

Fordonsdiagnostik mot bandvagnar

Av: Anton Rytterstedt och Mattias Viberg

2006-09-29

Sammanfattning

Under åttiotalets början startade fordonstillverkare kontrollera motorns funktioner och diagnosticera motorproblem elektroniskt. Anledningen till detta var främst de hårdare miljökraven. Varje fordonstillverkare utvecklade sitt eget system för On-Board Diagnostics (OBD) och ett behov av en gemensam standard uppstod. I mitten av nittioalet släppte en miljöorganisation från Kalifornien, Air Resource Board en standard kallad OBD-II. Standarden erbjuder närmast komplett kontroll över motorn och övervakar även delar av chassit och karossen samtidigt som den hanterar diagnostik för fordonet. OBD-II har haft stor genomslagskraft och är numera allmänt gällande i USA och Europa.

Syftet med detta examensarbete är att undersöka huruvida BVS10, en bandvagn tillverkad av BAE Systems Hägglunds AB i Örnsköldsvik, har stöd för denna standard. Efter utredningen skulle en applikation för att läsa ut och hantera diagnostisk information från bandvagnens motor och växellåda utvecklas, antingen genom att använda OBD-II eller något liknande alternativ.

Under undersökningen kom det fram att BVS10 inte hade stöd för OBD-II men eftersom det fanns stöd för ett av kommunikationsprotokollen hos OBD-II så fanns möjligheten att implementera en diagnostisk applikation. Denna diagnosapplikation riktad mot BVS10 har integrerats i Hägglunds befintliga diagnosprogramvara DIS (Diagnostic Information System). DIS erbjöd i starten av examensarbetet endast diagnostisering av Hägglunds stridsfordon (CV90), efter avslutat arbete har även stöd för diagnosticering av BVS10 byggts in i DIS applikationen. Det utökade stödet leder till ett mervärde hos DIS produkten eftersom kunder som köper stridsfordon också köper bandvagnar och nu kan diagnosticera dessa med samma programvara.

Abstract

During the early eighties vehicle manufacturers started to control the engine functions and diagnose engine problems electronically. The reason for this was the harder environmental legislations. Every vehicle manufacturer developed their own system for On-Board Diagnostics (OBD) and a need for a common standard rised. In the mid ninties a environmental organisation from California, Air Resource Board, released standard called OBD-II. This standard offer full control over the engine and monitor parts of the chassis and body as well as the diagnostic control network of the vehicle.

The purpose of this theis is to examine whether BVS10, an all-terrain vehicle manufactured by BAE System Hägglunds AB in Örnsköldsvik, supports this standard. After this investigation an application for reading and handling diagnostic information from the engine and the gearbox of the vehicle should be implemented . Either by using OBD-II or some similar alternative standard.

During the investigation it emerged that BVS10 did not have support for OBD-II, but because it supported a communications protocol supported by OBD-II, it was possible to implement a diagnostic application. This diagnostic application for BVS10 was integrated into Hägglunds existing diagnostic application DIS (Diagnostic Information System). At the start of the project DIS only offered possibility to run diagnostics on Hägglunds combatvehicles (CV90), after finished work DIS now also supports diagnostics for BVS10. The extended support for diagnostics makes the DIS product more valuable because customers that buys comat vehicles also buys BVS10 and now kan use DIS for diagnostics on both of the vehicles.

Innehållsförteckning

1 Inledning	9
1.1 Terminologi	9
1.2 Rapportens disposition.....	10
2 Problemspecifikation	12
3 Metod.....	14
4 Diagnossystem för motorfordon.....	15
4.1 Bakgrund.....	15
4.2 OBD-II.....	17
4.2.1 Hur OBD-II systemet fungerar	17
4.2.2 OBD-II systemets uppbyggnad.....	18
4.2.3 Felkoder för OBD-II system	20
4.2.4 Moder	22
4.2.5 Kommunikationsprotokoll för OBD-II	23
4.3 HD OBD	28
4.4 WWH OBD	29
5 Befintlig lösning	30
5.1 Nuvarande funktionalitet	33
5.2 Fordonsdatabas	36
6 Implementation	38
6.1 Krav.....	38
6.2 Design.....	38
6.3 Systembeskrivning	40
6.3.1 J1939Transport	40
6.3.2 J1939SignalValues.....	46
6.3.3 J1939Diagnosis	48
6.5 Användargränssnitt	53
6.5.1 Diagnostik.....	53
6.5.2 Signalhantering.....	57
6.6 Lösningens begränsningar	58
7 Framtida arbete	60
8 Diskussion	62
9 Slutsats.....	65
Referenser.....	66
Appendix A – ISO 11898 (CAN)	68
Dokumentstruktur.....	68
Datalänklagret.....	68
Meddelandeformat	69
Felhantering	69
Det fysiska lagret	70
Appendix B – SAE J1939	71
Dokumentstruktur.....	71
Meddelandeformat	72
J1939-PDU	72
Meddelandetyper	73
Command.....	73

Request	73
Response	74
Acknowledgement	74
Kommunikationstyper.....	74
Ta emot meddelanden.....	74
Transport.....	75
Flerpakets broadcast.....	75
Uppkoppling	76
Fördefinierade värden.....	77
Parametergruppnummer (PGN)	78
Datafältsgruppering	78
Industrigrupp	78
Adresser	79
Appendix C – ISO 15765.....	80
Dokumentstruktur.....	80
Nätverkslagret.....	81
Tjänster till de övre lagren	82
Interna operationer	82
Applikationslagret.....	83
Tjänster till det övre lagret	83
Appendix D– ISO 9141	85
Appendix E– Krav på OBD-II systemets kontakt	90
Appendix F – Krav på extern diagnosenhet.....	92
Appendix G – Förslag till design av J1939 meddelande och signal i FDB.....	94

1 Inledning

Denna rapport är en del i ett examensarbete utfört vid BAE Systems Hägglunds AB i Örnsköldsvik. Examensarbetet är inriktat mot fordonsdiagnostik för bandvagnar genom användandet av standarden OBD-II (On-Board Diagnostics). Standarden är den andra generationen av en världsomfattande standard som funnits sedan slutet av 80-talet men fortsatt att utvecklas under åren. Idag används OBD-II enligt lag i alla ny tillverkade bilar för att minska avgasutsläpp samt att erbjuda diagnostisering och felsökning. Syftet med examensarbetet är att utreda om standarden OBD-II lämpar sig, samt går att använda för diagnostisering av bandvagnar. Efter utredningen ska en applikation implementeras som erbjuder diagnostik för bandvagnar.

1.2 Terminologi

Rapporten innehåller en mängd termer och förkortningar som för den ej insatte kan verka förvirrande. Vi har strävat efter att beskriva var och ett av dessa begrepp första gången vi använder dessa men för att det ska vara lätt att vid behov slå upp ett begrepp eller en förkortning presenterar vi de viktigaste här

ARB – California Air Resource Board, en miljöorganisation lokaliserad i Kalifornien. Arbetar med att standardisera OBD.

Allison – Tillverkare av växellådor.

BV206S – Bandvagn tillverkad av Hägglunds, den mindre modellen.

BVS10 – Bandvagn tillverkad av Hägglunds, den större modellen.

CAN – Controller Area Network, Ett nätverk för att kommunicera med olika noder hos ett fordon.

CV90 – Stridsfordon 90, tillverkad av Hägglunds.

Cummins – En motortillverkare som förser bland annat BVS10 med motorer.

DIS – Program för att läsa diagnostisk data ur stridsfordon.

DTC – Diagnostisk Trouble Code, diagnostiska felkoder som lagras i fordon och läses ut vid diagnos av detsamma.

DTB – DIS Test Box, Den hårdvara som krävs för att kunna koppla en dator med DIS till ett fordon.

EPA – Environmental Protection Agency, Amerikansk miljöorganisation, standardiserar OBD.

FMI – Fault Mode Identifier, talar om vad som orsakat en felkod som registrerats i fordonet, det vill säga om ett övre eller undre gränsvärde överskridits eller liknande.

Freezeframe – Information som lagras när en felkod registreras, innehåller viktig information om motorn vid det aktuella feltilfället.

HD OBD – En ny standard som inriktas mot diagnostik för tyngre fordon.

ISO 15765 – En standard för diagnostik mot fordon över CAN, tillåtet protokoll enligt OBD-II.

KWP2000 - En standard för diagnostik mot fordon över CAN, tillåtet protokoll enligt OBD-II.

MIL – Malfunction Identification Lamp, den lampa som tänds när fordonet registrerat ett allvarligt fel. Kallas även ”Check engine”-lampan.

OBD – On-board Diagnostics, system för att utläsa diagnostisk information ur fordon.

OBD-II – En OBD-standard riktad mot lättare fordon, såsom bilar och lätta lastbilar.

PGN – Parameter Group Number, identifierar ett meddelande som skickas enligt J1939.

SAE J1939 – En standard för kommunikation över ett fordons CAN-bus. Stöds genom en undantagsregel i OBD-II.

SPN – Suspect Problem Number, se DTC.

Steyr – Motortillverkare som förser BV206S med motorer.

WWH OBD – En standard som befinner sig i utvecklingsstadiet, är tänkt att ersätta OBD-II och HD OBD.

1.2 Rapportens disposition

Nedan ges en kort beskrivning över rapportens disposition. Utöver de nio kapitlen så finns även appendix A-F där vissa ämnen beskrivs mer ingående.

Kapitel 2 ger en kortare introduktion till företaget. Bakgrunden samt målen för examensarbetet presenteras.

Kapitel 3 beskriver metoden och tillvägagångssättet för examensarbetet.

Kapitel 4 ger en introduktion till standarden OBD-II vilken examensarbetet är inriktat mot.

Kapitel 5 presenterar den befintliga lösningen som Hägglunds idag använder för diagnostisering av stridsfordon.

Kapitel 6 presenterar resultatet av examensarbetet. Här beskrivs hur applikationen för diagnostisering av bandvagnar implementerats.

Kapitel 7 beskriver möjligt framtida arbete, här adresseras de brister som togs upp under lösningens begränsningar.

Kapitel 8 diskuterar kring de problem som uppkommit under utvecklingen av applikationen samt reflekterar över den förvirring som råder inom OBD i dagsläget.

Kapitel 9 är en slutsats där vi återkopplar mot de frågor och mål som sätts upp i kapitel 2.

2 Problemspecifikation

OBD-II är andra generationen av ett system kallat "On-Board Diagnostics". Systemet är kravspecificerat och framställt av *California Air Resource Board* (ARB), det primära målet med systemet är att minska luftföroreningar orsakade av avgasutsläpp från bilar. Systemet används även för diagnostisering och felsökning av fordon eftersom det övervakar komponenter och funktioner samt indikerar när dessa inte fungerar korrekt.

