanliga datornätverk så specificeras endast 4 av de 7 lagren. De kvarvarande tre lagrens funktioner sköts av andra lager eller behövs inte alls. J1939 använder sig av CAN-B (CAN extended frame) som grund för sina två understa lager, dvs. de lager som CAN definierar.

J1939 består av ett antal understandarder, uppdelat efter lagren i OSI-modellen. För varje lager som implementeras direkt av J1939 finns minst ett dokument som beskriver hur J1939 implementerar det. Dokumenten har namn på formen J1939\XY där X representerar dokumentgrupp, oftast ett lager i OSI-modellen, och Y representerar dokumentinstans i gruppen.

Kärnan i J1939 utgörs av de fyra underdokumenten J1939\11[41], J1939\21[44], J1939\31[45] och J1939\71[46]. Med dessa fyra kan man konstruera ett fungerande J1939-nätverk. Ett J1939-nätverk måste dock inte nödvändigtvis använda just dessa fyra, exempelvis kan man använda ett fysiskt lager som är annorlunda från det som specificeras i J1939\11. Detta kan vara användbart om man behöver ett system med speciella egenskaper.

Applikationslager	J1939\71
Nätverkslager	J1939\31
Datalänklager	J1939\21
Fysiskt Lager	J1939\11

Bild 9. Förenklad dokumentstrukturr hos J1939

- J1939\11 definierar OSI lager 1, det fysiska lagret. Detta dokument beskriver hur det fysiska nätverket skall se ut: hur ledningarna skall vara konstruerade, vilka spänningsnivåer som skall användas, vad som händer vid kortslutning och liknande. Det kan ersättas av exempelvis J1939\15[43], eller av något annat dokument som beskriver ett fysiskt lager.
- J1939\21 definierar OSI lager 2, datalänklagret. Här beskrivs meddelandestrukturen i detalj. J1939 använder ”CAN extended frame”, en av de två standardiserade typer av CAN-meddelande som finns. Här beskrivs också ett transportprotokoll.
- J1939\31 definierar OSI lager 3, nätverkslagret. Nätverket kan vara uppbyggt av undernätverk som sitter ihop med hjälp av noder av typen ”Network interconnection

ECU” dvs. repeater, bridge, router och gateway. Dokumentet beskriver även det protokoll som används för att konfigurera nätverket.

- J1939\71 definierar OSI lager 7, applikationslagret. Här beskrivs den information som skickas över ett J1939-nätverk. De egenskaper som standarden omfattar räknas upp och beskrivs. Dessa egenskaper grupperas sedan i meddelanden, kallade ”Parameter Groups”, som kännetecknas av sitt ”Parameter Group Number”, PGN (se stycke 3.3.5). Dokumentet beskriver även hur information som inte täcks av standarden bör formateras. De egenskaper som finns definierade är de som kan vilja användas i ett fordon.

De övriga dokumenten i J1939 utgör antingen alternativ till någon av dessa fyra, sammanfattande eller kompletterande dokument.

3.3.3 Metodik

J1939 består av ett CAN-nätverk. Varje nod i nätverket är en ECU med en viss uppgift. Exempel på en uppgift är att kontrollera ett av fordonets delsystem, såsom motor, växling och klimatkontroll. En FMS-gateway är en ECU med uppgift att vidarebefordra information till en extern mottagare. Varje ansluten ECU får en adress tilldelad enligt en algoritm beskriven i J1939\81[50]. När en ECU har fått sin adress börjar den sköta sin uppgift.

3.3.4 Meddelanden

I J1939 finns olika sätt att skicka meddelanden. Man kan skicka dem ”broadcast”, för meddelanden som skulle kunna vara av intresse för flera ECU:er eller till en specifik adress. Att skicka till en specifik adress är bra om man har flera enheter som skulle kunna använda meddelandet men man vill att bara en ska det. Det finns även en global adress som gör att meddelanden kommer till alla enheter trots att man inte använder ett broadcastmeddelande. J1939 rekommenderar att man använder meddelanden av typen broadcast om det inte finns en bra anledning att inte göra det.

Meddelandena är också uppdelande i standardiserade och tillverkarspecifika meddelande. De standardiserade meddelandena är de som beskrivs i J1939\71, men de täcker inte upp hela rymden av möjliga meddelande. Delar av det kvarvarande utrymmet är reserverat för att tillverkare skall kunna bygga enheter som använder information som inte täcks av standarden. Detta leder till att man skall vara försiktig om man använder ECU:er från olika tillverkare som använder tillverkarspecifika meddelanden. Om de skulle råka ha valt samma meddelandenummer kan det leda till mycket underliga resultat.

3.3.5 Parameter Group Numbers

J1939 använder sig av något som kallas ”parameter group number”, PGN. Ett PGN beskriver en grupp av relaterade värden. Värdena kan vara relaterade i antingen funktion, ursprung eller uppdateringsfrekvens. Dessa är fast definierade för den standardiserade delen (men naturligtvis inte i den tillverkarspecifika) där ett visst PGN har alltid samma betydelse oavsett i vilken maskin den används. Ett PGN är vad som vanligtvis utgör datadelen i J1939-meddelanden. Ett exempel är meddelandet ”Electronic Brake Controller #1” som har PGN 61441 (0x00F001) och innehåller värden som används för att kontrollera bromsar.

Ett PGN är 3 byte långt och ser ut enligt bild 10.
De sex första bitarna i ett PGN är alltid nollor, de är endast utfyllnad. När ett meddelande sänds så existerar de bitarna endast implicit. ”Data page bit” säger på vilken av två ”sidor” som ett viss PGN finns. Alla nu definierade PGN finns på första sidan (som representeras av bitvärdet noll), vilket betyder att biten är till för framtida utökningar. ”Reserved bit” är också till för framtida utökningar, men vilka utökningar det rör sig om har ännu inte bestämts. Den sänds alltid som noll.
Meddelandena finns i två versioner: adresspecifikt eller ”broadcast”. Om värdet för fältet ”PDU Format” är mindre än 240 är meddelandet av typen ”PDU1” och skall skickas till en specifik adress på nätverket. Adressen anges i det efterföljande fältet som då kallas ”Destination Adress”. Om PDU Format är större än eller lika med 240 är meddelandet på formatet ”PDU2” och skickas som ”broadcast”, dvs. till samtliga noder på nätverket. Det efterföljande fältet kallas då ”Group Extention” och används tillsammans med PDU Format för att avgöra vilket PGN det är.

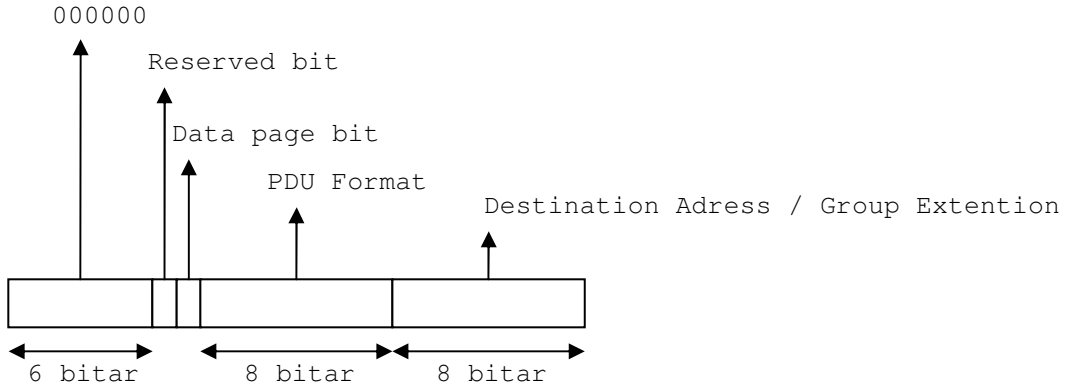

Bild 10: Parameter Group Number

3.3.6 Meddelandestruktur

Meddelanden i J1939 använder CAN-B. Ett meddelande ser ut precis som ett sådant CAN-meddelande (se sektion 3.4.2.1) med tillägget att fälten ”Identifier A” och ”Identifier B” är ifyllda med J1939-specifik information. Bild 11 nedan visar vad dessa fält ersätts med. Betydelserna av fälten beskrivs i den efterföljande texten. Bild 11 visar inte hela CAN-meddelandet. Den del som utlämnats ser ut som i bild 13.

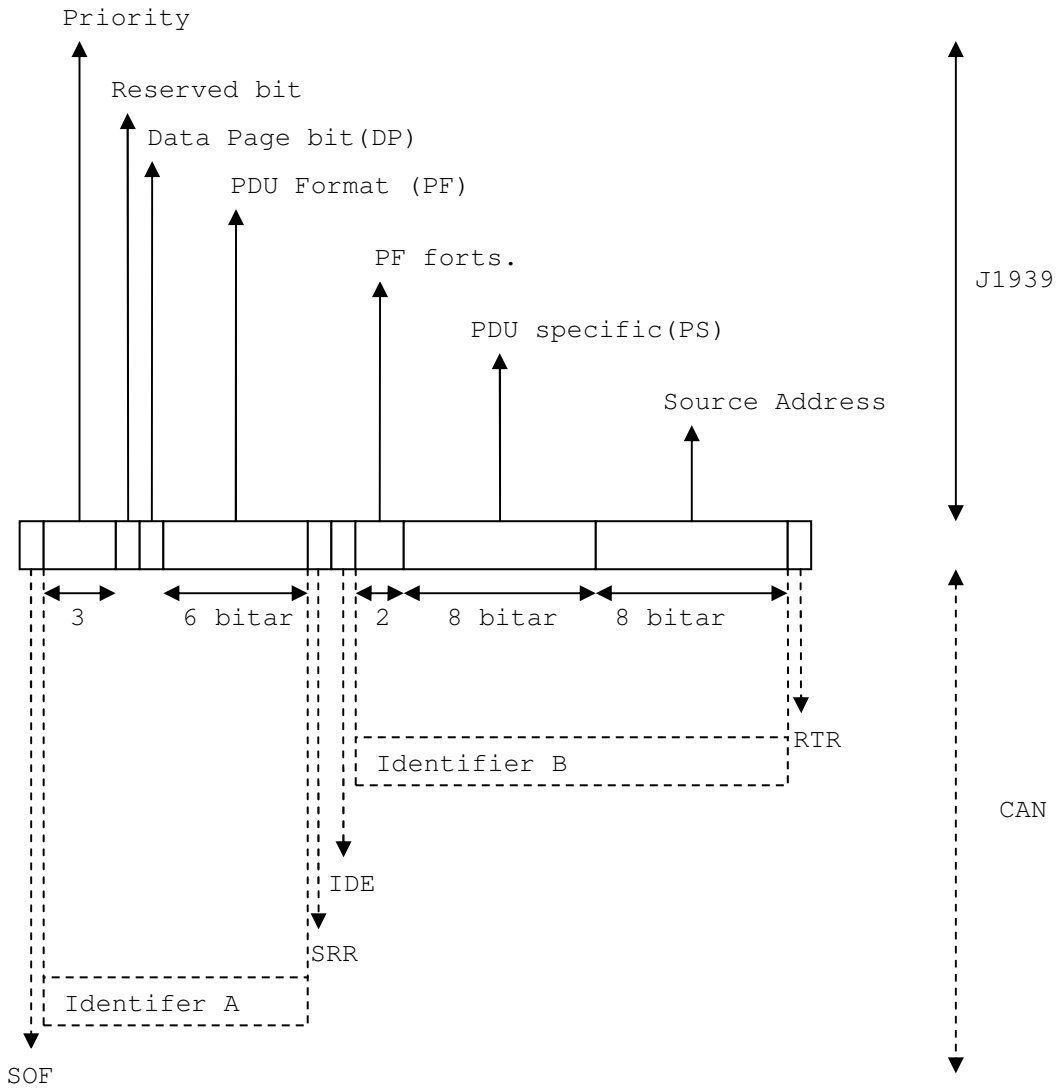

Bild 11 :Meddelandestruktur J1939

Förklaring av fälten (för CAN-specifika fält (streckade pilar), se förklaringen till bild 13):

- Priority: ett 3 bitar långt fält som används för att ange prioriteten som meddelandet har. Låga värden har företräde framför höga.
- Reserverad bit: för framtida utökningar.
- Data Page: en utökning till PDU Format
- PDU Format: avgör huruvida meddelandet skall till en enskild mottagare, samt deklarerar minst delvis meddelandets innehåll.
- PDU Specific: beroende på PDU Format, antingen destinationsadressen för meddelandet, eller avgör tillsammans med PDU Format meddelandets innehåll.
- Source Adress: Sändarens adress.