Hägglunds har utvecklat ett eget diagnossystem kallat *Diagnostic Information System* (DIS), för att övervaka sensorerna som existerar i deras CV90. DIS systemet utnyttjar den inbyggda CAN-bussen i fordonen som kopplar samman all elektronik och sedan kan sensordata läsas ut genom inkoppling av en testbox kallad *DIS Test Box* (DTB). Testboxen kopplas in direkt på CAN-bussen och sedan kopplas testboxen vidare till en dator för att kunna köra mjukvaran DIS och få tillgång till de olika funktionerna som erbjuds. DIS erbjuder förutom diagnostik av fordonet även möjligheter till fjärrstyrning, dessa har dock begränsats av säkerhetsskäl. Några av de grundläggande tankarna bakom OBD-II systemen finns alltså tillgängliga i stridsfordonens hårdvara samt via funktionaliteten som DIS erbjuder.

DIS-systemet är endast inriktat på diagnos av CV90. Bandvagnarna som är Hägglunds andra produktkategori, skild från stridsfordonen, saknar i dagsläget de diagnosmöjligheter som DIS erbjuder. Hägglunds är intresserade av att undersöka vad som idag saknas i bandvagnarna för att kunna diagnosticera dessa med en form av OBD-II-system, eller liknande. Efter utredning ska DIS-applikationen anpassas för att kunna läsa av diagnostisk data från bandvagnarna. Motivationen till detta är att det leder till ett mervärde hos DIS eftersom kunderna utöver att diagnostisera CV90 även kan diagnostisera bandvagnar (de kunder som beställer systemet använder i regel också bandvagnar).

Examensarbetet delas in i två skilda delar en fördjupningsstudie samt en implementation av ett diagnossystem för bandvagn. Fördjupningsstudien kommer att ske mot OBD-II. Systemets uppbyggnad samt dess olika kommunikationsprotokoll kommer att analyseras och jämföras. Studien utreder även om det finns andra existerande lösningar än OBD-II. Exakt hur implementationen kommer att ske, bestäms efter att fördjupningsstudien är genomförd. Ifall det är möjligt ska stöd för OBD-II implementeras i det befintliga DIS systemet. Om detta ej är möjligt ska en frståående

applikation som stödjer läsning av OBD-II information från bandvagn utvecklas. Från Hägglunds sida är man främst intresserade att använda OBD-II för att läsa ut information från motor och växellåda i bandvagnarna. Ifall tid finns tillgänglig efter detta så finns även en önskan att utreda och utveckla ytterligare stöd för något eller några protokoll specificerade av standarden OBD-II.

Nedan presenteras frågor som anses viktiga för examensarbetet:

- Vad är OBD-II?
- Vilken funktionalitet erbjuder ett OBD-II-system?
- Hur är ett OBD-II-system uppbyggt?
- Vilka krav ställs för att ett system ska få klassificeras som ett OBD-II system?
- Finns det andra existerande standarder för system liknande OBD-II?
- Uppfyller DIS idag OBD-II standarden? Om inte vad saknas för att uppfylla den?
- Går det att använda OBD-II till att läsa information från motor och växellåda i bandvagn?
- Uppfyller hårdvaran bandvagnarna specifikationerna för OBD-II standarden?

Efter avslutat exjobb så är tanken att följande mål ha nåtts:

1. Fördjupning i OBD-II standarden samt andra alternativ ifall sådana existerar.
2. Utrett vad som saknas i DIS eller om DIS redan uppfyller OBD-II standarden
3. Utvecklat mjukvara för läsning av information från motor och växellåda i bandvagn.
4. Om tid finns tillgänglig ska ytterligare stöd utvecklas för något eller några protokoll specificerade av OBD-II standarden.

3 Metod

Arbetet påbörjades med en fördjupning inom OBD för att skapa en bild om hur standarden fungerade och vilka krav den ställde på mjukvara och hårdvara. Efter detta undersöktes de specifika standarderna som tillsammans bildar OBD-II. När kunskap skaffats kring OBD-II studerades Hägglunds nuvarande lösning för att få en bild om hur mycket denna skiljde sig från OBD-II-standarderna.

Efter detta studerades den bandvagn som det först var tänkt arbetet skulle inrikta sig mot, BV206S. Detta i syfte att försöka ta reda på hur elektroniken i den kommunicerade, vilka protokoll som användes etc. Detta visade sig dock vara en inte helt enkel uppgift då det bland de sakkunniga på Hägglunds rådde en del förvirring vilka standarder som stöddes av motor och växellåda på BV206S. Tillslut kontaktades motortillverkaren Steyr i Österrike som förser BV206S med motorer för att ta reda på vilka protokoll de använde sig av. Det visade sig att motorn i BV206S använde sig utav ISO 9141 protokollet, samt att man utvecklat ett eget diagnosverktyg som kommunicerade med motorn genom detta protokoll. Den person som var ansvarig för diagnostiken hos Steyr var bortrest under den period och för att examensarbetet skulle kunna fortsätta behövdes uppgifter kring vilken version av ISO 9141 som användes. Då detta drog ut på tiden diskuterades fortsättningen med handledaren på Hägglunds och tillsammans med chefen på avdelningen, Niclas Lindberg, bestämdes att det vore bäst om arbetet istället riktades mot den nyare bandvagnen, BVS10.

Efter detta kunde examensarbetet fortsätta tack vare att diagnostiken på BVS10 var betydligt bättre dokumenterad än BV206. Implementationen av applikationen började med att ta fram en enkel loggningsapplikation som läste ut information från bandvagnen. Detta för att säkerställa att vissa antaganden kring kommunikationen hos fordonet var korrekta. Därefter togs ett designförslag fram som presenterades för handledaren. Efter detta varvades implementeringar med testkörningar fram till dess att applikationen ansågs färdig och uppfyllde de huvudsakliga målen för examensarbetet.

4 Diagnossystem för motorfordon

Syftet med detta kapitel är att ge insikt i utvecklingen inom diagnostiska system för motorfordon, så kallade OBD system. Kapitlet inleder med en enklare bakgrund till den utveckling som skett och övergår sedan till att beskriva de standarder och bestämmelser som finns idag, med tonvikten på världsstandarden OBD-II. Kapitlet behandlar även de olika kommunikationsprotokoll som OBD-II standarden stödjer. Vidare behandlas standarden HD OBD som är tänkt att vara OBD-II:s motsvarighet för tyngre fordon samt WWH OBD som är ett tänkbart alternativ till HD OBD.

4.1 Bakgrund

I mitten av 80-talet införde USA som ett av de första länderna lagar mot luftföroreningar orsakade av avgasutsläpp från fordon. Det var framförallt delstaten Kalifornien som var drivande bakom denna lagstiftning. Resultatet av lagstiftningen blev att biltillverkare tvingades att införa diagnossystem i sina fordon. Dessa system kallades för OBD (On-Board Diagnostics). Den första versionen av OBD utvecklades 1985 av ARB (Air Resource Board) som svar på den rådande lagstiftningen, och togs i bruk 1988. Alla fordon med en totalvikt under 6,5 ton som tillverkats efter 1988 skulle vara utrustade med någon typ av OBD-system. I många andra länder har kostnaden lagts på fordonsägaren genom att införa olika former av skatt på ”rena” och ”smutsiga” fordon, USA har istället flyttat över ansvaret till fordonstillverkarna. [1]

Första generationens OBD system, bestod av enkla elektroniska övervakningssystem som främst användes för att kontrollera fordonets avgasutsläpp och motorfunktioner. Systemets uppgift var att övervaka de olika funktioner och komponenter i bilen som kan orsaka höga avgasutsläpp och skada fordonets motor ifall de inte fungerar korrekt. Ifall diagnossystemet upptäcker att utsläppen var högre än den tillåtna gränsen eller om ett allvarigare motorfel misstänks så informeras föraren om detta och kan lämna in fordonet för service.

Den första versionen av OBD-system följde ingen specifik standard utan tillät att tillverkarna själva utformade diagnossystemen. Tillverkarna använde sig av olika kontakter för inkoppling mot fordonsdatorn, samt skilda kommunikationsprotokoll. Resultatet av detta blev bland annat att felkoderna från OBD-systemen tolkades på olika sätt. Men den största nackdelen med de tidiga

OBD-systemen var att tillverkarna hade olika uppfattning om hur stor avvikelse en komponent tilläts ha innan systemet rapporterade komponenten som defekt. Detta ledde till att ett OBD-system kunde informera fordonsägaren att en komponent var defekt och ökade avgasutsläppen, medan ett annat system inte rapporterade samma typ av fel. Eftersom diagnosutrustningen också var tillverkarspecifik samt dyr behövde en fordonsägare alltid uppsöka en märkesverkstad för att få sitt fordon diagnostiserat.

1989 hade ARB vidareutvecklat OBD-systemet och ersatt det med det SAE (Society of Automotive Engineers) standardiserade OBD-II systemet i ett försök att komma ifrån de tillverkarspecifika systemen som existerade. Ett beslut togs att alla bilar sålda 1996 och senare i USA skulle vara OBD-II kompatibla. Detta gällde alla bilar, oavsett om de tillverkats i USA eller importerats. Enligt ett EU direktiv daterat 1998 måste fordonstillverkare i Europa också vara tvungna att införa diagnosystem enligt modellen som används i USA. Riktlinjerna från EU gäller tändstiftsmotorer (bensinmotorer) registrerade år 2000 och senare, samt självtändande motorer (dieselmotorer) registrerade år 2003 och senare.

Utvecklingen mot OBD-II var först och främst inriktad mot diagnostik för lättare fordon med en totalvikt på under 6,5 ton. Parallellt med detta utvecklades en standard för kommunikation hos systemen inuti tyngre fordon. 1998 publicerade SAE J1939 en samling specifikationer för att uppfylla just detta. ARB har anammat denna standard i sin OBD-II specifikation där man säger att mellanklassade fordon med en totalvikt mellan X och Y får använda sig utav J1939 som underliggande protokoll.

ARB påbörjade även ett arbete med att skapa en bestämmelse om OBD för tyngre fordon. Denna bestämmelse kallas Heavy-Duty OBD (HD OBD) och trädde i kraft den 17e Mars i år. Parallellt med ARB:s arbete med att utveckla en bestämmelse för tyngre fordon har även en organisation kallad Word Wide Harmonization (WWH) startat en utredning under namnet Ad Hoc WWH Single OBD HD Protocol Task Force där möjligheten att ta fram en generell OBD-standard tänkt att gälla alla fordon undersöks.

4.2 OBD-II

OBD-II är andra generationen av det diagnosystem (OBD) som ARB utvecklade. Införandet av denna nya standard innebar att problemen med alla tidigare tillverkarspecifika OBD-system reducerades. Standarden tvingar alla tillverkare att använda samma kontakt, samma tolkning av felkoder samt att man specificerar gränsvärden för komponenter och avgasutsläpp.

Enligt den nya standarden har ett OBD-II system följande huvuduppgifter:

1. Att minska höga utsläpp, som beror på att funktioner eller komponenter i fordonet inte fungerar som de ska.
2. Att reducera tiden mellan uppkomsten av ett fel, detektion av felet och reparation av komponenten/funktionen.
3. Att bistå mekaniker i diagnos av fordon och reparation av komponenter/funktioner.

OBD-II systemet sköter dessa genom att övervaka varje enskild komponent och funktion som kan ha inverkan på fordonets avgasutsläpp. Ifall systemet upptäcker ett att en komponent eller funktion faller utanför sina gränsvärden och påverkar avgasutsläppen eller riskerar skada på fordonets motor, så tänds en lampa i instrumentpanelen kallad MIL (Malfunction Indicator Light). Samtidigt som felet upptäcks lagras också felkoder (beskriver vad som är fel) och en *freezeframe* i fordonet. En freezeframe är en samling av information som systemet läste av i precis det ögonblick ett fel hos en komponent eller funktion upptäcktes. I en freezeframe lagras t.ex. varvtal, hastighet, kylvatten temperatur med mera för att bistå mekaniker att lokalisera och reparera komponenten/funktionen.

4.2.1 Hur OBD-II systemet fungerar

När tändningen på fordonet vrids på ska MIL tändas för att sedan släckas, detta indikerar att OBD-II systemet är redo att genomföra en felsökning av fordonet. Efter MIL släckts vid starten av fordonet ska lampan fortsätta att vara släckt under körning förutsatt att inget fel upptäcks. Ifall MIL tänds under körning så indikerar detta att OBD-II systemet har upptäckt ett problem. Problemet kan vara allt ifrån att fordonet drar onormalt med bensin, att motorn misständer eller att onormalt höga avgasutsläpp har detekterats. Om MIL däremot skulle blinka så betyder det att fordonsdatorn har

upptäckt ett allvarligt fel hos någon komponent som kan skada motorn och att felet bör undersökas och åtgärdas direkt. Fordonet kan fortfarande framföras men föraren bör minimera körsträckan och försöka att inte belasta motorn onormalt genom höga farter eller att dra tung last. Föraren bör uppsöka en verkstad snarast. Om MIL lampan skulle slockna under körning så indikerar detta att problemet som fick lampan att tändas har hanterats av OBD-II systemet. Att MIL släckts är inte en indikation på att OBD-II systemet slutat fungera utan att det troligen rörde sig om ett temporärt problem som fordonsdatorn lyckades åtgärda, fordonet behöver ingen speciell service förutsatt att inte lampan tänds igen.

En verkstad som använder sig av ett diagnostiseringsverktyg som stöder OBD-II standarden, kan lätt koppla in sig mot bilens OBD-II kontakt och ladda ner information från fordonsdatorn angående vilka komponenter och funktioner som är defekta. Detta leder till en snabbare felsökning jämfört med traditionella metoder och förhoppningsvis kan kostnaden för reparationer hållas nere.

4.2.2 OBD-II systemets uppbyggnad

Nedan kommer en beskrivning av hur OBD-II systemet är uppbyggt samt, en överblick av dess viktigaste funktioner att ges.

Själva hjärnan i ett OBD-II system är den så kallade fordonsdatorn. Fordonsdatorns främsta uppgifter är att styra tändning, bränsletillförsel och övervaka fordonets alla system. Användningen av en fordonsdator erbjuder fordonsägaren god bränsleekonomi, minimala utsläpp samt en säkerhet eftersom den övervakar motorns kritiska funktioner.

För att fordonsdatorn överhuvudtaget ska kunna styra och reglera motorn i fordonet krävs att den har tillgång till information från flertalet sensorer. Sensorer är ofta givare som omvandlar motorns fysiska värden till elektriska signaler som OBD-II systemet kan tolka. Exempel på fysiska värden som omvandlas är t.ex. temperatur, luftflöde, bränsleflöde och motorvarvtal.

Förutom det program i fordonsdatorn som reglerar fordonets motor krävs också enligt OBD-II standarden att det finns övervakningsfunktioner inbyggda i systemet. Dessa funktioner kallas för monitorer och delas in i två typer, kontinuerliga och icke-kontinuerliga. En kontinuerlig

övervakningsfunktion ska alltid vara verksam då fordonet befinner sig i ett normalt drifttillstånd. Funktioner eller monitorer som är icke-kontinuerliga löper tills den datamängd som önskas insamlas har uppnåtts eller för ett visst förprogrammerat tidsintervall. Båda dessa typer av monitorer används för att testa komponenter med avsikt att detektera de komponenter som levererar felaktig information. Ifall en komponent testats och resultatet ligger utanför de tillåtna värdena tänds MIL på fordonets instrumentpanel och felkod samt freeze-frame lagras i fordonsdatorns minne. Enligt OBD-II standarden krävs följande tre typer av kontinuerlig övervakning.

Komponentövervakning – Testsystem för motorkomponenter som inte kontinuerligt övervakas av någon av de övriga monitorerna. Komponenterna som övervakas är de som påverkar motorn om fel eller försämring av komponenten sker.

Bränslesystemövervakning – Denna monitor används för motorns adaptiva bränslekontrollsystem. I en bensinmotor ska förhållandet mellan bränsle och luft vara 1:14.7 för att ge optimal förbränning. Detta värde är det bränsleövervakningssystemet strävar att uppnå.

Misständningsövervakning – Detekterar om misständningar som är skadliga för motorn uppstår. Förutom tändsystemet övervakas också andra system som kan ge misständningar t.ex. EGR-system, felaktig bränsleblandning etc.

Nedan beskrivs några monitorer som används för icke-kontinuerlig övervakning. Endast en delmängd av dessa behöver implementeras enligt OBD-II standarden.

EGR-övervakning (Exhaust Gas Recirculation) – Övervakar EGR-ventilen och gasflödet från EGR.

Lambdasondövervakning – Den viktigaste av alla monitorer. Informerar fordonsdatorn om hur mycket bränsle som förbränts eller inte förbränts.

Sekundärt luftsystem – Systemet består av en pump som matar luft till katalysatorn under uppvärmningen. Används för att katalysatorn fortare ska nå arbetstemperatur och starta med rening av avgasutsläppen. Monitorn övervakar uppvärmningssystemets funktion.

Lambdasondfövärmning – Vissa lambdasonder har ett inbyggt värmeelement som ser till att lambdasonden vid kallstart snabbt når upp till arbetstemperatur. Monitorn övervakar fövärmningen.

Katalysatoruppvärmning – Övervakningen kontrollerar effektiviteten på katalysatorns uppvärmningscykel. Ser till att katalysatorn når upp till arbetstemperatur inom rimlig tid.

Katalysatorövervakning – Övervakar katalysatorns effektivitet. Sker genom jämförelse av två lambdasonder, en före och en efter katalysatorn.

Bränsleavdunstningssystem – Dagens bilar är utrustade med slutna bränslesystem för att inte ånga från bensinen ska nå ut i omgivningen. Då bilen är avstängd samlas ångorna från tanken upp i en behållare och vid start av fordonet dras ångorna in i motorn och förbränns. Detta övervakningssystem kontrollerar gastrycket så att inte bränsletanken exploderar vid temperaturförändringar, även gasläckage vid tätningar och kopplingar övervakas så att inte bränsleledningen läcker.

Vevhusventilation – Övervakar den backventil som finns i systemet och dess ledningar.

Luftkonditionering – Övervakar fordonets AC genom att kontrollera ändringar i motorbelastning då AC systemets kompressor kopplas in.

4.2.3 Felkoder för OBD-II system

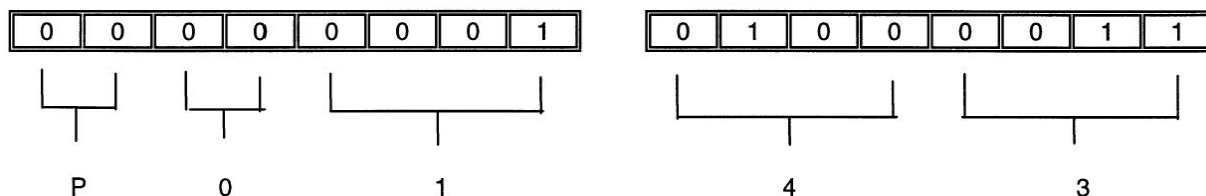
Felkoderna eller DTC (Diagnostic Trouble Codes) består alltid av fem tecken, där det första tecknet är en bokstav följt av fyra siffror. Bokstaven i felkoden kan bara vara en av följande bokstäver P, C, B eller U.[2] Betydelsen är följande:

P – Står för motor och transmission (Powertrain). Felkoder relaterade till motor eller kraftöverföring börjar med denna bokstav.

C – Står för chassi (Chassis). Felkoder relaterade till fordonets chassi t.ex. bromsar och hjul börjar med denna bokstav.

B – Står för kaross (Body). Felkoder relaterade till karossen t.ex. instrumentpanelen börjar med denna bokstav.

U – Står för datanätverk (Network). Felkoder relaterade till fordonsdatorn eller kommunikationssystemet börjar med denna bokstav.



Figur 1. Hur felkoden tolkas. [2]

Dessa fyra bokstäver tillsammans med första siffran i felkoden är kodade enligt tabellen nedan. Första siffran avgör om det är en standard eller tillverkarspecifik felkod. Anledningen till tillverkarspecifika felkoder existerar är resultatet av att tillverkare redan hade hunnit införa egna koder innan OBD-II standarden trädde i kraft. De reserverade koderna är till för framtida utbyggnad av systemet.

Hexkod	Identifierare	Betydelse
0	P0	Motor/Transmission, SAE/ISO – definierad
1	P1	” , Tillverkardefinierad
2	P2	” , Tillverkardefinierad
3	P3	” , Reserverad
4	C0	Chassikod, SAE/ISO – definierad
5	C1	” , Tillverkardefinierad
6	C2	” , Tillverkardefinierad
7	C3	” , Reserverad
8	B0	Karosskod, SAE/ISO – definierad
9	B1	” , Tillverkardefinierad
A	B2	” , Tillverkardefinierad
B	B3	” , Reserverad

C	C0	Nätverk, SAE/ISO – definierad
D	C1	” , Tillverkardefinierad
E	C2	” , Tillverkardefinierad
F	C3	” , Reserverad

Tabell 1. Tolkning av felkodsidentifierare.

Andra siffran i felkoden ger en indikation vilket system felet berör. Tredje och fjärde siffran i koden hör ihop och specificerar vad som är fel. En felkodstabell används för att omvandla felkoden till klartext.

4.2.4 Moder

Enligt OBD-II standarden ska kommunikationen använda sig av nio olika moder. [3] De moder som systemet ska stödja är följande:

Mod 01: Realtidsdata

Aktivering av detta mod tillåter att aktuell driftdata hämtats från fordonsdatorn t.ex. varvtal, hastighet, motortemperatur.