3.3.7 Enhetsidentitet

Alla enheter som är anslutna till ett J1939-nätverk har en 8-bitars adress, som används när man skickar meddelanden. Alla anslutna enheter har också ett unikt namn, kallat NAME. Det är ett 64-bitars värde som identifierar en viss komponent. Det innehåller typiskt information om tillverkare, modell och serienummer. NAME används bl.a. vid tilldelning av nätverksadresser.

3.3.8 Transportprotokoll

Många av de meddelanden som skickas med J1939 är så pass korta att de får plats i ett 8-byte stort meddelande, men inte alla. NAME är ett exempel på ett värde som inte gör det. För att skicka sådana värden använder J1939 ett transportprotokolet ”Broadcast Announce Message”, BAM, som beskrivs i J1939\21.

3.4 CAN

Controller Area Network, vanligtvis förkortat CAN, är ett lågnivånätverk skapat för att användas i tuffa miljöer, dvs. miljöer med förhållandevis mycket störning. Det har därför konstruerats för att vara tålig mot sådan störning. Den använder sig av korta, frekventa meddelanden, vilket leder till att om det upptäcks ett fel på något meddelande så är det en ganska liten mängd information som behöver sändas om. För att ytterligare skydda systemet används ett kontrollvärde (”checksum”), beräknat med en algoritm som heter CRC-15 [5]. Detta kontrollvärde garanterar att alla meddelanden som har upp till 5 bitar i rad förvrängda kommer att upptäckas av systemet. Dessa två egenskaper leder till ett system som är väldigt pålitligt, detta har gjort att CAN har används i fordons interna system och inom industrin. En annan anledning på senare tid att använda CAN är att det är väl utbrett och det finns många tillverkare av CAN-kompatibla komponenter. Detta har gjort att dessa komponenter har blivigt ganska billiga.

Företaget Robert Bosch GmbH utvecklade CAN under mitten av 80-talet. Den första definitionen färdigställdes 1986 och implementerades tillsammans med Intel 1987. CAN har sedan dess vuxit i popularitet och används nu flitigt.

CAN kan beskrivas i ISO:s OSI-modell(se 2.6). I den täcker CAN de två lägsta lagren, det fysiska lagret och datalänklagret. De övre lagren lämnas odefinierade och flera protokoll som bygger på CAN har vuxit fram för olika applikationsområden. Exempel på sådana är CANopen[57], DeviceNET[58] och J1939[36]. CAN har standardiserat i flera olika varianter av flera olika organisationer. Det som skiljer mellan olika definitioner är det fysiska lagret. Den mest välkända varianten är den som beskrivs i ISO 11898[29][30][31][32][33]. Fördelen med att kunna implementera CAN över olika fysiska lager är att man då kan välja ett lager som erbjuder de egenskaper som man helst vill framhäva i just det aktuella systemet. Vissa varianter maximerar överföringshastigheten medan andra ger utökad feltolerans.

3.4.1 Fysiskt lager

Ett CAN-nätverk består an en linjär seriell databuss. På denna buss finns ett antal noder inkopplade. Dessa noder kallas Electronic Control Units (ECU). Varje nod som är inkopplad på nätverket har möjligheten att skicka och ta emot meddelanden på bussen. Längden på bussen är begränsad, eftersom signalstyrkan och överföringshastigheten minskar med avståndet. Problemet med signalstyrkan kan lösas genom att lägga till noder som förstärker signalen när den passerar dem. Exempel på sådana noder är ”bridge”, ”gateway” och ”repeater”. De ger även möjlighet att använda ett mer avancerat nätverk än en linjär buss, eftersom de kan skicka signalen vidare över flera bussar. Nätverkets egentliga struktur har dock ingen betydelse för en enskild nod (annat än just de noder som existerar för att utöka nätverksstrukturen), dessa kan och ska betrakta nätverket som en linjär buss. För att systemet skall fungerar korrekt så måste ändarna av bussen termineras med ett motstånd, förutsatt att det fysiska lagret inte är av sådan typ att det inte behövs (t.ex. en optisk länk).

CAN använder sig av en bitkodning som heter Non-Return-to-Zero, NRZ(se stycke 2.5.3.2.1). En nod i ett CAN-nätverk måste alltid skicka med sin egen adress när den skickar ett meddelande på bussen, och alla noder i måste ha unika adresser. Detta garanterar att två olika noder aldrig kan sända identiska meddelandehuvuden. Därmed behöver inte kollisionshanteringen användas över den del av meddelandet som innehåller data, endast över meddelandets huvud.

En annan egenskap av NRZ är att CAN-nätverket inte använder sig av en central klocka. Om samma spänning ligger aktiv en längre stund kan därför noder ibland tappa bort var de var någonstans. Därför används en teknik som kallas ”bit stuffing”. Detta kräver att om fem likadana bitar sänds i rad så måste en motsatt bit följa direkt efter dem. Denna bit är extrainsatt och den kommer att hoppas över när betydelsen av meddelandet läses av. Den biten existerar enbart för att påminna noderna i nätverket om hur lång en bit skall vara.

3.4.2 Datalänklager

Kommunikation i CAN sker via ”broadcast”, eftersom alla noder är kopplade till samma buss. Det betyder alla noder ser alla meddelanden som skickas. En egenskap av detta är att meddelanden inte i sig har en destinationsadress. Varje nod som är inkopplad på nätverket får själva avgöra om ett inkommet meddelande är relevant eller inte.

3.4.2.1 Meddelandetyper

Ett CAN-meddelande kan vara av fyra olika typer:

- Data frame: innehåller information.
- Request frame: begär att ett visst data frame skall skickas.
- Error frame: signalerar att en nod har upptäckt ett korrupt meddelande.
- Overload frame: signalerar att en nod fått motta för mycket information på för kort tid för att hinna behandla den.

3.4.2.1.1 Error Frame

En ”Error frame” består av 6 nollor följt av 8 ettor. När en nod upptäcker ett fel av något slag skickar den en ”error frame”. En ”error frame” bryter medvetet mot ”bit stuffing”-regeln, och får därmed de övriga noderna att också signalera fel. När alla är klara med felrapporteringen kommer systemet befinna sig i ett stabilt tillstånd, dvs. alla noder är redo att ta emot ny information.

3.4.2.1.2 Overload Frame

Overload frame används av en nod för att signalera att den har mottagit information i högre takt än vad den klarar att hantera. Overload frame fungerar på ett sätt liknande error frame men används vanligtvis inte längre eftersom dagens hårdvara klarar meddelanden i hög hastighet.

3.4.2.1.3 Data Frame och Request Frame

Det finns två format för data- och request-meddelanden i CAN. Den första kallas ”CAN base frame”, eller ”CAN-A”. Den andra kallas ”CAN extended frame” eller CAN-B. Skillnaden mellan de två är att CAN-A använder en 11-bitars identifierare medan CAN-B använder en 29

bitars identifierare. Den exakta meddelandestrukturen för meddelanden av typen data eller remote ser ut på så sätt som visas i bild 12 och bild 13 och förklaras i den efterföljande texten.

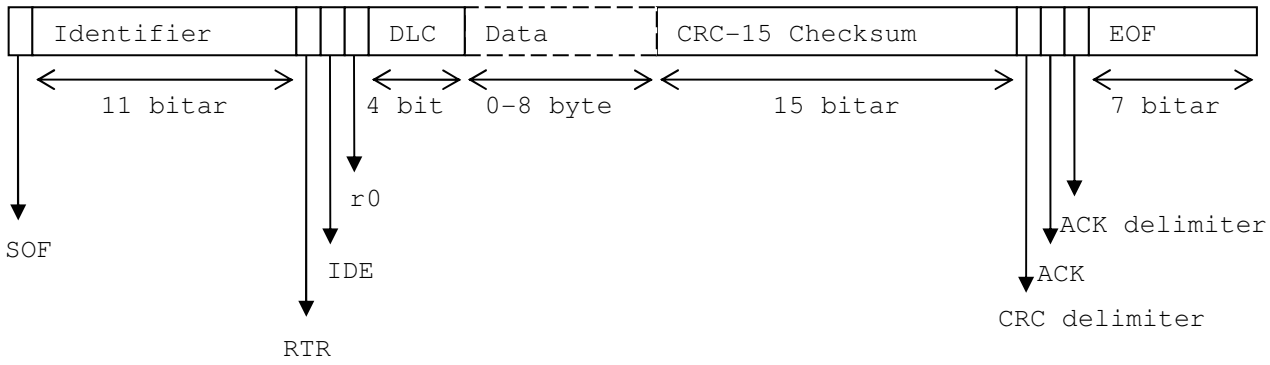

Bild 12: Meddelandeformat CAN-A

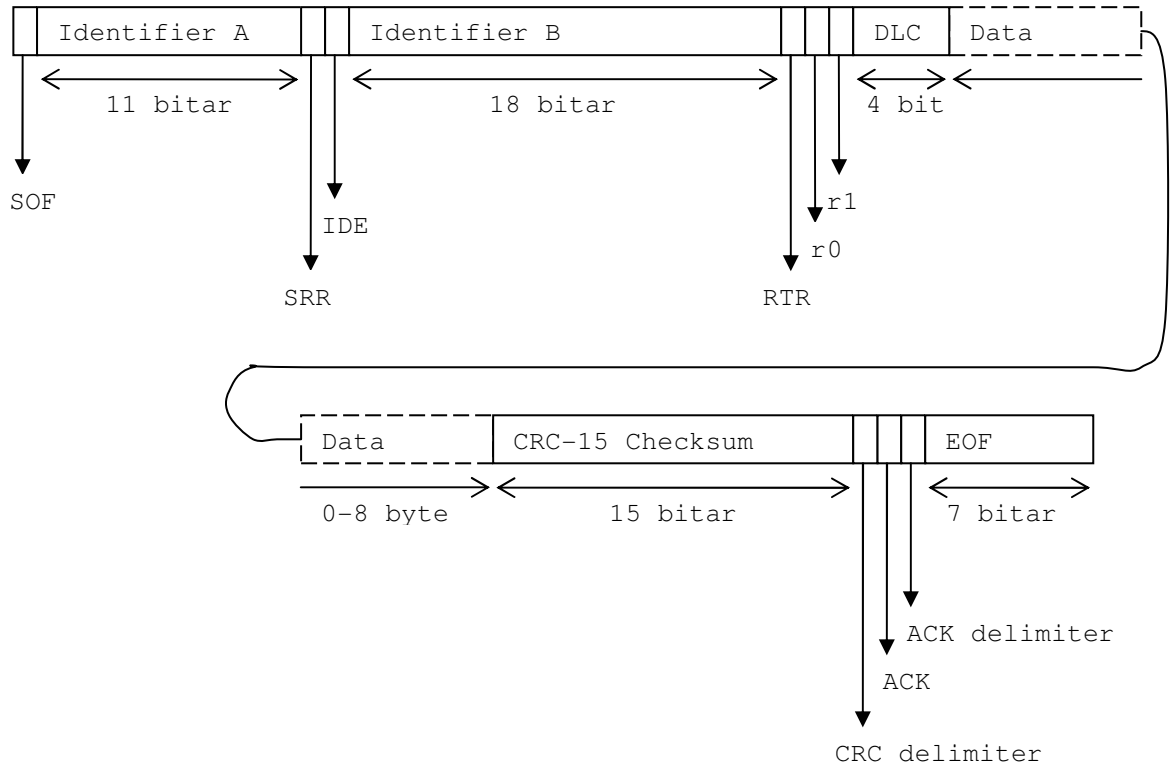

Bild 13: Meddelandeformat CAN-B

Förklaring av fälten:

- SOF: Start-Of-Frame: en bit som signalerar att ett meddelande börjar sändas. Är alltid 0.
- Identifier: Ett elva bitars fält som innehåller användardefinierat header-information. Den skall innehålla sändarens adress. Används i CAN-A
- RTR: Remote Transmission Request: avgör om meddelandet är ett datameddelande eller ett requestmeddelande.
- IDE: IDentifier Extention bit: avgör om meddelandet är CAN-A eller CAN-B.
- r0: Reserverad bit. Sannolikt för framtida utökningar av systemet. Måste skickas som 0, men accepteras som antingen 0 eller 1.