Mod 02: Freezeframe data

Vid aktivering av denna mod kan diagnosenheten hämta information lagrad i fordonet. Informationen i en freezeframe lagrades i samma ögonblick som ett fel detekterades och MIL tändes.

Mod 03: Lagrade felkoder

Låter diagnosenheten läsa ut felkoder som finns lagrade i systemet.

Mod 04: Systemåterställning av fordonsdator

Detta mod tillåter att MIL återställs, samt att felkoder och freezeframes raderas. Även fordonsdatorns egna kalibreringsvärden nollställs.

Mod 05: Resultat av lambdasondens självttest

Hämtar information om testresultaten från lambdasondövervakningen. Är ej ett krav enligt OBD-II standarden och saknas därför i många fordonsdatorer. [3]

Mod 06: Testresultat och gränsvärden

Returnerar testresultat och gränsvärden från de övervakningssystem (monitorer) som finns implementerade. Se tabell.

Mod 07: Temporärt lagrade felkoder

Dessa felkoder har samma definition som permanent lagrade felkoder. När ett fel upptäcks så lagras fordonsdatorn koden här och räknar sedan gångerna felet återkommit och lagrats. Om felet inte uppstår igen under ett fördefinierat antal körningar kommer räknaren minska och ifall den når noll kommer den temporära felkoden att raderas. Skulle felkoden däremot uppkomma igen under flera körningar tolkas felet som statiskt och felkoden lagras permanent samt att MIL tänds. Diagnosenheten kan sedan läsa ut denna felkod genom att använda sig av mod 3. Hur många gånger ett fel kan lagras temporärt utan att MIL tänds beror helt på vilken typ av fel det är. Tanken med detta mod 7 är att möjliggöra en kontroll direkt efter reparation.

Mod 08: Kontrollmod

Möjliggör att via yttre utrustning kalibrera fordonsdatorns system, utföra tester och kontrollera komponenter.

Mod 09: Fordonsdata

Hämtar data för det aktuella fordonet, t.ex. fordonets serienummer (VIN)

4.2.5 Kommunikationsprotokoll för OBD-II

ARB lyckades inte helt skapa en enhetlig standard med införandet av OBD-II. Eftersom de tidiga OBD-systemen var tillverkarspecifika så hade fordonstillverkare använt sig av olika kommunikationsprotokoll. För att kunna genomföra en OBD-II standardisering var man tvungen att tillåta tillverkarna att använda sina befintliga kommunikationsprotokoll. Resultatet blev att de största och mest använda protokollen tillåts i OBD-II standarden. De protokoll som stöds i dagsläget är:

1. **SAE J1850**

Det finns två protokoll som följer J1850-standarden, nämligen VPW och PWM.

VPW (Variable Pulse Width) är ett SAE J1850-certificerat protokoll som primärt används av GM och Chrysler. Använder sig av en kabel för kommunikation och överföringshastigheten är 10.4 kbps. *PWM* (Pulse Width Modulation) är liksom VPW ett protokoll som specificeras i SAE J1850. Protokollet kallas även SCP (Standard Corporate Protocol) och används främst av Ford men även i vissa bilar från Mazda. Protokollet har en överföringshastighet på 41.6 kbps och använder två kablar för överföring av data. [4]

2. **ISO 9141**

ISO 9141 är det protokoll som europeiska och asiatiska biltillverkare i regel använder sig av. Den första definitionen kom redan 1989 och erbjöd diagnostisk kommunikation i en hastighet mellan 10 och 10000 bps. Kommunikationen påbörjas med en initiering med hastigheten 5bps. Den första specifikationen stödjer ej OBD2, och därför kom det 1994 en ny specifikation kallad ISO 9141-2 där nya krav infördes för att möta OBD2-standarden. Enligt ISO 9141-2 specifikationen bestäms hastigheten till 10.4 kbps, men för att uppnå bakåtkompatibilitet mot ISO 9141 har man kvar initieringsfasen på 5 bps. ISO 9141-2 hanterar till skillnad från sin föregångare endast 12V-system och stöd för 24V som i ISO 9141 erbjuder existerar inte längre. Det finns även en tredje upplaga av specifikationen som rekommenderar testmetoder för att säkerställa att ett fordon eller en OBD-II diagnosenheter kan kommunicera enligt ISO 9141-2. Den testar dock ej enskilda delar utav ISO 9141-2 utan det är bara kommunikationen mellan ISO 9141-2 och en diagnosenheter som avses. [5,6] För mer utförlig information kring ISO 9141 och exempel på hur kommunikationen är uppbyggd, se Appendix F.

3. **ISO 14230**

ISO 14230 är en specifikation i tre delar som specificerar ett protokoll mycket likt ISO 9141-2. Specifikationens tre delar beskriver det fysiska lagret, länklagret samt applikationslagret. Protokollet kallas allmänt för Keyword Protocol 2000 (KWP2000). Den största skillnaden jämfört med ISO 9141-2 är att man har en snabbare initieringsmetod då man ej behöver uppnå bakåtkompatibilitet mot ISO 9141. Dessutom har KWP2000 i likhet med ISO 9141 men till skillnad mot ISO 9141-2 stöd för både 12 och 24V-system.

4. ISO 15765

ISO 15765 är en specifikation från 1999 i fyra delar som beskriver diagnostik över CAN-bussen. 15765-1 är en översikt över hela specifikationen, 15765-2 specificerar nätverkslagret och 15765-3 beskriver applikationslagret. De tre inledande delarna erbjuder inget stöd för OBD-II utan detta definieras istället i 15765-4 där nätverkslagret i 15765-2 anpassats för att stödja OBD-II samtidigt som man refererar till ISO 15031-5 för applikationslagret. Jämfört med de övriga överföringsprotokollen är ISO 15765 det överlägset snabbaste, med en överföringshastighet på antingen 250 kbps eller 500 kbps och enligt nya bestämmelser skall alla OBD-II system använda sig utav ISO 15765 från och med 2008. För mer information kring ISO 15765 se Appendix C.

Utöver de fyra nämnda standarderna ovan finns det även en undantagsregel som tillåter mellanklassade fordon tillverkade 2004 och senare att använda sig utav SAE J1939 standarden. Mer information om J1939 finns under rubriken J1939.

EPA (Environmental Protection Agency) publicerade 2003 en bestämmelse där man förutom några mindre förändringar till ARB:s OBD-II standard beslutade att från 2008 ska all kommunikation för OBD-II ske med CAN-buss protokollet ISO 15765-4. Till skillnad från ARB:s krav i 1968.2 gör man inte längre någon skillnad mellan lätt- och medelklassade fordon och därmed skulle inte medelklassade fordon längre undantas från kravet att använda sig utav den diagnostiska kontakten specificerad i ISO 15031-3/SAE J1962 samt kraven att använda sig utav något av de bestämda protokollen så länge som man använder sig utav protokoll och kontakt specificerad i SAE J1939.

Fabrikat	2004	2005	2006	2007	2008
Audi/WW	-	-	-	-	100
Chrysler	5	15	35	85	100
Ford	50	85	90	100	100
GM	15	40	70	70	100
Volvo	10	95	100	100	100

Tabell 2. Procentandel för användandet av CAN-bus i nya bilar. [1]

Engine Manufacturers Association (EMA) samt motortillverkaren Cummins påpekade att denna bestämmelse innebar problem för många tillverkare av medelklassade fordon eftersom de skulle vara tvungna att byta från sitt existerande protokoll, SAE 1939. Detta protokoll har utvecklats parallellt med utvecklingen av OBD-II för att passa till tyngre dieselmotorer. EMA och Cummins kommenterade vidare att detta skulle innebära orimliga kostnader för industrin samtidigt som att det skulle innebära att ett system som skräddarsyts för att passa dieseldrivna fordon skulle förkastas för ett liknande protokoll som är utvecklat mot helt andra fordon. EPA tog till sig av denna kritik och återkallade kravet att ISO 15765-4 skulle vara det enda accepterade protokollet efter 2008. Kraven i det slutgiltiga bestämmelsen från EPA gav effekten att från 2008 kommer ISO 15765-4 var det enda godkända protokollet i fordon klassade som lättklassade (med en bruttovikt upp till 8500 pund). Samtidigt som medelklassade (med en bruttovikt mellan 8500 och 14000 pund) i fortsättningen kommer få använda sig av antingen ISO 15765-4 eller SAE J1939. Detta innebär att protokollen ISO 9141-2, ISO 14230-4 samt SAE J1850 (PWM och VPW) från och med 2008 ej längre tillåts användas. [7]

De tidigare kapitlen har beskrivit de olika delarna och funktionaliteten i OBD-II systemen. Det finns ISO och SAE specifikationer som definierar de olika lagren i ett OBD-II system enligt OSI (Open Systems Interconnection) modellen. Tabellen nedan beskriver vilket lager som specificeras av en given ISO eller SAE specifikation.

Lager enligt OSI					
Fysiska lagret	ISO 9141-2	ISO 14230-1	SAE J1850	ISO11898, ISO15765-4	SAE J1939
Datalänklagret	ISO 9141-2	ISO 14230-2	SAE J1850	ISO 11898, ISO 15765-4	SAE J1939
Nätverkslagret	-	-	-	ISO 15765-2, ISO 15765-4	SAE J1939
Transportlagret	-	-	-	-	SAE J1939
Sessionslagret	-	-	-	ISO 15765-4	-
Presentationslagret	-	-	-	-	-
Applikationslagret	ISO 15031-5	ISO 15031-5	ISO 15031-5	ISO 15031-5	SAE J1939

Tabell 3. Mappning av ISO-specifikationer mot OSI-modellen.

4.3 HD OBD

Eftersom OBD-II standarden endast ställer krav på fordon med en totalvikt på upp till 6,5 ton (lätt- och medelklassade fordon) så fanns ett intresse att utveckla ett liknande system för tyngre fordon då dessa står för en betydande del av luftföroreningarna i vårt samhälle. 2004 beslutade ARB i sin bestämmelse 1971 att samtliga motortillverkare för tyngre fordon skulle installera Engine Manufacturer Diagnostic (EMD) på sina motorer. EMD är dock långt ifrån lika omfattande som OBD-II standarden och kräver egentligen enbart att man övervakar några av de större utsläppskällorna samtidigt som man inte ställer några standardiserade krav på hur informationen ska hanteras.

I huvudsak utvecklades EMD för att försäkra att motortillverkare till tyngre fordon åtminstone implementerade en grundläggande diagnostisk funktion för kontroll av utsläppen på sina motorer. Under utvecklandet av EMD konstaterade man hos ARB att syftet var att under 2005 påbörja arbetet med en mer omfattande standardisering kring diagnostik på tyngre fordon. ARB har nu påbörjat arbete med bestämmelse 1971.1, där man kravställer tungklassade fordon. Man befinner sig vid just nu i beslutsstadiet och standarden ska efter att fått sin slutgiltiga form träda i kraft först år 2010.

Standarden baseras på SAE J1939 som används hos en övervägande majoritet av motortillverkare till tyngre fordon, detta innebär att både kontakt och protokoll skiljer sig från OBD-II standarden. Till skillnad från den tidigare EMD-bestämmelsen kräver HD OBD att man övervakar samtliga utsläppsrelaterade komponenter hos motorn, såsom bränslesystem, katalystor, partikelfilter etc. Ett fel på en sådan komponent ska indikeras innan utsläppen överstiger ett bestämt tröskelvärde. För andra komponenter ej relaterade till fordonets utsläpp ska även felindikering implementeras.

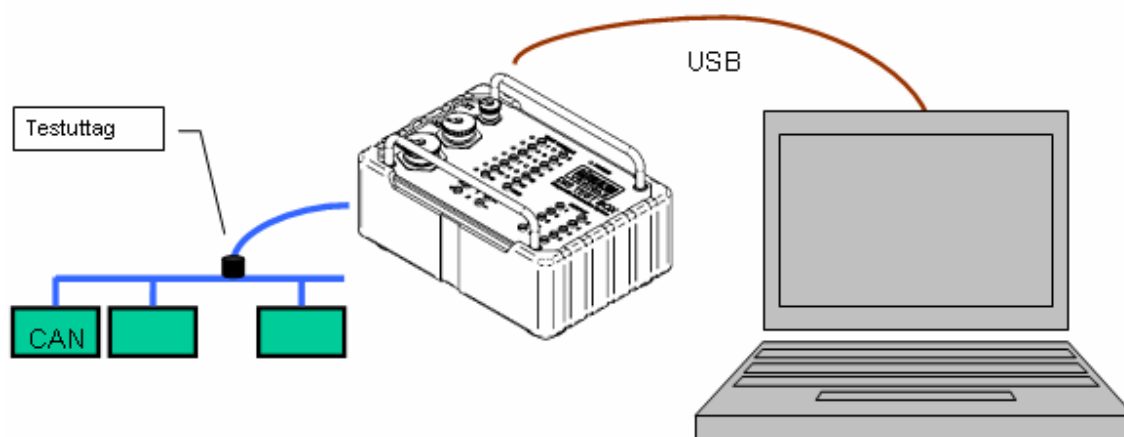
För att motortillverkarna ska hinna med att anpassa sig till den nya standarden kräver man att åtminstone en av motortillverkarnas produktfamiljer ska stödja standarden från och med 2010 fram till 2012. Under denna tid ska tillverkarnas övriga produktfamiljer fortsätta stödja EMD med ett tillägg, nämligen att kväveutsläppen ska övervakas. Från och med 2013 skall dock samtliga motorer från motortillverkarna stödja kraven för HD OBD. Denna gradvisa övergång till den nya standarden är tänkt för att underlätta omställningen för motortillverkarna, genom att under den inledande fasen få erfarenhet kring standarden innan man implementerar den hos samtliga produkter.

4.4 WWH OBD

Utvecklingen av denna standard befinner sig bara på planeringsstadiet. Det finns en mängd förslag på hur detta ska utformas men det populäraste alternativet är att man tillåter både SAE J1939 och ISO 15765-4 som underliggande protokoll men använder sig av en annan typ av anslutning än de som används i dagsläget. Troligtvis kommer Ethernet- eller WLAN-anslutning att användas. Man har även påbörjat arbetet med ett utkast till en ny ISO-specifikation där man bestämt terminologin samt ställt upp grundläggande krav för ett eventuellt system. Utvecklingen sker i samråd med branschen och flera tunga organisationer, bland annat EPA och ARB.

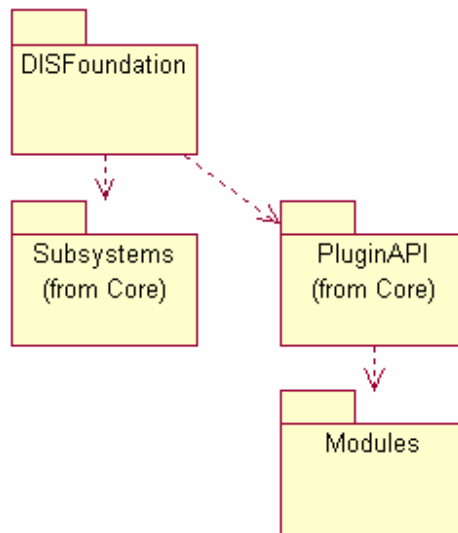
5 Befintlig lösning

Detta kapitel presenterar den befintliga lösningen som Hägglunds i dagsläget använder för diagnos av CV90. Diagnosapplikationen som Hägglunds använder kallas DIS. DIS är ett program till största delen utvecklat i C++ som körs i Windows. För att kunna läsa ut data ur ett fordon kopplar man in sig med ett hårdvaruinterface kallat DTB som omvandlar CAN-meddelanden skickade på fordonets CAN-bus till ett format som DIS kan läsa. Meddelandena som skickas på fordonets CAN-nätverk innehåller diagnostik enligt ISO 15765, I/O-signaler samt SIL-signaler. I/O-signaler används för att styra vissa funktioner hos fordonet och är som namnet antyder dubbelriktade. SIL-signaler är Hägglunds egendefinierade signalformat som enbart används till att läsa ut information från fordonen.



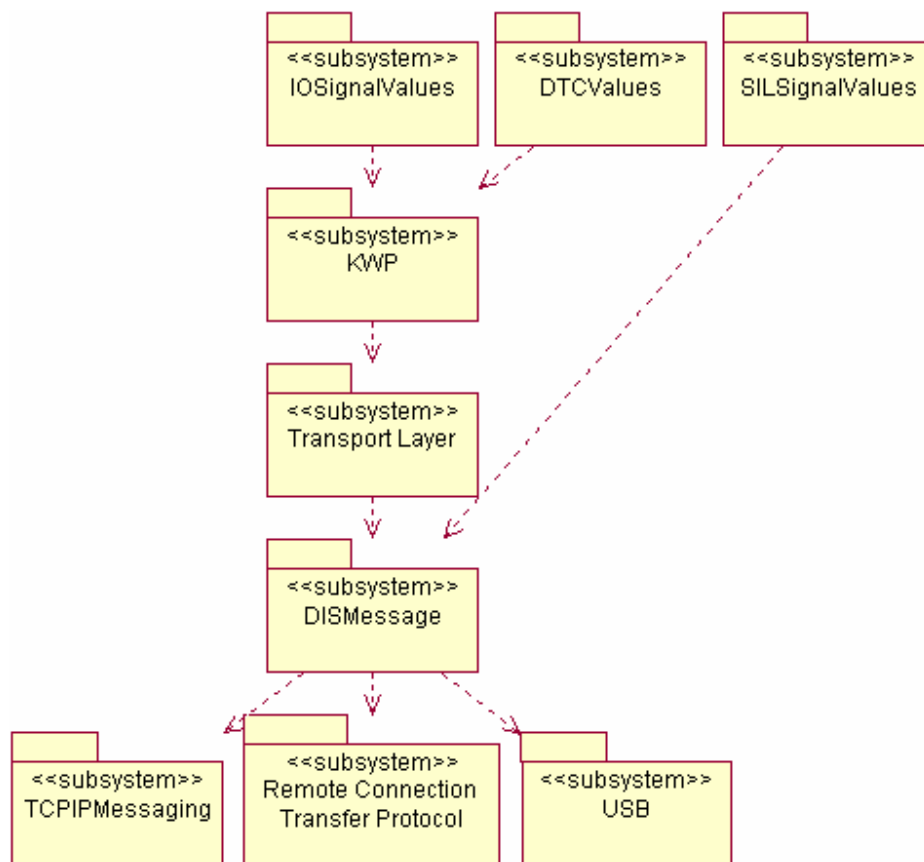
Figur 2. Schema över hur en dator via DTB. kopplas till fordonets CAN-bus

Själva applikationen är uppbyggd kring en kärna kallad DISFoundation. Till denna kärna kopplas en mängd subsystem samt moduler via ett plugin-API (Se figur 3). Subsystemen är inkluderade i applikationen och modulerna är DLL-filer som laddas in vid uppstart av programmet. Subsystemen sköter de underliggande funktionerna och modulerna sköter presentationen av informationen.



Figur 3. Schema över hur DIS är uppbyggt.

Subsystemen är uppdelade så att varje enskilt subsystem löser en viss eller flera uppgifter, dessa uppgifter kan ofta mappas mot OSI-modellen. I figur 4 presenteras en begränsad bild över hur de viktigaste subsystemen hos programmet är uppbyggda. Subsystemen längst ner kan mappas mot det fysiska lagret enligt OSI-modellen, här tolkas de mediaspecifika signalerna till format som de övre lagren kan hantera. Nästa lager i ordningen, *DISMessage*, mappas mot datalänklagret. *DISMessage* är en väldigt central del i DIS, alla relevanta meddelande kommer att passera detta subsystem på sin väg upp i systemet. Lagret ovan, *Transport*, kan liknas vid OSI-modellens transportlager, här sköts uppkopplingar mellan olika noder samt funktionalitet för att garantera att meddelanden når sin destination. Nästa lager, *KWP* kan liknas vid sessionslagret, här sköts specifika operationer för meddelanden som sänds på *KWP*-format, så att de övre lagren ska kunna hantera denna information. Det översta lagret, kan ses som en kombination av presentationslagret och applikationslagret, här översätts informationen till ett format som gör den enkel att presenteras i själva applikationen.



Figur 4. Schema över de viktigaste subsystemen i DIS.

När meddelanden från CAN-bussen når DIS via DTB:n hanteras dessa olika beroende på vilken typ av information de innehåller. Meddelanden som innehåller vanliga signaler (såsom varvtal och däcktryck) skickas i regel till ett subsystem kallat Signal Information, medan meddelanden som innehåller diagnostisk information, såsom felkoder och freezeframes, skickas till ett subsystem kallat Diagnostics. DIS är tack vare sin modulära uppbyggnad väldigt lätt att utöka med nya funktioner. Om man önskar att utöka systemets funktionalitet behöver man bara lägga till ett nytt subsystem samt en modul för att presentera den nya informationen.