- DLC: Data Length Code: beskriver hur många byte data som meddelandets datafält innehåller. Giltiga värden är 0 till och med 8. För requests berättar den hur många meddelanden som förväntas.
- Data: Innehållet i meddelandet.
- CRC: Cyclic-Redundance-Check: ett värde som används för att kontrollera att meddelandet framförts intakt.
- CRC delimiter: är alltid 1.
- ACK: Acknowledgement: skickas som 1, men en mottagare kan istället skicka 0 för att indikera att meddelandet inte tagits emot korrekt.
- ACK delimiter: är alltid 1.
- EOF: End-of-Frame: signalerar att meddelandet är slut. Består alltid av sju 1:or.
- Identifier A: första delen av identifierarfältet i CAN-B. Identifier A är 11 bitar långt.
- SRR: Substitute Remote Request: är alltid 1.
- Identifier B: andra delen av identifierarfältet i CAN-B. Identifier B är 18 bitar långt, vilket tillsammans med Identifier A blir 29 bitar.
- r1: Reserverad bit. Se r0.

Egentligen är det inte så stora skillnader mellan de olika formaten. I stort kan man säga att CAN-B har fler möjliga meddelanden, tar något längre tid att skicka och har lite sämre felhantering eftersom valet av CRC-algoritm är baserat på att CAN-A används.

3.4.3 Felhantering

Som det beskrivs under 3.4.2.1.1 har varje nod möjlighet att ogiltigförklara ett meddelande. Detta skulle kunna vara farligt, om det blev fel på en nod så att den tyckte att alla meddelanden är felaktiga. För att undvika detta så har CAN-komponenter en inbyggd säkerhetsspärr som säger att om den hittar för många fel inom en viss tid, eller om för många meddelanden som den skickat anmäls som felaktigs så skall den sluta sända ett tag. Om inte det gör saken bättre skall noden stänga av sig.

3.5 Andra standarder

Det finns andra standarder med användningsområden eller egenskaper som liknar de som OBD och FMS-Standard har. Tanken var ursprungligen att dessa skulle beskrivas i viss detalj, men det har det inte funnits tid nog för. Mycket kort kommer dock de som påträffats på att nämnas.

3.5.1 Heavy-Duty On-Board Diagnostics

HD-OBD[55] är en version av OBD som är menat att användas i tunga lastbilar. Det betyder att den skulle vara en konkurrent alternativt ett komplement till FMS-Standard. Den skall träda i kraft i USA 2010, efter att ha blivit framskjutet från 2007. Detta verkar vara en mycket intressant standard som bör undersökas.

3.5.2 J1587 / J1708

Detta system är den närmaste föregångaren till J1939. Det är ett äldre system som är avsett att användas i tunga maskiner, främst industri. J1587[53] är ”huvudstandarden” och J1708[54] beskriver det fysiska lagret. I jämförelse med J1939 har detta system mindre funktionalitet och lägre kommunikationshastighet.

3.5.3 Bus FMS

Bus FMS[56] är en variant av FMS-Standard som är avsett att användas i bussar istället för lastbilar. Skilladen ligger i att viss information som inte finns hos lastbilar kan förmedlas.

3.5.4 J1939-varianter

Det finns ett flertal system som utvecklats från J1939 som åtminstone teoretiskt skulle kunna användas i dess ställe. NMEA2000[59] och ISO 11992[60][61][62][63] är två sådana.

3.5.5 Tillverkarspecifika diagnostiksystem

Det finns sannolikt egendefinierade diagnostiksystem som används internt av fordonstillverkare, oavsett typ av fordon (privatbilar, lastbilar, båtar, flygplan osv.). Inget publikt sådant har hittats under examensarbetet och därmed har inga undersökningar skett.

4 ANALYS AV OBD OCH FMS-STANDARD

När man väl vet hur OBD och FMS-Standard är uppbyggda och vad de har för kapacitet kan man diskutera vad detta ger för effekter.

Om man jämför OBD och FMS-Standard mot varandra märker man att de inte har identiskt samma funktion. Det betyder att det finns saker som vartdera systemet inte är kapabelt till. En saknad funktion är en brist i systemet, förutsatt att den saknade funktionaliteten vore önskvärd. Om funktionen å andra sidan inte behövs bör den inte vara med alls. Är den det tar den upp plats och bidrar till att öka komplexiteten i standarden utan att samtidigt öka funktionaliteten.

Utöver att jämföra dem mot varandra vill man också jämföra dem mot den potentiella funktionaliteten, dvs. man vill hitta funktionalitet som både OBD och FMS-Standard saknar. Detta beror dock väldigt mycket på användningsområde, och kommer därför att hållas kort.

4.1 Syfte och ursprung

Standarderna OBD och FMS-Standard har i grunden olika syften och ursprung.

FMS-Standard är avsett att användas av företag för att övervaka tunga vägfordon och dess förare. Anledningen till att det finns tillgängligt är för att standardens skapare vill att deras kunder skall vara nöjda och för att få dem att inte använda någon annan teknik.

OBD är från dess skapares sida avsett att användas av privatpersoner för att deras personbilar inte skall släppa ut för stora mängder avgaser. Anledningen till att OBD skapats är att myndigheterna vill skydda miljön.

Denna skillnad leder till att systemen fokuserar på olika områden. FMS-Standard har fokus på att övervaka realtidsdata hos fordonet, medan OBD avser begränsa fordonets utsläpp.

4.2 Omfattning

Systemen skiljer sig från varandra på så sätt att de utför olika mängd arbete.

FMS-Standard
- läser data
- vidarebefordrar data

OBD
- läser data
- analyserar data
- sparar data om det är viktig
- varnar föraren om någonting verkar konstigt
- vidarebefordrar data

OBD är ett mer avancerat system än vad FMS-Standard är. Följden av detta är att FMS-Standard kräver mer områdeskunskap för att användas framgångsrikt, men att FMS-Standard är ett enklare system att använda överhuvudtaget.

4.3 Målgrupp

FMS-Standard är anpassat för lastbilar och fordon av liknande storlek (sopbilar, jordbruksfordon, etc.). Standarden utgår från att fordonet använder ett J1939-nätverk internt. FMS-Standard är i allmänhet ett tillval i de fordon den omfattar.

OBD finns i personbilar och lätta lastbilar. Detta är en mycket stor grupp fordon. OBD är oftast obligatoriskt i fordon som omfattas av standarden.

OBD har både större målgrupp och större utbredning inom målgruppen. Det leder till att den har avsevärt fler aktiva system och att det finns fler aktörer inom området.
Exempel på sådana aktörer är:

- företag som vill påverka standarden att passa bättre med deras produkter eller tjänster
- företag som informerar om fördelarna med systemet
- fordonstillverkare eller andra företag som vill sälja kringutrustning (testapparater etc.)
- politiker som vill influera bestämmelser som rör standarden
- myndigheter som informerar om gällande lagar
- användare som informerar om alla möjliga aspekter av systemet

Resultatet av att olika aktörer vill olika saker är att kompromisser och specialfall förs in i standarden.

4.4 Användningsområde

Den information man får ut genom att använda sig av fordonsdiagnostik kan användas till ett antal olika saker. De primära användningsområdena är:

- Realtidsvärden under körning
- Analys av fordon
- Analys av förare
- Övervakning av utsläpp

Standardernas skilda ursprung har lett till att de presterar olika väl inom varje område.

4.4.1 Realtidsvärden under körning

Det finns en del egenskaper som är av stor vikt för föraren av ett fordon under körning. Två exempel är fordonshastighet och bränslenivå. Dessa är dock så viktiga att de i allmänhet redan finns tillgängliga för föraren på fordonets kontrollpanel, vilket minskar värdet av denna funktion. Det kan dock finnas situationer där den är användbar:

- om en för tillfället viktig egenskap som inte finns tillgänglig på kontrollpanelen behöver övervakas
- om precisionen på kontrollpanelen är sämre
- om kontrollpanelens mätare inte är tillgänglig (trasig, skymd etc.)
- om någon annan än föraren vill övervaka dessa egenskaper.

FMS-Standard och OBD övervakar bägge egenskaper som skulle kunna vara användbara inom området. FMS-Standard har 27 egenskaper som den förmedlar värden för, dessa är utvalda för att de är av intresse för användaren. OBD ger tillgång många gånger fler egenskaper, men en

stor del av dessa är små detaljer av fordonets system och av ringa intresse för en mänsklig observatör. De viktigaste värdena rapporteras dock av båda systemen, vilket leder till att de i praktiken är jämlika på denna punkt. En eventuell nackdel för OBD är att ifall någon av de långsammare infrastrukturerna används kan det märkas i form av låg uppdateringsfrekvens.

4.4.2 Analys av fordon

Om ett fordons prestation försämras på grund av ett fel eller slitage vill man upptäcka det. Genom att spara de realtidsvärden som fordonet rapporterar och jämföra dem över tid kan man upptäcka förändringar i fordonets beteende. Utöver att upptäcka att det är fel får man ofta även en indikation på vad som är fel.

I OBD-systemet är fordonsanalys en central del. OBD-systemet övervakar på egen hand fordonet och genomför analyser för att se till att alla värden ligger inom förutbestämda gränser. Här är de detaljerade egenskaper som OBD övervakar användbara. Att OBD övervakar små detaljer gör att små fel kan upptäckas, som till exempel små luftläckor. OBD genomför övervakningen och när den hittar ett fel väljer den ut den av de fördefinierade felkoderna som bäst beskriver felet. Denna felkod sparas undan tillsammans med en ögonblicksbild av realtidsvärdena för att senare kunna inspekteras av en reparatör. Felkoderna beskriver dock oftast effekterna av ett fel, inte orsakerna, vilket gör att manuell felsökning också måste användas. Denna är dock avsevärt enklare, eftersom man från felkoden vet vad man söker efter och man kan använda ögonblicksbilden och OBD-systemets funktion att rapportera realtidsdata som stöd.

Hos FMS-Standard är detta område inte fullt lika utvecklat. Standarden tillåter övervakning av ett flertal intressanta egenskaper, men inte på samma detaljnivå som OBD. FMS-Standard utför heller inte någon egen analys; den bara rapporterar värden. Det betyder att man som användare av systemet själv måste stå för all kunskap om vilka värden hos vilka egenskaper som indikerar att fordonet fungerar sämre än det borde.

4.4.3 Analys av förare

Olika personer kör på olika sätt. Vissa av dessa sätt är bättre än andra. Kraftiga inbromsningar och accelerationer är dåliga med avseende på bränsleeffektivitet, miljövänlighet och slitage. De värden som kan rapporteras kan användas för att få en indikation på hur ett fordon har framförts. Man kan alltså upptäcka när en förare beter sig på ett icke önskvärt sätt.

FMS-Standard har bra möjligheter för detta. Den rapporterar kontinuerligt status för gaspedal, bromspedal, växling, varvtal och hastighet. Dessa kan användas för att bedöma en förares körteknik. Dessutom ger FMS-Standard tillgång till information som samlas in av den digitala tachografen, som är till för att just övervaka föraren.

OBD rapporterar data som kan användas för att analysera förarens beteende, men den information erbjuds av OBD är inte lika omfattande den som erbjuds av FMS-Standard. OBD tillhandahåller dock en del värdefull information, främst hastighet, gaspedalposition och varvtal. OBD som standard har inget stöd för tachograf-data, men man skulle kunna tänka sig att om en personbil skulle vara utrustad med en tachograf så skulle den kunna rapportera data via egendefinierade värden.