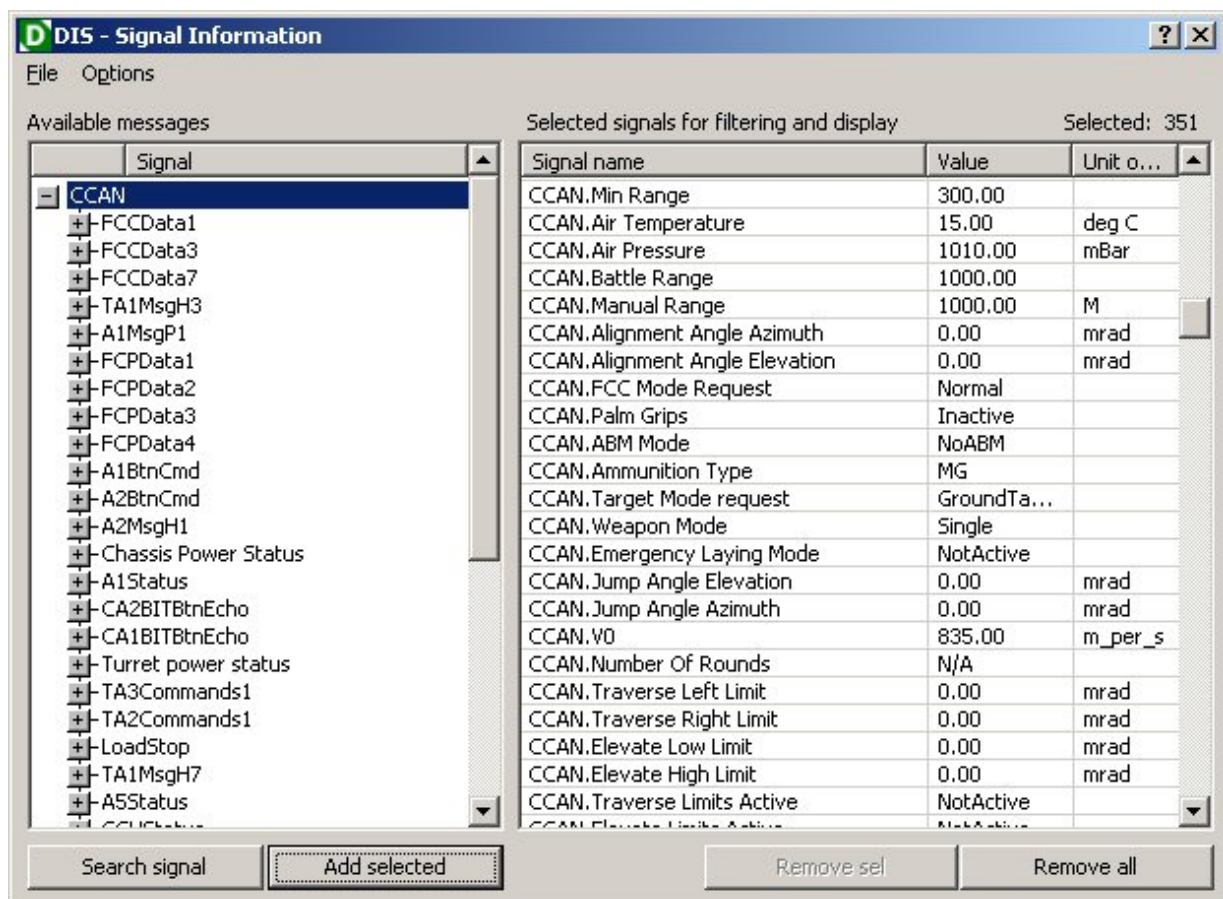
Motiveringen till att använda DIS är att signaler ej kan utläsas med hjälp av vanliga multimetrar eller oscilloskop. Eftersom datatrafiken är multiplexerad skulle resultatet se ut som elektriskt brus och därför behövs en programvara som tolkar informationen och presenterar den digitalt. Det finns visserligen tredjepartslösningar som erbjuder liknande funktionalitet men där presenteras inte själva informationen eftersom dess definition till största delen är Hägglunds-specifik

5.1 Nuvarande funktionalitet

I dagsläget har DIS utvecklats mot CV90, och man har byggt in en hög grad av funktionalitet i programmet. Några av de funktioner som just nu finns i DIS är:

Presentera signalinformation

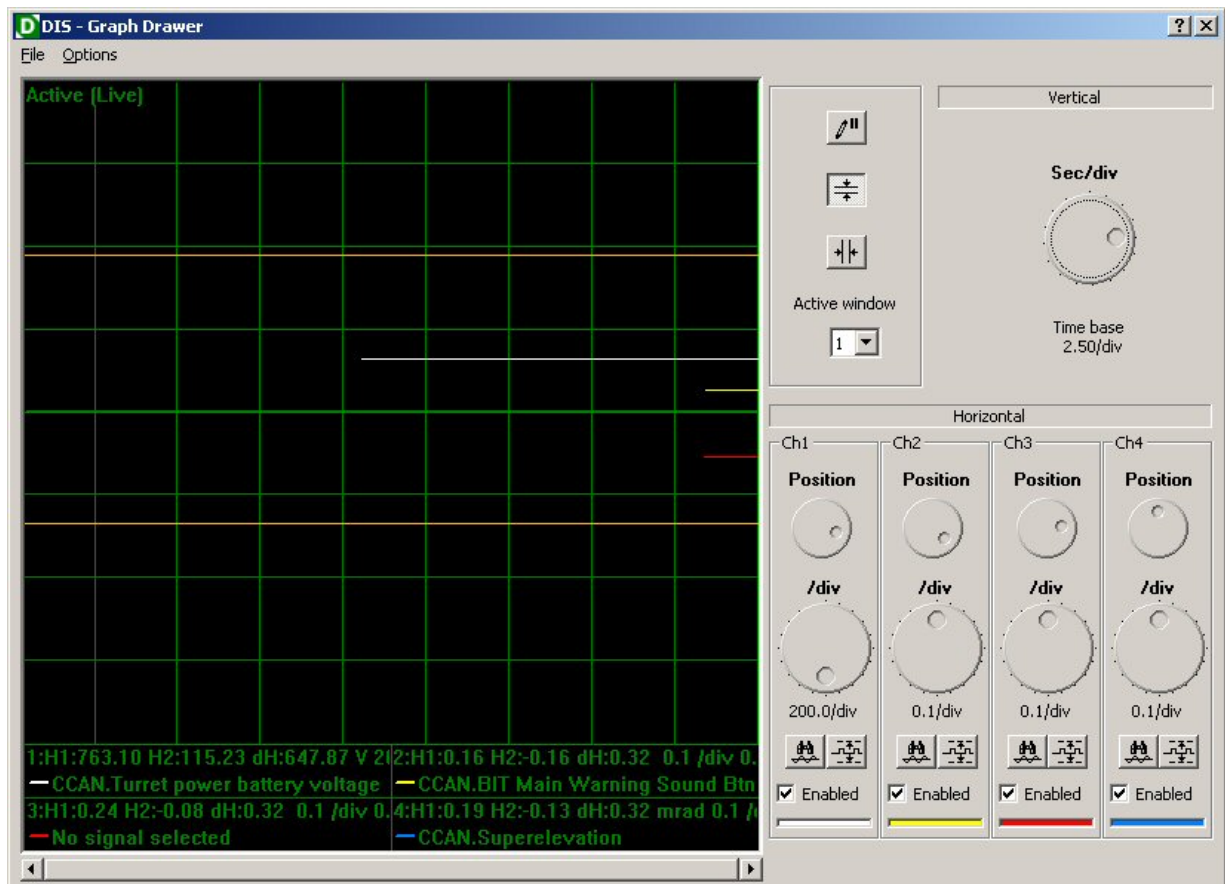
Hantering av signaler är en viktig funktion i DIS och för att presentera de signaler som sänds på fordonets CAN-bus finns en modul kallad Signal Information. Då det i regel finns väldigt många signaler på bussen fungerar Signal Information så att användaren först väljer ut vilka signaler som anses intresseranta bland de signaler som stöds i programmet. När dessa är valda presenteras deras information i den hastighet de uppdateras. Det finns även möjlighet att söka bland de tillgängliga signalerna. För att välja vilka signaler som ska stödjas använder man ett verktyg kallat fordonsdatabasen, se kapitel 5.2 för mer information.



Figur 5. Signal Information-modulen

Grafitare

Användaren kan välja en eller flera signaler och presentera som en graf, liknande ett oscilloskop. Upp till fyra olika signaler kan väljas och sedan ställas in så de presenteras mot varandra.



Figur 6. Modul för att simulera oscilloskop.

Analogmätare

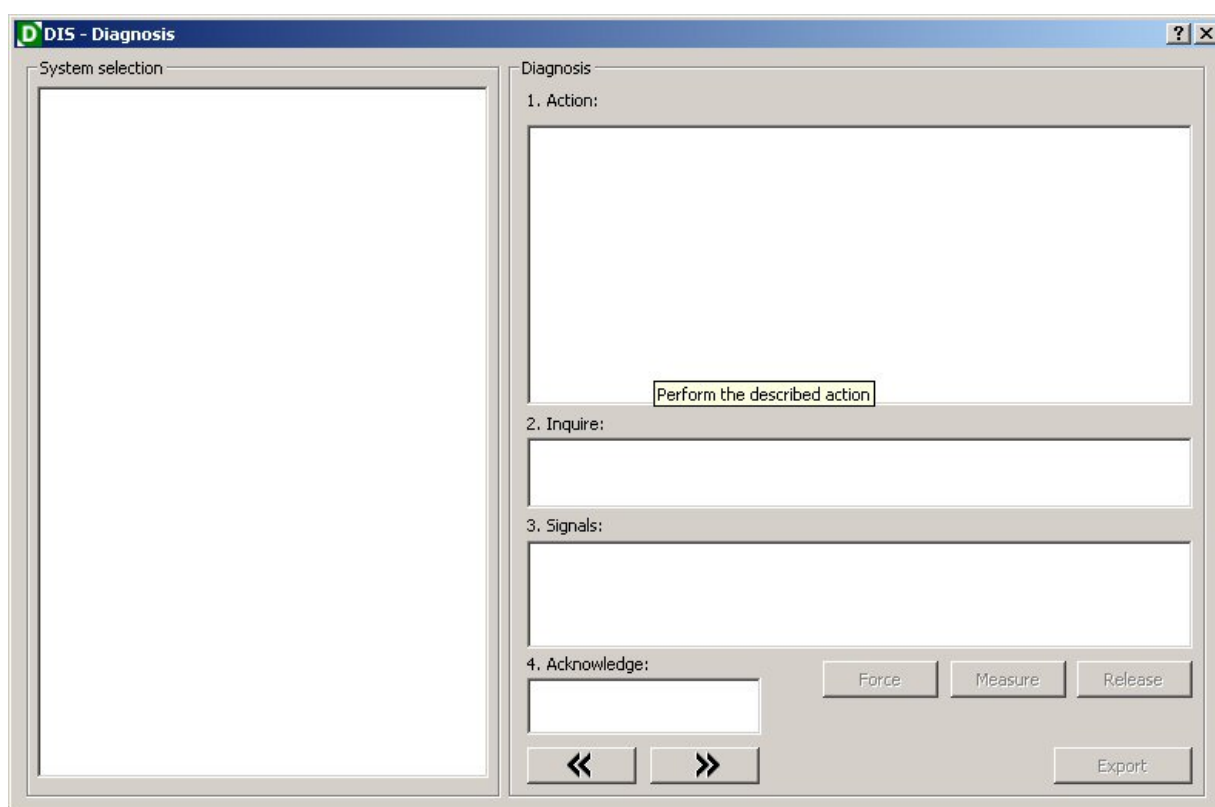
Används för att mäta analoga signaler på fordonet eller på testboxens analoga ingångar. Kan liknas vid en multimeter.