4.4.4 Övervakning av utsläpp

I dagens samhälle börjar intresset för utsläpp bli allt större. Det diskuteras mycket om den globala uppvärmningen och andra negativa effekter av utsläpp. Därför är det av intresse att veta hur mycket avgaser ett fordon släpper ut. Dels av vilja att miljön skall må bra och dels för att utsläpp kan begränsas i lagar.

OBD är avsett att sköta just detta. OBD har i sitt system inprogrammerat gränser för de värden den övervakar som relaterar till utsläpp. Om någon gräns överskrids kommer OBD att rapportera ett fel. Så länge OBD inte har rapporterat ett fel släpper fordonet inte ut mer avgaser än vad som är tillåtet. Dock ger OBD inte en användare direkt tillgång till utsläppsvärden. Om man vill ha information om utsläpp utanför den automatiska analysen blir man tvungen att beräkna dessa utifrån andra övervakade parametrar (vid sådan beräkning vet man säkert, via den automatiska analysen, huruvida några gränsvärden för utsläpp överskrids).

FMS-Standard har ingen automatisk felsökning, och övervakar inte tillräckligt många egenskaper för att felsökning garanterat kan göras manuellt. Standarden kan dock användas för att hitta vissa avvikande beteenden (ex. om mycket mer bränsle går åt än vad som borde kan man anta att utsläppen också är högre än normalt). FMS-Standard rapporterar dock värden som gör det möjligt att beräkna mängden utsläpp under en viss färd, under antagandet att fordonet inte överskrider sina normala utsläppsvärden.

4.5 Olika betraktningssätt

Analys av fordon kan delas upp på andra sätt än vilket resultat man söker. Man kan även utföra analys över olika lång tid: korttidsanalys och långtidsanalys.

Med korttidsanalys menas att den information som ett system tillhandahåller som är användbar direkt. Att veta hur fort fordonet färdas eller hur mycket bränsle som finns kvar är värdefullt direkt när informationen finns tillgänglig.

Viss information behöver övervakas under lång tid för att bli användbar. Att spara insamlat fordonsdata och göra analyser på stora mängder kan visa på mönster som kan vara av intresse. Via denna teknik kan man hitta beteenden som är större än omedelbara fel, men det är ofta svårt att veta vad man skall söka efter. Ett enkelt exempel som redan tagits upp är att analysera en förares körteknik över lång tid. Ett annat exempel kan vara att det upptäcks att fordon oftare råkar ut för fel under vintern, kanske pga. kyla eller vägsalt.

Realtidsdata som rapporters av OBD och FMS-Standard innefattar båda en del egenskaper som kan vara användbara för korttidsanalys och en del för långtidsanalys och en del värden som är användbara för båda. Den automatiska felsökningen i OBD använder båda dessa tillvägagångssätt.

4.6 Användare

FMS-standard är skräddarsytt för att företag i åkeribranschen skall använda det till att övervaka sin vagnpark. Det ger företaget en möjlighet att övervaka hur väl deras fordon fungerar, hur väl deras förare kör och i viss mån även utsläpp.

Även OBD kan användas av företag för att övervaka fordon och förare. Här gäller dock en annan klass av fordon, och därmed andra företag. En budfirma eller taxifirma kan mycket väl använda ett stort antal personbilar utrustade med OBD.

För företag inom den sortens branscher finns ett intresse av att i första hand veta vad systemen i deras fordon är kapabla till och därefter att använda dem för att förbättra sin verksamhet.

OBD används även automatiskt av alla som äger ett fordon utrustat med det, vare sig dessa är privatpersoner eller företag. Den användningen är dock ofta begränsat till att låta reparera fordonet när MIL tänds.

4.7 Dokumentation

Eftersom OBD och FMS-Standard har olika målgrupper har de även information formaterad på olika sätt.

Informationen som finns att tillgå om FMS-standard är i princip begränsad till standarddokumentet. Detta leder till höga krav på användaren som förväntas vara veta vad FMS-Standard är och vad det skall användas till. Standard-dokumentet i sig är ganska sparsamt beskrivet och utgår från att läsaren är bekant med J1939. Dess användare förväntas förstå sig på den korthuggna beskrivningen, de förväntas veta vad FMS-Standard är och varför de vill använda systemet. De system som FMS-Standard bygger på, J1939 och CAN, finns det utförlig information om.

Information om OBD finns i två varianter. Den första är information till bilägare, som fokuserar på miljöaspekten, de ekonomiska fördelarna och de praktiska effekter som användandet av OBD har för en bilägare. Denna information utgår från att dess användare är ”vanliga människor”, utan någon specialkunskap. Den här sortens information finns i sin tur från två sorters källor, från myndigheter som vill informera om gällande bestämmelser och från företag som vill sälja OBD-relaterade produkter.

Den andra varianten är teknisk information. Där inkluderas dels standarddokumenten själva och dels mer informella beskrivningar av systemet från ett tekniskt perspektiv. Information av den senare typen är inte lätt att hitta. I kombination med att dokumentationen av OBD är utspridd över många olika dokument blir det svårt att få en överblicksbild av hur systemet är uppbyggt. Avsaknaden av en bra översikt leder till svårigheter med att förstå vad ett visst dokument beskriver och varför det ”saknas” delar (som då beskrivs i ett annat dokument). Själva standarddokumenten är visserligen inte svåra att hitta, men att söka efter dem förutsätter att man vet vad vilka de är.

4.8 Övervakade egenskaper

FMS-Standard och OBD har vardera en utvald mängd egenskaper som de rapporterar värden för. Dessa egenskaper är utvalda för att de skulle kunna vara av intresse för användaren eller, i fallet OBD, det automatiska diagnostiksystemet. Vardera standarderna rapporterar värden för egenskaper som inte den andra stödjer. Att det finns egenskaper som en av standarderna inte stödjer leder till att det kan finnas egenskaper som båda systemen inte stödjer.

Vilka egenskaper som detta skulle vara är dessvärre mycket svårt att veta utan god områdeskunskap. Någonting som dock skulle kunna vara av intresse är att få tillgång till mätvärden från de sensorer som mäter mängden utsläpp som bilen avger (eventuellt har OBD detta genom något av dess mer obskyra värden).

FMS-standard har i dagsläget bara begränsade möjligheter till att övervaka utsläpp. För att få ut något värde för utsläpp måste man räkna på andra värden som systemet har matat ut, i kombination med fordonets nominella utsläppsvärden och antagandet att dessa utsläppsvärden inte överskrids (t.ex. p.g.a. att någon komponent i fordonet är trasig).
OBD gör beräkningar som verifierar att fordonet inte överskrider sina utsläppsvärden, men dessa sker internt i OBD, och det resultat man får ut är antingen ja eller nej. Utöver att kunna vara säker på att fordonet inte överskrider sina utsläppsvärden är man begränsad till att använda samma metod som för FMS-Standard. Att ha tillgång till utsläppsvärden direkt skulle kringgå den beräkningen och sannolikt ha bättre precision.

Att ha för få värden är alltså dåligt, men inte heller är det bra att ha för många. Om ett diagnostiksystem rapporterar en stor mängd värden som ingen användare är intresserad av leder det till att standarden blir större och mer komplex utan att samtidigt öka dess funktionalitet.

FMS-Standard har endast ett litet antal egenskaper som rapporteras. Dessa är av sådan typ att det är lätt att förstå vad de representerar och vad de skulle kunna användas till.

OBD har ett stort antal egenskaper. En del av dessa är väldigt detaljerade och har inte något uppenbart användningsområde. Detta är i och för sig endast en liten nackdel för OBD eftersom kommunikationen är av typen request-response. Är man inte intresserad av ett visst värde behöver man inte fråga om det.

4.9 Sammanfattning

OBD och FMS-Standard är bägge två solida system som uppfyller sina respektive syften.
De har konstruerats av människor som haft ett specifikt mål i åtanke och vetat hur de skall nå det. Därför finns det inga större fel eller brister att peka på, och de som finns beror på att avvägningar behövts göras. Man har helt enkelt blivit tvingade att välja mellan egenskaper som helt enkelt inte går att kombinera (t.ex. hög överföringshastighet mot hög feltolerans).

De två systemen är dock olika och det leder till att de har lite olika funktionalitet. Skillnaderna har dock mycket liten praktisk effekt eftersom de olika teknikerna inte används i samma fordon.

De två teknikerna erbjuder vardera tjänster som kan vara av stort intresse om man har en grupp fordon. För ägare av enstaka fordon minskar värdet, eftersom det krävs en viss mängd ansträngning för att kunna börja använda ett diagnostiksystem och flera av tjänsterna tjänar på att ha data från andra fordon att jämföra emot.

5 FMSLIB

I examensarbetet ”Fordonsdiagnostik” ingår en utvecklingsdel: En implementation av FMS-Standard i form att ett programbibliotek. Detta bibliotek har fått namnet ”FMSlib”. FMSlib och dess utveckling beskrivs i detta kapitel.

I stycke 5.1 beskrivs systemets utseende och funktion och i 5.2 dess utveckling. I stycke 5.3 beskrivs hur ett motsvarande bibliotek baserat på OBD skulle kunna utvecklas.

5.1 Systembeskrivning

FMSlib är ett programbibliotek som kan användas av andra program som ett gränssnitt mot en FMS-gateway. Detta bibliotek är avsett att användas på en dator som använder Windows CE och har en fysisk CAN anslutning.

5.1.1 Systemöversikt

Målet var att skapa ett bibliotek som skulle kunna användas av andra program för att få tillgång den information erbjuds av FMS-Standard.

I en verklig miljö kommer det FMSlib att interagera med två komponenter. Dels med ett fordon med vilken den delar en CAN-buss och dels med det program som använder sig av FMSlib för att få tillgång till informationen som finns i FMS-meddelandena. Denna kringmiljö som biblioteket avses användas i fanns dock inte tillgänglig under utvecklingsarbetet, och behövde därför simuleras.

För att simulera ett fordon används ett program som körs på en vanlig PC med en inkopplad CAN-anslutning. Programmet har ett grafiskt gränssnitt där man kan ställa in vilka värden som ”fordonet” för tillfället har. De värdena skickas därefter på CAN-anslutningen.

FMSlib är ett bibliotek; för att kunna användas behövs en användarapplikation som anropar det. Ett sådant program har också skapats för att bibliotekets funktionalitet skall kunna verifieras. Detta program frågar FMSlib om data och visar den på skärmen.

Systemet som implementerats består av tre delar

- Fordonssimulator: Ett program som simulerar ett fordon genom att skicka FMS-meddelanden
- FMSlib: Ett bibliotek med funktionen att tillgodogöra sig FMS-meddelanden och vidarebefordra informationen i dem
- Användarapplikation: En ”generisk användarapplikation” med grafiskt gränssnitt som använder data som hämtas från biblioteket

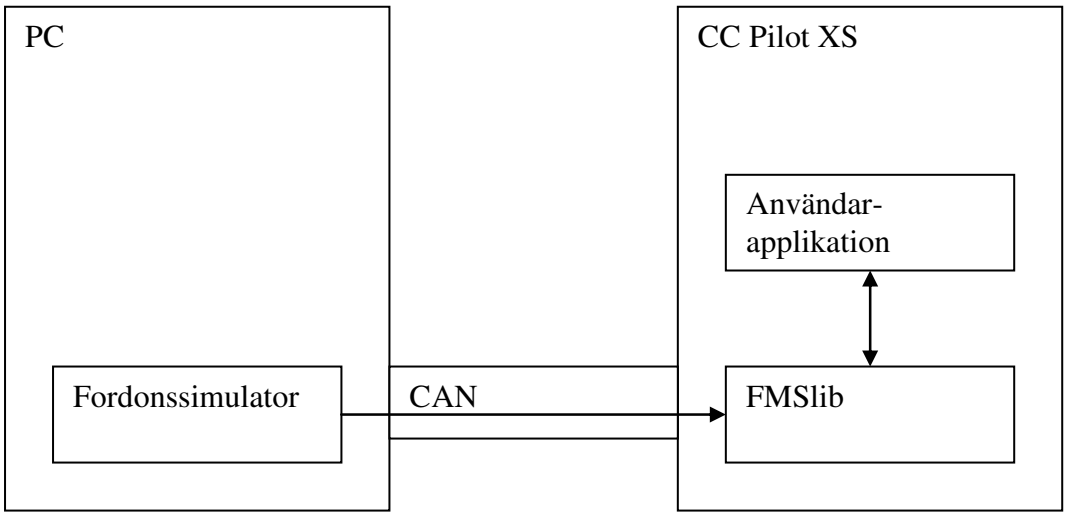

Bild 14 FMS-lib med testmiljö

5.1.1.1 Exekveringsmodell

Detta är en beskrivning av hur informationsflödet i systemet ser ur.

Användaren matar in ett värde i ett av fordonssimulatorns GUI-fält. Med jämna mellanrum genereras ett FMS-meddelande innehållandes information om vad som finns i det fältet. Det meddelande skickas sedan ut på den CAN-anslutning som finns kopplad till datorn som simulatorn exekverar på.

Inkopplad på samma CAN-anslutning finns också en CCP XS[69], som är en maskin som kör Windows CE och har en fysisk CAN-anslutning. På denna maskin exekverar en användarapplikation som använder FMSlib. FMSlib vet om att det finns en CAN-anslutning och lyssnar på den. När ett FMS-meddelande kommer fram behandlar FMSlib detta genom att extrahera informationen som finns i meddelandet och spara undan den för lagring i FMSlib:s lokala minne. Efter det kommer FMSlib att invänta ytterligare händelser.

Användarapplikationen kommer med jämna mellanrum att fråga FMSlib vad värdet för ett visst fält är för tillfället. FMSlib svarar då med det värde den för tillfället har tillgängligt i sitt lokala minne. Användarapplikationen använder värdet den fått till att uppdatera ett värde som visas på skärmen. Användaren kan visuellt inspektera det för tillfället gällande värdet.

5.1.2 Fordonssimulator

Fordonssimulatorn är ett program som genererar FMS-meddelanden som om det vore ett fordon. För en observatör som sitter vid ”fordonets” utåtgående CAN-buss skall det upplevas som att den är kopplad till ett faktiskt fordon.

Den fysiska komponenten som skickar FMS-meddelanden kallas FMS-gateway. En FMS-gateway är kopplad till två olika CAN-anslutningar, dels till fordonets interna CAN-buss som används för kommunikation från och till exempelvis motor och bromssystem och dels till sin externa CAN anslutning där en utomstående övervakare tillåts ansluta för att ta del av den information som erbjuds av FMS-standard. Den uppgift en FMS-gateway utför är att anpassa

de meddelande som tas mot från fordonets interna CAN-buss och vidarebefordra informationen på den externa CAN-anslutningen i enlighet med FMS-standarden.

Fordonssimulatorn är ett program som exekverar på en traditionell PC. Denna PC har en fysisk CAN-anslutning kopplad till sig. Fordonssimulatorn efterliknar beteendet hos en FMS-gateway genom att skicka FMS-meddelanden över den anslutna CAN-bussen.

Till skillnad från en FMS-gateway är fordonssimulatorn inte kopplad till något internt nätverk som den samlar data ifrån. Istället genereras FMS-meddelandena utifrån inställningar som kan modifieras via ett GUI.

Fordonssimulatorn är utvecklad i C# för PC.

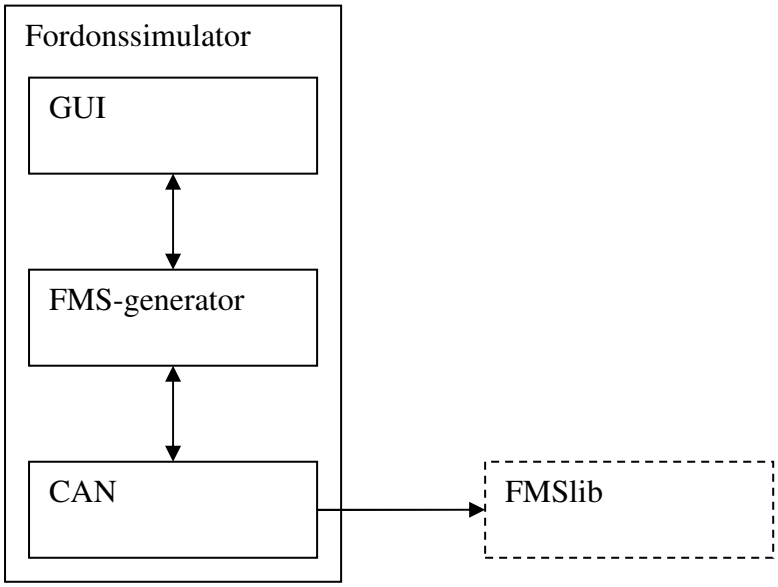

Bild 15. struktur Fordonssimulator

5.1.3 FMSlib

FMSlib är ett fristående bibliotek som skall kunna användas i andra projekt för att samla in information från fordon som använder FMS-Standard. I det här systemet kommer biblioteket att användas av användarprogrammet för att samla in de FMS-meddelanden som fordonssimulatorn genererar.

Anledningen till att implementera FMS-Standard som ett bibliotek och inte som ett fristående program är att ett bibliotek är bättre lämpat för det avsedda användningsområdet. Man vill inte ha ett program som bara kommunicerar, man vill ha en komponent som kan användas av ett program som har en anledning att kommunicera.

FMSlib erbjuder ett API som ger tillgång till informationen som samlas in från den anslutna CAN-bussen. Informationen kommer att vara formaterad för att skickas som svar på ett funktionsanrop, och därmed inte ha samma struktur som FMS-meddelandena. Det anropande programmet behöver inte veta hur FMS-meddelanden är uppbyggda.

FMSlib hanterar all kontakt med CAN-bussen. Det anropande programmet behöver endast berätta för FMSlib vilket CAN-nätverk den skall ansluta till. I enlighet med FMS-standard kommer FMSlib aldrig att skicka applikationsinformation på CAN-bussen, bara ta emot.

FMSlib är avsett att kunna användas i en CCP XS och använder därför endast de funktioner av C# som finns tillgängliga i det begränsade körtidsmiljön, ”.NET Compact Framework”, som finns till Windows CE.

FMSlib lagrar en lokal kopia av all information som den tar emot i sitt temporära lagringsutrymme. När ny information erhålls ersätts det tidigare lagrade värdet. Det betyder att ett värde alltid är vara tillgängligt när användarprogrammet efterfrågar det.

Eftersom lagringsutrymmet används av två olika delar av programmet behövs en koordinationsenhet som ser till att båda delarna inte försöker använda samma värden samtidigt.

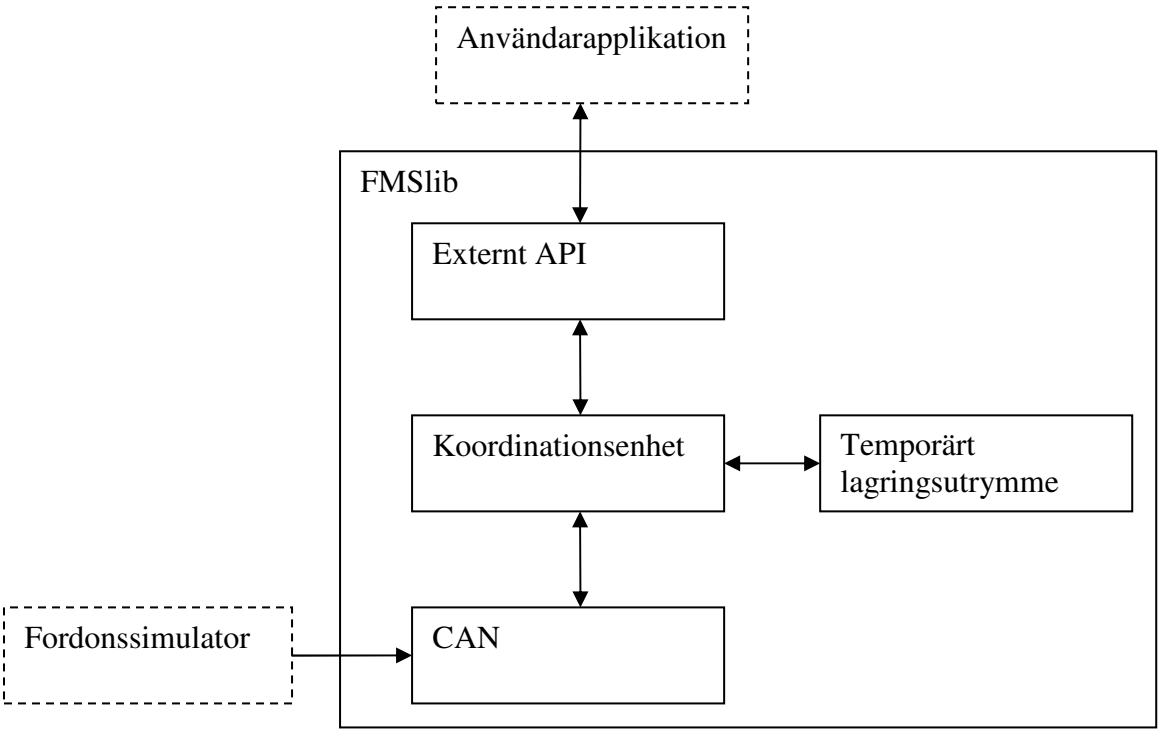

Bild 16. struktur FMS-lib

5.1.4 Användarapplikation

FMSlib är skapat för att användas av en annan applikation. Detta är den applikation som implementerats under examensarbetet för att kunna testa FMSlib. Till skillnad från ”normala” användarapplikationer är denna applikation endast till för att använda FMSlib; att verifiera FMSlib:s korrekta beteende.

Det programmet gör är att be FMSlib om information för att sedan visa den mottagna informationen i ett grafiskt användargränssnitt.

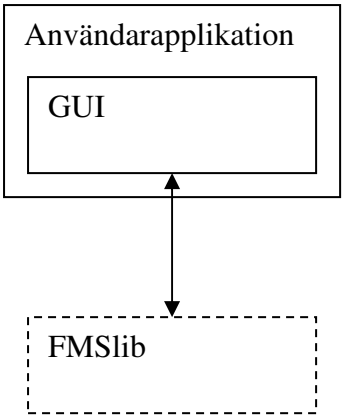

Bild 17. struktur användarapplikatinon

5.1.5 Utökningar

FMSlib är begränsat i omfång av den tid som kan avsättas för dess utveckling. Det finns funktioner som skulle kunna vara användbara för FMSlib men som av tidsskäl inte kan implementeras inom examensarbetet. Potentiella utökningar presenteras här.

En funktion som skulle kunna vara användbar är att informationen som skickas samlas i någon sorts permanent lagring: en loggfil, en databas eller liknande. Denna funktion vore antagligen eftertraktad bland användare av programmet för att kunna föra statistik över längre tid. En annan fördel med att kunna göra så vore att man skulle kunna samla in data från ett riktigt fordon och sedan låta fordonssimulatorn ”spela upp” den. Det skulle ge väldigt realistiska simuleringar. Funktionaliteten för att samla data i en logg skulle kunna finnas antingen i FMSlib, förslagsvis som ett underbibliotek till detta, eller i användarapplikationen.

En funktion för att ”direkt” generera statistik skulle också kunna vara tänkbart. ”Ge mig medelvärdet av momentanförbrukningen av bränsle för de nästa 5 minuterna” skulle kunna vara ett anrop som denna funktion kan hantera. En annan variant på att öka statistiktillgängligheten vore att låta den temporära lagringen innehålla ett visst antal nyligen inkomna meddelande. Då skulle man omedelbart kunna svara på en del statistikförfrågningar. En nackdel med båda dessa utökningar är att de skulle öka bibliotekets minnesanvändning, något som bör undvikas för ett program som avses exekvera på en hårdvara med begränsade resurser.

Det skulle kunna vara önskvärt för vissa användarapplikationer om biblioteket kunde övervaka vissa värden och meddela sin användare om specifika situationer uppstår, exempelvis om bränslenivån blir för låg.

Att direkt utöka bibliotekets funktion skulle störa dess ”klarhet”. Därför är det antagligen bäst att inte göra direkta utökningar i biblioteket utan hellre lägga dessa antingen i en användarapplikation, som ett lager ovanför FMSlib eller i ett underbibliotek.