Exportera information

Erbjuder möjlighet att spela in information på ett format som lätt kan importeras i till exempel Excel. Både signaler på CAN-bussen och loggar kan spelas in. Vid inspelning av signaler fungerar det så att användaren först väljer signalen i Signal Information och sedan erbjuder Datarecorder funktionalitet för att spela in dessa. När inspelningen avbryts kan användaren välja att spara denna till en fil.

Diagnostik

Här väljer användaren vilken diagnostisk funktion denne vill utföra, såsom att läsa ut felkoder eller att nollställa dessa.



Figur x: Modul för diagnostik av stridsfordon

Ladda ner ny programvara

Möjlighet att uppgradera programvaran i fordonet via DIS. Både programvara i mikrokontrollers och inbyggda system kan uppgraderas via den här funktionen.

Fjärranslutning

DIS kan kopplas till en fjärransluten dator där användaren får se exakt samma information som den lokalt anslutne.

Designers toolkit

Den här funktionen är dold för vanliga användare och används till att assistera utvecklare av programmet.

Hämta loggar

Hämtar sparade loggar från fordonet. Dessa kan även nollställas med denna modul.

Ytterligare moduler

Utöver de ovan nämnda modulerna går det även att synkronisera klockan hos fordonet, sätta alarm mot specifika signaler, köra självttest av DIS-hårdvaran, testa kablar samt välja hur avancerat användargränssnitt som ska användas.

5.2 Fordonsdatabas

För att underlätta hanteringen signaler har Hägglunds utvecklat en informationsdatabas kallad *fordonsdatabasen* (FDB). Fordonsdatabasen är enkelt förklarad en databas knuten till ett webbgränssnitt som är tillgängligt på Hägglunds intranät för de med behörighet. Databasen är anpassad för att innehålla de mängder av olika signaler och meddelanden som bland annat DIS hanterar. Hantering av dessa sker via webbaserade formulär där möjlighet att fylla i nya signaler och meddelanden erbjuds. Databasen erbjuder vidare möjligheten att hämta ut den information som önskas till en xml-fil på ett format som DIS kan läsa. I dagsläget har fordonsdatabasen ett begränsat stöd för meddelanden och signaler av J1939-typ, även om den ej är helt anpassad till dessa.

FRAME-J1939 (ID = 13211)

Uppdaterad: 2006-03-30 10:55:05 av Viberg Mattias

Språk: ENGLISH Spara Ta bort Skapa kopia Stäng

BESKRIVNING

A CAN frame containing J1939 structured data

* obligatorisk 🚫 översättningsbar

ATTRIBUT

Namn	Värde	Beskrivning	Ändra
Name *	Fan Drive		✖
SoftwareName *	FD		✖
Description * 🚫	Fan Drive		✖
ID *	0x18FEBD00		✖
CanFrame_IsExtended *	<input checked="" type="checkbox"/>		✖
J1939Frame_Period *	1000		✖

	CanFrame_Bus	Vilken bus framen går på
	CanFrame_Recievers	Vilka noder som tar emot denna frame
	CANFrame_Transmitter	Vem sänder på den bussen
	Document	Vilka dokument som påverkas av denna instans

GRUPPER

Namn	Värde	Beskrivning
FRAME-CAN_LOCATION		Vilken CAN buss datapaketet sänds på och vem som sänder på den CAN bus

Spara Ta bort Stäng

Figur 7. Fordonsdatabasens gränssnitt.

6 Implementation

Detta kapitel presenterar den lösning som examensarbetet resulterade i. Här presenteras de krav som ställs på applikationen, den design som valts och motiveringen till denna. Även en genomgående beskrivning av applikationens uppbyggnad ges.

6.1 Krav

Nedan presenteras de krav som framkommit under fördjupningsstudien samt de krav som ställdes på applikationen från Hägglunds sida.

Kravet från Hägglunds var att applikationen ska erbjuda diagnostik av bandvagnens motor och växellåda. Resultatet ska vara en applikation som kan köras under Windows XP. Följande information och funktionalitet ska finnas i användargränssnittet:

- Läs och presentera realtidsdata
- Läs och presentera freezeframes från motor
- Läs och presentera aktiva och tidigare aktiva felkoder
- Möjlighet att radera lagrad data som t.ex. freezeframes och felkoder

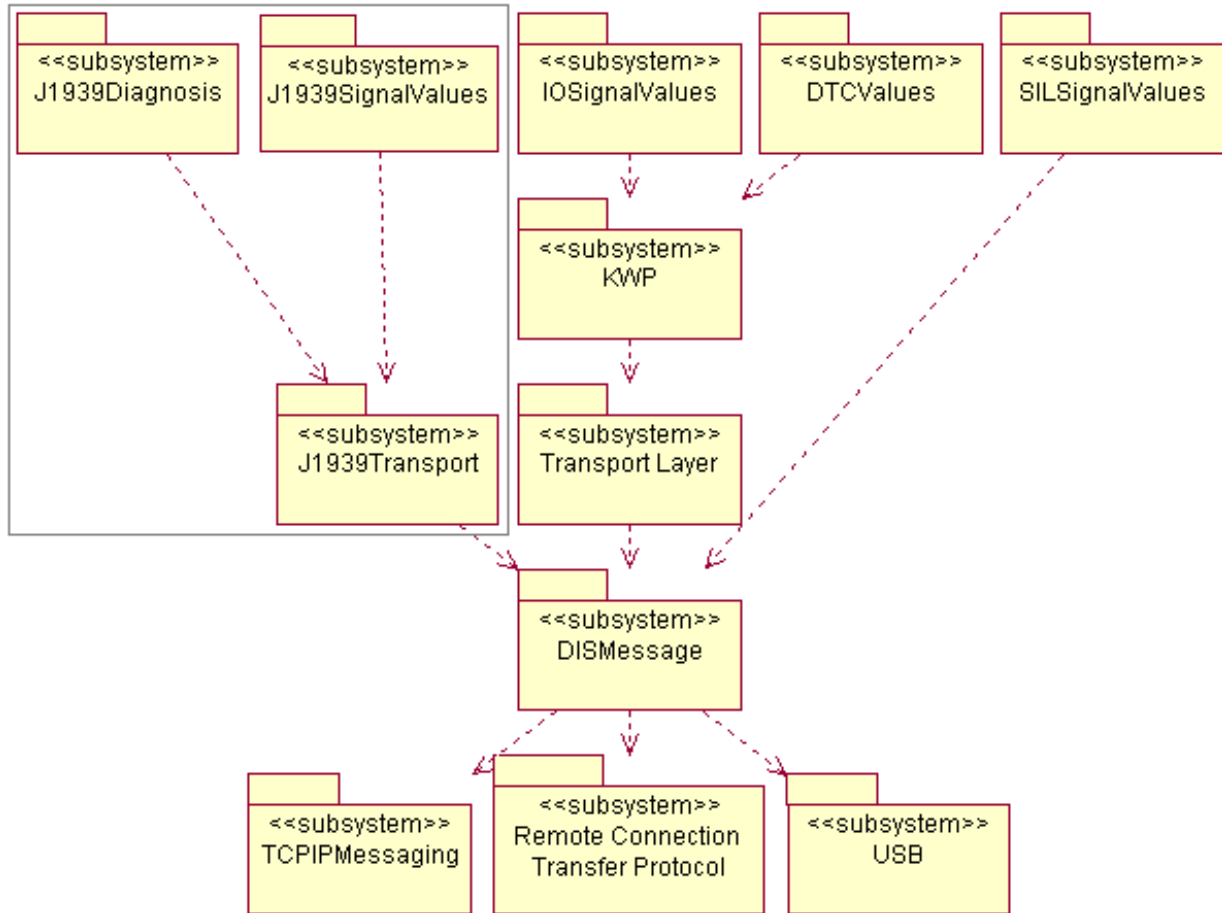
Efter analysen av den befintliga diagnosprogramvaran (DIS), framgick det att möjligheten att bygga in stöd för diagnos mot bandvagn i DIS existerade. Anledningen till detta är att DTB:n (se kap 5) använder sig utav CAN-meddelanden för att skicka information till och ta emot information från bandvagnen. I vår lösning kan vi använda oss av DTB:n eftersom information på J1939 format också skickas i CAN-meddelanden, men med viss modifikation från den nuvarande lösningen.

6.2 Design

Nedan presenteras designen för applikationen samt hur den har integrerats i DIS programvaran.

I och med den modulära uppbyggnaden hos DIS var det från början självklart att vår lösning skulle implementeras i form av subsystem som skulle sköta J1939-kommunikationen samt erbjuda tjänster så att inläst information lätt skulle kunna avläsas av en annan modul. I designen utgick vi från tre

huvudfunktioner som implementerades i varsitt motsvarande subsystem. Det första av dessa var ett transportlager (enligt OSI-modellen samt specifikationen i J1939/21), detta subsystem skulle sköta all logisk hantering av meddelanden. Vidare så designades ett subsystem för att hantera utläst realtidsdata samt ett för att hantera diagnostiken, vilket var själva kärnan i examensarbetet. Figur 7 visar en översiktlig bild över vår design, vår del markerad med en grå rektangel.



Figur 8. Utökning av DIS applikationen.

6.3 Systembeskrivning

Nedan kommer de subsystem som använts för att implementera diagnostik mot bandvagn i DIS-applikationen att presenteras. De subsystem som ingår i implementationen är följande: J1939Transport som motsvarar transportlagret enligt OSI-modellen, J1939Diagnosis och J1939Signalvalues som mappas mot både presentationslagret och applikationslagret i modellen. Först ges en presentation av det aktuella subsystemet och funktionaliteten det erbjuder därefter följer en mer detaljerad beskrivning av subsystemets olika delar.

6.3.1 J1939Transport

J1939Transport-subsystemet är motsvarigheten till Transport subsystemet i DIS-applikationen. Eftersom DIS använder sig av ISO 15765-protokollet (Se appendix C) för kommunikation krävdes implementation av ett nytt transportlager som erbjöd stöd för kommunikation enligt J1939-protokollet.

J1939Transport-subsystemets huvuduppgift är att ta emot och skicka J1939-meddelanden som används för kommunikation. Subsystemet ansvarar även för att sätta ihop fragmenterade meddelanden (se appendix B) samt återsändningar av meddelanden som inte levererats till motorns eller växellådans ECU (Electronic Control Unit).