En annan utökning, eller snarare en ändring, vore att samla all data om protokollet i en extern konfigurationsfil. Beroende på innehållet i denna konfigurationsfil skulle biblioteket kunna acceptera olika meddelandeuppsättningar, och därmed också kunna erbjuda olika API-funktioner. På det sättet skulle FMSlib vara möjligt att anpassa till externa krav utan att man behöver ändra i programkoden.

5.1.6 Klasschema

Här presenteras konceptuella klasstrukturer för delprogrammen. Dessa strukturer skiljer sig något från de som faktiskt implementerats: kontrolldata har utelämnats och en del strukturer förenklats. I bilaga A beskrivs det faktiska externa gränssnittet.

5.1.6.1 Användarprogram

UserApplication

Private:

```
Form GUI
FMSlib F
Main
```

DescriptionBox

Public:

```
Label Name
TextBox Value
Label Unit
```

GUI

Private:

```
DescriptionBox WheelSpeed
DescriptionBox ClutchSwitchIsPressed
DescriptionBox BrakeSwitchIsPressed
DescriptionBox CruiseControlIsActive
DescriptionBox PtoState
DescriptionBox AcceleratorPosition
DescriptionBox TotalFuelUsed
DescriptionBox FuelLevel
DescriptionBox EngineSpeed
DescriptionBox AxleWeight00
DescriptionBox AxleWeight01
DescriptionBox AxleWeight02
DescriptionBox AxleWeight03
DescriptionBox AxleWeight04
DescriptionBox AxleWeight05
DescriptionBox AxleWeight10
DescriptionBox AxleWeight11
DescriptionBox AxleWeight12
      .             .
      .             .
DescriptionBox AxleWeight54
DescriptionBox AxleWeight55
DescriptionBox TotalEngineHours
DescriptionBox VehicleIdentification
DescriptionBox RequestesSupported
DescriptionBox DiagnosticsSupported
DescriptionBox SoftwareVersion
DescriptionBox VehicleDistance
DescriptionBox ServiceDistance
DescriptionBox TachoDriveRecognized
DescriptionBox TachoDriver1WorkingState
DescriptionBox TachoDriver1WorkingState
DescriptionBox TachoOverSpeed
DescriptionBox TachoDriver1Card
DescriptionBox TachoDriver2Card
DescriptionBox TachoDriver1TimeState
DescriptionBox TachoDriver2TimeState
DescriptionBox TachoDirection
DescriptionBox TachoPerformanceAnalysis
DescriptionBox TachoHandlingInformation
DescriptionBox TachoSystemEvent
DescriptionBox TachoVehicleSpeed
DescriptionBox EngineCoolantTemperature
Button StartRecieving
Button StopRecieving
Button Exit
```

Bild 18: klasschema användarprogram

5.1.6.2 FMSlib

FMSlib

Private:

```
BufferArea B
CanControl FMSconnection
```

Public:

```
GetWheelSpeed()
ClutchSwitchIsPressed()
BrakeSwitchIsPressed()
CruiseControlIsActive()
GetPtoState()
GetAcceleratorPosition()
GetTotalFuelUsed()
GetFuelLevel()
GetEngineSpeed()
GetAxleWeight(
           uint axleColumn,
           uint axleRow)
GetTotalEngineHours()
GetVehicleIdentification()
RequestesSupported()
DiagnosticsSupported()
GetSoftwareVersion()
GetVehicleDistance()
GetServiceDistance()
GetTachoDriveRecognized()
GetTachoDriver1WorkingState
()
GetTachoDriver1WorkingState
()
GetTachoOverSpeed()
GetTachoDriver1Card()
GetTachoDriver2Card()
GetTachoDriver1TimeState()
GetTachoDriver2TimeState()
GetTachoDirection()
GetTachoPerformanceAnalysis
()
TachoHandlingInformationIsA
vailable()
TachoSystemEventHasOccured(
)
GetEngineCoolantTemperature
()
```

BufferArea

Properties: (get & set)(synchronized)

```
float WheelSpeed
bool ClutchSwitchIsPressed
bool BrakeSwitchIsPressed
bool CruiseControlIsActive
PtoState PtoState
float AcceleratorPosition
float TotalFuelUsed
float FuelLevel
float EngineSpeed
float AxleWeight00
float AxleWeight01
float AxleWeight02
float AxleWeight03
float AxleWeight04
float AxleWeight05
float AxleWeight10
float AxleWeight11
float AxleWeight12
        .
        .
float AxleWeight54
float AxleWeight55
float TotalEngineHours
String VehicleIdentification
bool RequestesSupported
bool DiagnosticsSupported
int SoftwareVersionMajor
int SoftwareVersionMinor
ulong VehicleDistance
int ServiceDistance
bool TachoDriveRecognized
TachoDriverState TachoDriver1WorkingState
TachoDriverState TachoDriver2WorkingState
bool TachoOverSpeed
bool TachoDriver1Card
bool TachoDriver2Card
TachoDriverTimeState TachoDriver1TimeState
TachoDriverTimeState TachoDriver2TimeState
TachoDirection TachoDirection
TachoPerformace TachoPerformanceAnalysis
bool TachoHandlingInformation
bool TachoSystemEvent
float TachoVehicleSpeed
short EngineCoolantTemperature
```

Bild 19: klassschema FMSlib

5.1.6.3 Fordonssimulatorn

Vehicle

Private:

```
Form GUI
CanControl CC
```

VerifiedInput

Public:

```
Label Name
Constant Label NewValue
TextBox Input
Button Verify
Constant Label CurrentValue
TextBox Verified
Label Unit
```

GUI

Private:

```
VerifiedInput WheelSpeed
VerifiedInput ClutchSwitchIsPressed
VerifiedInput BrakeSwitchIsPressed
VerifiedInput CruiseControlIsActive
VerifiedInput PtoState
VerifiedInput AcceleratorPosition
VerifiedInput TotalFuelUsed
VerifiedInput FuelLevel
VerifiedInput EngineSpeed
VerifiedInput AxleWeight00
VerifiedInput AxleWeight01
VerifiedInput AxleWeight02
VerifiedInput AxleWeight03
VerifiedInput AxleWeight04
VerifiedInput AxleWeight05
VerifiedInput AxleWeight10
VerifiedInput AxleWeight11
VerifiedInput AxleWeight12
      .            .
      .            .
VerifiedInput AxleWeight55
VerifiedInput TotalEngineHours
VerifiedInput VehicleIdentification
VerifiedInput RequestesSupported
VerifiedInput DiagnosticsSupported
VerifiedInput SoftwareVersion
VerifiedInput VehicleDistance
VerifiedInput ServiceDistance
VerifiedInput TachoDriveRecognized
VerifiedInput TachoDriver1WorkingState
VerifiedInput TachoDriver1WorkingState
VerifiedInput TachoOverSpeed
VerifiedInput TachoDriver1Card
VerifiedInput TachoDriver2Card
VerifiedInput TachoDriver1TimeState
VerifiedInput TachoDriver2TimeState
VerifiedInput TachoDirection
VerifiedInput TachoPerformanceAnalysis
VerifiedInput TachoHandlingInformation
VerifiedInput TachoSystemEvent
VerifiedInput TachoVehicleSpeed
VerifiedInput EngineCoolantTemperature
Button StartSending
Button StopSending
Button Exit
```

Bild 20: klassschema fordonssimulator

5.1.7 GUI

5.1.7.1 Användarapplikation

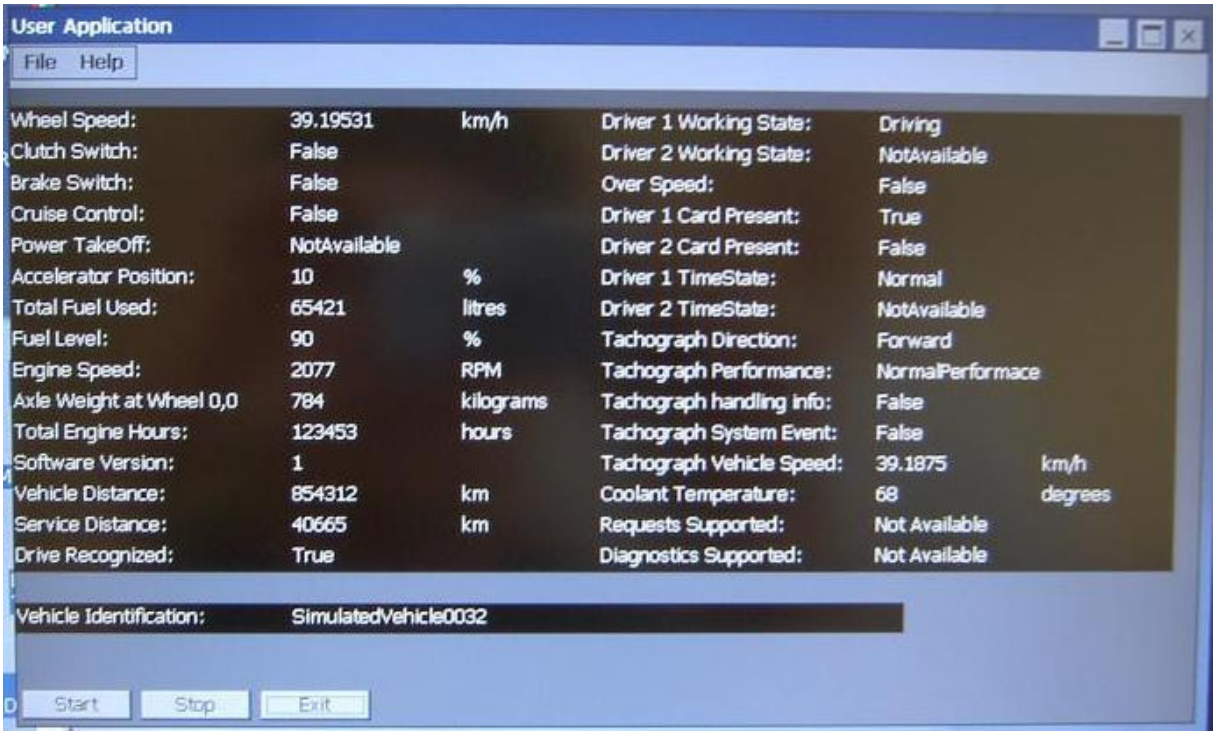

Bild 21: GUI användarapplikation

5.1.7.2 Fordonssimulator

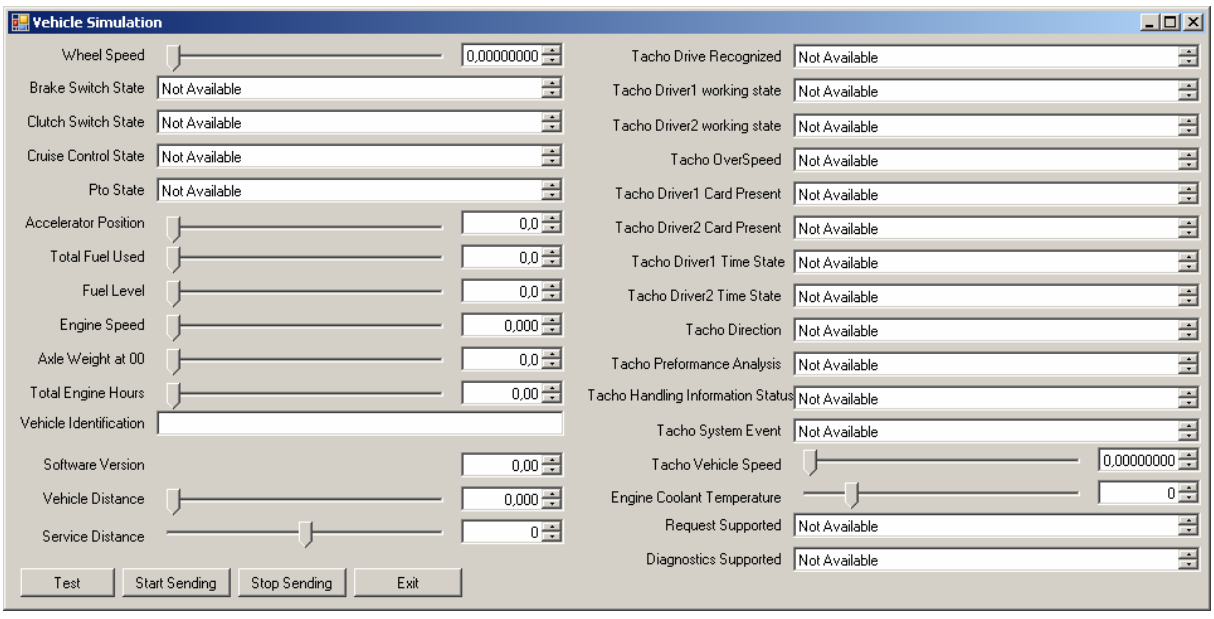

Bild 22: GUI fordonssimlutaor

5.2 Utvecklingsprocess

Detta stycke består av en stegvis beskrivning av hur utvecklingen av FMSlib gick till. Det innefattar förutom själva utvecklingsprocessen även beskrivning av de problem och andra situationer som uppstod samt kommentarer och åsikter kring dessa.

5.2.1 Upplägg

För att jag skulle kunna ha en överblick över hur utvecklingen framskred så valde jag att dela in utvecklingen i tre faser:

1. Konstrukion: att bestämma systemets utseende
2. Implementation: att skriva programkod
3. Testning: att testa systemet

Denna metod är allmänt vedertagen under beteckningen ”vattenfallsmetoden” och lämpar sig väl till utvecklingsarbeten av överskådlig storlek och komplexitet.

5.2.2 Konstruktion

Den specifikation som jag blev tilldelad var ganska lös, så jag spenderade första delen av utvecklingstiden med att själv definiera de krav som skulle komma att finnas på systemet. Dessa krav samlades i ett dokument. Det blev inte ett väldigt långt dokument då relativt få krav fanns och de som fanns kunde beskrivas koncist. Kravspecifikationen beskriver vad de olika delarna i systemet skall göra men inte hur.

När väl en fungerande kravspecifikation fanns att tillgå satte jag mig ner och designade ett system som skulle uppfylla dessa krav. Det är här som jag undersökte problemet och valde en lösning till det. Resultatet av detta blev den beskrivning som finns i kapitel 5. Där beskrivs systemet, dess komponenter och hur dessa hänger ihop. Denna beskrivning är medvetet lagt på en ganska hög nivå, för att tillåta anpassningar under utvecklingsarbetet. Beskrivningen innehåller ett förslag till klasschema för varje delkomponent i systemet. Dessa klasscheman kunde visserligen inte följas exakt, men de ger en tillräckligt god bild av systemet för att en annan utvecklare skall kunna förstå hur systemet är tänkt att fungera.

5.2.3 Implementation

Jag skulle utveckla ett verktyg som skickade data enligt ett specificerat format över en CAN-anslutning. Utvecklingen skulle ske i ett utvecklingsverktyg jag inte hade använt och ett språk jag inte heller använt, mot en extern hårdvara.

För att kunna klara detta behövde jag veta hur det fungerade att arbeta i den miljön. Jag började med att konstruera några simpla program. De programmen byggdes inte för att användas utan för att jag skulle lära mig att interagera med utvecklingsverktyget och hur det går till att utveckla program till en extern hårdvara (i motsats till program som går att köra på datorn man utvecklar på). Jag fick också ökad kunskap om de delar av programmeringsspråket som jag inte kände igen från liknande språk.

Därefter lades min fokus på att interagera mot CAN. Jag hade redan gjort den teoretiska undersökningen, men hur det fungerar i praktiken är ofta annorlunda från hur man tänker sig det. För att använda CAN fanns ett gränssnitt redan tillgängligt. Det gränssnittet studerade jag och lärde mig använda. Jag upptäckte efter ett tag ett problem med gränssnittet. Dess felrapportering var inte korrekt dokumenterat. Detta ledde till att jag spenderade en del tid med att söka efter fel som inte fanns innan jag lyckades fundera ut vad det var som var konstigt.

Efter ett tag hade jag fått klart för mig hur man gjorde för att kommunicera över CAN. Jag hade nu också en god inblick i hur C# fungerade, i grunden likt många andra språk men med en

del detaljer som var krångliga. Dess hantering av trådar och kommunikationen mellan dem gav mig vissa bekymmer, men de lyckades jag reda ut.

Efter det följde själva utvecklingsarbetet. Det innebar att applicera FMS-Standard. FMS-Standard hade jag redan innan undersökt. Den är ganska sparsamt förklarad, och refererar mestadels till J1939. Även denna hade jag studerat, och förstod mig på till den nivå att jag kunde implementera systemet.

Utvecklingen från det här skedet gick bra. Vissa motgångar inträffade, men inga oövervinneliga sådana. En del mindre avvikelser gjordes från min ursprungliga design för att passa bättre med programmeringsspråket.

5.2.4 Testning

När väl systemet var färdigt skulle det testas. Mina egna tester visade på att höga gränsvärden hanterades lite underligt. FMS-Standard sade att dessa maxvärden för en övervakade egenskap betyder ”not available”. J1939 säger att en del värden omedelbart innan maxvärdet representerar andra specialfall. Mitt program tog inte hänsyn till någon av de värdena.

Biblioteket har även testats mot extern data. Att de program som jag har skapat kan skicka data mellan varandra betyder inte nödvändigtvis att FMS-standard följs. Därför gjordes det tester mot inspelad fordonsdata. I början gav detta väldigt underliga resultat. Det visade sig att min tolkning av meddelandeformatet i FMS-Standard hade omvänd byte-ordning jämfört med vad ett faktiskt fordon hade. Jag tyckte, och tycker, detta verkar underligt men anpassade mitt program detta.

Ett annat bekymmer jag stötte på under testningen var att programmet inte avslutades när det skulle. Detta visade sig bero på ett antagande som inte var genomtänkt, den funktion som användes för att ta emot CAN-meddelanden hade en timeout som var satt väldigt högt, och programmet kunde inte avslutas så länge det väntade.

5.3 OBDlib

Detta stycke har för avsikt att kort beskriva ett förslag på hur ett system med samma funktion som FMSlib men som använder OBD skulle se ut. Detta system kommer att refereras till som ”OBDlib”.

5.3.1.1 Utveckling

Rent praktiskt så kan man följa samma förlopp som jag redan framgångsrikt brukat mig av. Man kan ha god hjälp av att känna till innehållet i de dokument som jag skapat. Man bör dock vara medveten om att OBD är en större standard, både i vad den kan förmedla, hur den gör detta och hur det är beskrivet.

En skillnad är att värden från FMS-Standard rapporteras kontinuerligt, medan i OBD behöver man skicka förfrågningar efter de värden man vill veta. Eftersom FMSlib fungerar genom förfrågningar kommer man då att kunna vidarebefordra förfrågningar från användarprogrammet direkt till OBD-systemet och på så sätt kunna klara sig utan de lokala kopior som FMSlib har för alla värden.

Användargränssnittet hos OBDlib skulle inte vara svår att implementera. I stort skulle det se likadant ut som för FMSlib, men undantag för att OBD har ett större antal värden.

Gränssnittet mot OBD kommer däremot att bli krångligare. För implementationen av FMS-standard fanns redan en fysisk anslutning med tillhörande gränssnitt redo att användas. OBD har fem olika underliggande tekniker, vilket skulle betyda att alla fem skulle behöva vara tillgängliga både i form av fysisk utrustning och i form av teknisk kunskap. Naturligtvis kan man välja att endast implementera ett eller några av protokollen om man vet på förhand att man inte kommer komma i kontakt med alla varianterna.

5.3.1.2 Överföringhastighet

FMS-Standard använder CAN, som har tillräckligt hög uppdateringshastighet för att en användare inte skall uppleva någon fördröjning mellan att värden begärs och erhålls. OBD kan använda CAN, men tillåts använda andra system som är avsevärt långsammare. Det skulle kunna ha negativ inverkar på systemet i form av märkbara fördröjningar mellan att värden begärs och svar erhålls.

FMS-standard kräver ca 100 meddelanden per sekund och varje meddelande är ca 100 bit. Detta leder till att det skickas ca 10 000 bps, eller 10kpbs, som är den minimala överföringshastigheten som OBD använder. I teorin skulle det kunna fungera, men eftersom de långsammaste versionerna av OBD använder andra meddelandestrukturer och kommunikationsprotokoll än FMS-Standard kan är det inte riktigt så enkelt. Dessutom innehåller OBD många fler värden än FMS-Standard, även om inte alla fordon stödjer alla värden.

Resultatet av detta skulle antagligen bli:

- ISO 9141 är något för långsamt
- KWP2000 är något för långsamt
- J1850 VPW är något för långsamt
- J1850 PWM skulle antagligen räcka till
- CAN skulle definitivt räcka till

Om man avser implementera systemet skulle detta behöva undersökas i högre detalj.

Om överföringen inte är tillräckligt snabb har man två möjligheter. Den ena är att minska antalet värden som övervakas och den andra att sänka uppdateringsfrekvensen hos vissa värden. Man kan då prioritera ”viktiga” värden och se till att de uppdateras pålitligt genom att välja bort eller sänka uppdateringsfrekvensen hos andra värden.

5.3.1.3 Tillgängliga värden

OBD och FMS-Standard förmedlar olika värden, alla egenskaper som används av FMS-standard kommer inte att kunna visas, eftersom inte alla förmedlas av OBD. Främst är det data från tachografen som faller bort. Istället får man tillgång till det stora utbud av värden som OBD stödjer.

6 SAMMANFATTNING OCH KOMMENTARER

Det här kapitlet presenterar mina åsikter och slutsatser om de moment som varit del i examensarbetet. I varje delområde tas det upp några specifika saker som stått ut tydligast.

6.1 FMS-Standard

Syftet med FMS-Standard är till att ge tillgång till information om ett fordon som kan vara användbart för personer i fordonets närhet. FMS-Standard utför detta framgångsrikt och informationen som tillgängliggörs kan användas konstruktivt.

6.1.1 Dokumentation

FMS-Standard är ganska dåligt dokumenterat i sig, det refererar till J1939 och inte mycket mer. Det är dock lätt att hitta till hur man får tag i standarden, vilket är bra.

Jag anser att FMS-Standard borde beskrivas i högre detalj, att förklara noggrannare vad systemet gör och hur man interagerar med det, utan att behöva förlita sig på information från andra källor (i detta fall SAE).

6.1.2 Struktur

FMS-Standard bygger på två mycket solida teknologier. Båda dessa är konstruerade på ett bra sätt och passar väl in i användningsområdet för FMS-Standard.

6.1.3 Övervakade egenskaper

I jämförelse med OBD ger FMS-Standard tillgång till väldigt få egenskapers värden. Visserligen innehåller OBD många egenskaper med ringa betydelse, men skillnaden i omfång väcker ändå frågan om FMS-Standard ger tillgång till alla värden som skulle kunna vara användbara.

6.2 OBD

OBD har som primära uppgift att hindra personbilar från att släppa ut onödigt mycket avgaser och detta lyckas den med. Utöver detta ger OBD tillgång till information från fordonet som kan vara användbar för fordonets ägare och förare.

6.2.1 Dokumentation och struktur

Dokumentation om OBD är svåröverskådlig. Det finns massvis med dokument. Det finns två dokument, som sägs vara tekniskt ekvivalenta, som beskriver högnivå-protokollet. Dessa pekar sedan vidare på de fem olika lågnivåprotokollen som används av olika tillverkare och modeller. På vissa ställen i högnivådokumentationen, och i omkringliggande dokumentation, så särbehandlas ett eller flera av dessa fem. De fem underprotokollen omfattar ibland inte hela de refererade dokumenten, utan endast delar av dem. Det finns ytterligare dokument som

beskriver olika detaljer som inte nödvändigtvis tas upp i något annat dokument, exempelvis finns det två officiella dokument som beskriver de felkoder som används av OBD.

OBD borde inte behöva se ut som det gör. Det hade gärna fått varit mer genomtänkt från början. Det finns flera sidospår och kringdetaljer som är utdaterade men fortfarande är en del av standarden. Det bidrar till att standarden blir stor och krånglig. En enkel strukturerad beskrivning som från början hade innehållit all information som behövs (t.ex. lågnivåprotokoll) och möjlighet till utökning av protokoll och prestanda borde ha varit ursprungsdefinitionen. Detta kommer dock att bli bättre eftersom ett av lågnivåprotokollen är standard från 2008. De andra protokollen kommer därefter sakta att fasas ut allteftersom de fordon som använder dem tas ur bruk.

Det skulle behöva finnas (det kanske det gör, i sånt fall behöver den bli synligare) en organisation med ansvar för att enbart för att övervaka/utveckla/handhålla OBD. Det skulle kunna var en branschförening där fordonstillverkare och myndigheter ingår. Jag tror att det skulle göra det avsevärt lättare att hitta teknisk information om OBD om man kunde få en bra ”top-down”-bild av systemet utan att behöva leta sig upp från botten. Gärna någon förklarande text (såsom den som nu finns i rapporten).

6.3 FMSlib

Utvecklingen av FMSlib var ett intressant och givande projekt som gav tillfälle att använda kunskaper i programkonstruktion och projektplanering. Den programvara som blev resultatet av utvecklingen visar att det går att använda fordonsdiagnostik i praktiken.

6.4 Fordonsdiagnostik

Det intryck som det här examensarbetet har givit om fordonsdiagnostik som helhet är att det är väldigt användbart. Det finns vinster att göra till kostnader som är överkomliga eller rentav låga.

6.4.1 Funktioner för fordonsägare

För ett företag som äger ett flertal fordon har fordonsdiagnostik stor potential. Information om ett visst fordon kan jämföras mot andra fordon för att hitta avvikande beteenden, som indikerar att ytterligare undersökning bör utföras. På samma sätt kan fordonets förare analyseras. Värt att nämna här är att det är ett känsligare ämne att granska personer än maskiner, men det kan finnas vinster att göra med detta. Även avgasutsläpp kan beräknas, under förutsättning att fordonet hålls i gott skick.

Detta område förlorar dock potential om det inte finns en tillräcklig stor mängd fordon att jämföra, men visst värde finns alltid kvar, även för ett ensamt fordon.

6.4.2 Funktioner för förare

Många av de egenskaper som man kan få tillgång till via fordonsdiagnostik är av sådan natur att de är viktiga för föraren. Oftast är de dock så viktiga att de redan finns tillgängliga via fordonets kontrollpanel (hastighet, bränslenivå). Därför är användbarheten här begränsad till att komplettera eventuell saknad information.

En möjlighet med just information av den här naturen vore att använda informationen i ett mer avancerat övervaknings-/varningssystem placerat i fordonet. Ett sådant skulle exempelvis kunna varna föraren om hastighetsbegränsningen överskrids genom att kombinera fordonsdata med GPS och väginformation.

6.4.3 Frågor med svar

I 1.2 ställdes några frågor, här tänkte jag ge kortfattade svar på dem. Utförligare svar finns att finna i rapporten.

1. Varför vill man kommunicera med personbilar och andra fordon?
   För att komma åt information som kan vara föredelaktig att ha.
2. Vad behöver man för att kunna göra detta?
   Sensorer som mäter informationen, en dator som hanterar den och en anslutning däremellan.
3. Vilka tekniker finns inom detta område?
   OBD och FMS-Standard, samt varianter av och föregångare till dessa. Det finns antagligen även företagsspecifika system.
4. Hur ser ett dessa tekniker ut?
   Se avsnitt 3.1 resp 3.2.
5. Hur använder man dem?
   Man ser till att de är installerade i ett fordon, ansluter en dator till diagnostikanslutningen och kommunicerar enligt standarden.
6. Vad kan förmedlas av varje teknik?
   Information relaterad till fordonets tillstånd, förarens beteende och fordonets utsläpp.
7. Vad har teknikerna för positiva egenskaper man vill komma åt?
   FMS-Standard rapporterar data med hög uppdateringshastighet, och tillhandahåller många viktiga värden.
   OBD genomför felsökning nästan helt automatiskt, och tillhandahåller många viktiga värden.
8. Vad har teknikerna för negativa egenskaper man gärna undviker?
   FMS-Standard är inte obligatoriskt och kan behöva installeras. FMS-Standard övervakar kanske inte all data man kan vara intresserad av. FMS-Standard är väldigt enkelt och kräver mycket av användaren för att vara användbar.
   OBD övervakar kanske inte all data man kan vara intresserad av. OBD som standard och system är stort och krångligt.

REFERENSER

Bakgrund

[1] Gerard J. Holzmann: Design and Validation of Computer Protocols. Prentice Hall, 1991, ISBN 0-13-539925-4
[2] Global Positioning System: Theory and Applications Vol I & Vol II, Edited by Bradford W Parkinson and James J. Spilker, Volume 163 and 164, Progress in Astronautics and Aeronautics.
[3] Andrew S. Tanenbaum, Computer Networks. Prentice Hall, Upper Saddle River, NJ (2003). 892 pp. ISBN 0-13-066102-3
[4] X.200 : Information technology - Open Systems Interconnection - Basic Reference Model: The basic model
[5] Cyclic Codes for Error Detection - Peterson, W.W.; Brown, D.T. -Proceedings of the IRE, Volume 49, Issue 1, Jan. 1961 Page(s):228 - 235

OBD

[6] ”On-Board Diagnostics (OBD) | US EPA”, http://www.epa.gov/obd/ , 2009-05-03
[7] “Background Material: Frequently Asked Questions (FAQ)
About On-Board Diagnostic II (OBD II) Systems” -
http://www.arb.ca.gov/msprog/obdprog/obdfaq.htm - 2009-06-18
[8] EU Directives EC70/220, EC98/69, EC99/102, ECE R83
[9] ISO 15031-5:2006 - Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 5: Emissions-related diagnostic services
[10] ISO/DIS 15031-6 - Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 6: Diagnostic trouble code definition
[11] SAE J2012 (R) Diagnostic Trouble Code Definitions
[12] SAE J1979 (R) E/E Diagnostic Test Modes
[13] ISO 9141:1989 - Road vehicles -- Diagnostic systems -- Requirements for interchange of digital information
[14] ISO 9141-2 - Road vehicles -- Diagnostic systems -- Part 2: CARB requirements for interchange of digital information
[15] ISO 9141-3:1998 -Road vehicles -- Diagnostic systems -- Part 3: Verification of the communication between vehicle and OBD II scan tool
[16] SAE J1850 Class B Data Communications Network Interface
[17] ISO 14230-1:1999 - Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 -- Part 1: Physical layer
[18] ISO 14230-2:1999 - Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 -- Part 2: Data link layer
[19] ISO 14230-3:1999 - Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 -- Part 3: Application layer
[20] ISO 14230-4:2000 - Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 -- Part 4: Requirements for emission-related systems
[21] ISO 15765-1:2004 - Road vehicles -- Diagnostics on Controller Area Networks (CAN) -- Part 1: General information
[22] ISO 15765-2:2004 - Road vehicles -- Diagnostics on Controller Area Networks (CAN) -- Part 2: Network layer services

[23] ISO 15765-3:2004 - Road vehicles -- Diagnostics on Controller Area Networks (CAN) -- Part 3: Implementation of unified diagnostic services (UDS on CAN)
[24] ISO 15765-4:2005 - Road vehicles -- Diagnostics on Controller Area Networks (CAN) -- Part 4: Requirements for emissions-related systems
[25] J1978 - OBD II Scan Tool -- Equivalent to ISO/DIS 15031-4:December 14, 2001
[26] Intel(R) Implementing the J1850 Protocol - D. John Oliver -Intel Corporation
[27] Mode $06 Information, http://techinfo.honda.com/rjanisis/RJAAI001_mode.asp , 2009-06-18
[28] Electronics Industries Association, "EIA Standard RS-232-C Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Data Interchange", August 1969, reprinted in Telebyte Technology Data Communication Library, Greenlawn NY, 1985
[29] ISO 11898-1:2003 - Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signalling
[30] ISO 11898-2:2003 - Road vehicles -- Controller area network (CAN) -- Part 2: High-speed medium access unit
[31] ISO 11898-3:2006 - Road vehicles -- Controller area network (CAN) -- Part 3: Low-speed, fault-tolerant, medium-dependent interface
[32] ISO 11898-4:2004 - Road vehicles -- Controller area network (CAN) -- Part 4: Time-triggered communication
[33] ISO 11898-5:2007 - Road vehicles -- Controller area network (CAN) -- Part 5: High-speed medium access unit with low-power mode

FMS

[34] FMS-standard, FMS-Standard Interface description Vers. 01.00, http://fms-standard.com/, 2009-06-18
[35] Letter to European Institutes - http://fms-standard.com/down_load/letter_acea.pdf, 2009-06-18

J1939

[36] SAE J1939 Recommended Practice for a Serial Control and Communications Vehicle Network
[37] J1939-01 - Recommended Practice for Control And Communications Network for On-Highway Equipment
[38] J1939-02 - Agricultural and Forestry Off-Road Machinery Control and Communication Network
[39] J1939-03 - On Board Diagnostics Implementation Guide
[40] J1939-05 - Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
[41] J1939-11 - Physical Layer - 250k bits/s, Twisted Shielded Pair
[42] J1939-13 - Off-Board Diagnostic Connector
[43] J1939-15 - Reduced Physical Layer, 250K bits/sec, Un-Shielded Twisted Pair (UTP)
[44] J1939-21 - Data Link Layer
[45] J1939-31 - Network Layer
[46] J1939-71 - Vehicle Application Layer
[47] J1939-73 - Application Layer - Diagnostics
[48] J1939-74 - Application - Configurable Messaging
[49] J1939-75 - Application Layer - Generator Sets and Industrial

[50] J1939-81 - Network Management
[51] J1939-82 - Compliance - Truck and Bus
[52] J1939-84 - OBD Communications Compliance Test Cases for Heavy Duty Components and Vehicles
[53] J1587 - Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications
[54] J1708 : Serial Data Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications

Andra standarder

[55] Heavy Duty Vehicles | On-Board Diagnostics (OBD) | US EPA - http://www.epa.gov/obd/regtech/heavy.htm , 2009-06-18
[56] "Bus-FMS-Standard - Information about the Bus-FMS-Standard", http://bus-fms-standard.com/ , 2009-06-18
[57] Industrial communication subsystem based on ISO 11898 (CAN) for controller-device interfaces Part 4: CANopen - EN 50325-4
[58] Low-voltage switchgear and controlgear - Controller-device interfaces (CDIs) - Part 3: DeviceNet - IEC 62026-3 Ed. 2.0
[59] Maritime navigation and radiocommunication equipment and systems - Digital interfaces - Part 3: Serial data instrument network, IEC 61162-3
[60] ISO 11992-1:2003 - Road vehicles -- Interchange of digital information on electrical connections between towing and towed vehicles -- Part 1: Physical and data-link layers
[61] ISO 11992-2:2003 - Road vehicles -- Interchange of digital information on electrical connections between towing and towed vehicles -- Part 2: Application layer for brakes and running gear
[62] ISO 11992-3:2003 - Road vehicles -- Interchange of digital information on electrical connections between towing and towed vehicles -- Part 3: Application layer for equipment other than brakes and running gear
[63] ISO 11992-4:2005 - Road vehicles -- Interchange of digital information on electrical connections between towing and towed vehicles -- Part 4: Diagnostics

Företag, organisationer och myndigheter:

[64] "CC Systems" - www.ccsystems.se - 2009-06-18
[65] "Bilprovningen" - http://www.bilprovningen.se/ - 2009-05-05
[66] "California Air Resource Board" - http://www.arb.ca.gov/homepage.htm , 2009-06-18
[67] European Automobile Manufacturers' Association - http://www.acea.be/ , 2009-06-18

Övrigt:

[68] "Uppsala Universitet – Kursplan Examensarbete DV3", www.selma.uu.se/publik/main?AF=0200&funktion=kplan&kurs=1DT343 , 2009-06-18
[69] "CC Pilot XS" , http://www.cc-systems.com/Portals/0/PDF/Products/Leaflet/CCPilotXS2009.pdf, 2009-06-18