Transportlagret låter de överliggande subsystem som förväntar sig J1939-meddelanden med diagnosinformation att allokera en egen buffert hos transportlagret. I denna implementation är det subsystemen J1939Diagnosis och J1939SignalValues som allokerar buffrar i transportlagret. I buffrarna placeras de J1939-meddelanden som ska levereras till överliggande subsystem.

Vid varje exekvering av transport-subsystemet (inträffar var tionde millisekund), hämtas nyinkomna meddelanden upp från det underliggande subsystemet DISMessage och placeras i de två allokerade buffrarna. De överliggande systemen hämtar sedan upp dessa meddelanden för att få tillgång till diagnostisk information.

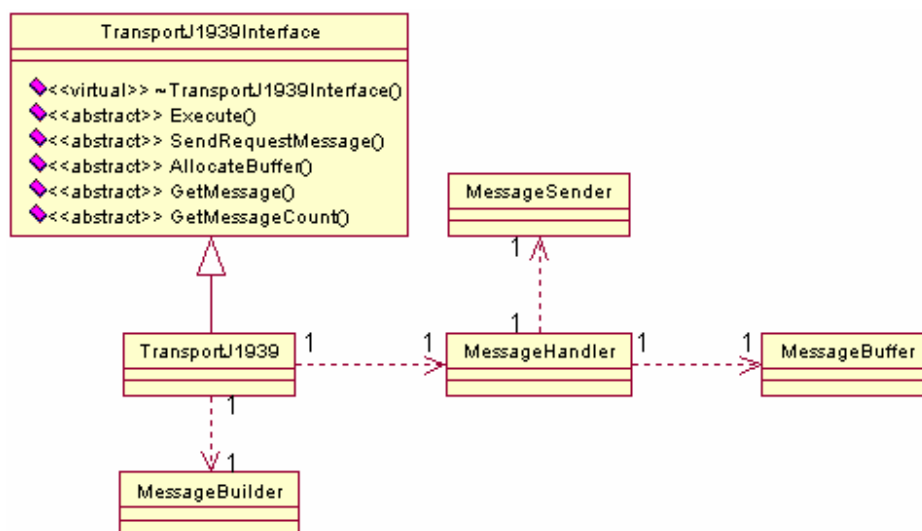
Transportlagret hanterar även sändningen av request-meddelanden. Om ett anrop från ett överliggande lager sker för att skicka ett request-meddelande (begära diagnosinformation), bygger

transport lagret ett J1939-meddelande och lägger in det i sändbufferten Vid varje exekvering töms denna buffert och varje meddelande som lagrats i bufferten skickas till DISMessage subsystemet för att där byggas ihop till ett CAN-meddelande och sedan skickas till fordonet via CAN-bussen.

Funktionaliteten för att garantera att J1939-meddelanden kommer fram till mottagaren för meddelandet ligger även i J1939Transport-subsystemet. Detta hanteras genom att när ett meddelande skickats väntar subsystemet på ett svar i form av ett ACK-meddelande. Ifall svar inte inkommit efter ett visst antal exekveringar återsänds meddelandet.

Nedan kommer de klasser som J1939Transport subsystemet består av, den funktionalitet dessa erbjuder samt hur de kommunicerar med varandra presenteras. De klasser som tillsammans bildar J1939Transport subsystemet är:

- J1939TransportInterface
- J1939Transport
- MessageHandler
- MessageBuilder
- MessageSender
- MessageBuffer



Figur 9. Klassdiagram över J1939Transport subsystemet.

J1939TransportInterface

Detta är det interface som de övriga subsystemen har tillgång till. Här finns metoder som tillåter andra subsystem att ta del av transportlagrets funktioner. Funktionalitet som J1939TransportInterface erbjuder är följande:

- Allokera en buffert i transportlagret
- Kontrollera om det finns meddelanden tillgängliga för leverans till överliggande subsystem
- Hämta meddelanden från den allokerade bufferten
- Sända meddelanden till fordonet

J1939Transport

J1939Transport är den centrala klassen i transportlagret. Klassen hanterar anrop från överliggande subsystem och skickar dessa anrop vidare till den klass i subsystemet som ansvarar för att hantera anropet. Tillåter de överliggande subsystemen att skicka meddelanden, allokera buffrar i J1939transport-subsystemet samt att hämta meddelanden ur dessa buffrar.

MessageBuilder

Ansvarar för att bygga ihop J1939-meddelanden som ska sändas till fordonet. När en begäran sker från ett subsystem med syfte att erhålla en viss typ av diagnosinformation bygger MessageBuilder-klassen detta meddelande. MessageBuilder tar in ett PGN (Parameter Group Number) och en adress dit request-meddelandet ska skickas och bygger sedan ett J1939-meddelande. Meddelandet består av en header och ett datafält, där headern beskriver vem som skickat meddelandet, vem mottagaren är samt vilket prioritet meddelandet har. Datafältet innehåller PGN-numret för att specificera vilken typ av information som begärs från mottagaren. Efter meddelandet byggts skickas det tillbaka till J1939Transport klassen för att skickas vidare till MessageSenderns sändbuffert. För ytterligare information hur headern och datafältet i meddelandet är uppbyggt se Appendix B.

MessageHandler

MessageHandler-klassen ansvarar för hantering av J1939-Meddelanden. Främsta uppgifterna är mottagning av singleframes samt hopsättning av fragmenterade meddelanden. MessageHandlern har en buffert allokerad i DISMessageClass, där alla CAN-meddelanden som kommer från fordonets

CAN-buss placeras. Vid varje execute kontrollerar MessageHandlern om det finns meddelanden tillgängliga i bufferten. Ifall CAN-meddelanden existerar hämtar MessageHandlern upp dessa för analysering och hantering. Meddelandena som levereras till MessageHandler kan vara av olika typer och ska hanteras olika beroende på typ. Typerna av meddelanden är:

- *Single CAN Frame*, all data sänds i ett CAN-meddelande
- *Broadcast Announce Message*, (BAM), indikerar att en överföring av fragmenterade meddelanden ska starta. Innehåller information om totala längden av alla meddelanden samt hur många meddelanden som ingår i överföringen.
- *Data Transfer*, är en del av en överföring av fragmenterade meddelanden. Första byten är sekvensnumret för meddelanden och efterföljande sju bytes innehåller data.
- *Ready To Send*, (RTS), en förfrågan ifall sändaren får påbörja en sändning av en stor datamängd. Besvaras med en CTS (Clear To Send) ifall mottagaren tillåter detta.
- *Acknowledgement*, (ACK), ett positivt svar på en förfrågan.
- *Negative Acknowledgement*, (NACK), ett negativt svar på en förfrågan.

Nedan beskrivs hur de olika typerna av meddelanden hanteras av MessageHandler.

Single CAN Frame

Ifall informationen som begärts skickas i ett enda CAN-meddelande sänds det som en Single CAN Frame. MessageHandlern behöver inte utföra någon speciell hantering av dessa meddelanden utan gör om detta till ett J1939-meddelande och lägger in det i varje mottagnings buffert som är allokerad av överliggande subsystem.

BAM (Broadcast Announce Message)

CAN-meddelanden erbjuder endast möjlighet att skicka åtta bytes data i ett meddelande (se Appendix A). När större datamängder än åtta bytes ska skickas så krävs det att informationen fragmenteras. Ett BAM-meddelande indikerar starten på en överföring av fragmenterad data.

När MessageHandlern mottar ett sådant meddelande läser den ut informationen i BAM-meddelandet (avsändare, total längd och antal meddelanden som kommer skickas) och sätter upp parametrar för att kunna ta emot och sätta ihop den fragmenterade informationen.

MessageHandlern behöver endast hantera ett BAM åt gången som skickas från en och samma sändare. Ifall ett nytt BAM-meddelande tas emot från samma sändare under pågående mottagning ignoreras helt enkelt detta BAM-meddelande. Detta fall är något som inte ska kunna ske enligt J1939 standarden. Däremot så klarar MessageHandlern av att hantera BAM meddelanden som skickats från olika sändare genom att kontrollera vem som är avsändare med hjälp av meddelandets unika identifierare.

Data Transfer

Ett datatransfermeddelande är en del i en överföring av fragmenterad information. Data Transfer meddelandet innehåller ett sekvensnummer på en byte och efterföljande sju bytes är data. När MessageHandlern mottar ett datatransfermeddelande kontrolleras först att ett BAM eller ett RTS mottagits från sändaren av meddelandet. Ifall ett datatransfermeddelande mottas utan att först föregås av ett BAM eller RTS från sändaren så ignoreras meddelandet och kastas bort.

Om ett BAM eller ett RTS mottagits från sändaren innan överföringen av data transfer meddelanden påbörjas så läser MessageHandlern ut informationen i det inkommande data transfer meddelandet och kontrollerar att sekvensnumret är det väntade. Ifall sekvensnumret inte överensstämmer med det förväntade sekvensnumret så ignoreras meddelandet och kastas bort.

Om meddelandet innehåller det förväntade sekvensnumret läses informationen ur meddelandet ut och läggs till den tidigare mottagna informationen. När det sista datatransfermeddelandet mottagits byggs ett meddelande innehållande all den fragmenterade informationen. Detta meddelande läggs sedan i varje mottagningsbuffert som de överliggande subsystemen allokerat. Överliggande subsystem kan därefter hämta det färdiga meddelandet innehållandes all information som ingick i överföringen.

RTS (Request To Send)

RTS meddelanden är väldigt lika BAM meddelanden, båda typerna indikerar att en överföring av en datamängd större än åtta bytes ska överföras. Skillnaden mellan dessa två är att en enhet som skickar ett RTS-meddelande förväntar sig ett CTS-meddelande innan överföringen påbörjas. Ett RTS-meddelande innehåller adressen för enheten som vill skicka data samt hur mycket data den vill skicka. När MessageHandlern mottar ett RTS-meddelande, så byggs ett CTS-meddelande ihop. Detta meddelande läggs sedan i sänd bufferten för att skickas till enheten som svar på RTS-meddelandet.

CTS (Clear To Send)

CTS är ett svar på ett RTS meddelande. Detta meddelande specificerar hur mycket information en mottagare kan ta emot av den totala mängden som specificeras i RTS-meddelandet. MessageHandlern bygger ett CTS som svar när ett RTS inkommer. Mängden data som kan tas emot av MessageHandlern sätts alltid till hela datamängden som angavs i RTS meddelandet. Anledningen till detta är att MessageHandlern hos applikationen inte har några problem med resurser när det gäller att ta emot denna data.

ACK (Positive Acknowledgement)

Detta meddelande indikerar ett positivt svar på en förfrågan. ACK-meddelanden skickas endast när en enhet skickar en förfrågan specifikt till en annan enhet och inte ifall sändaren broadcastar sin förfrågan. Används av MessageHandlern för att kontrollera ifall ett svar inkommit på ett request-meddelande. Ifall ett svar inte kommer in på ett request-meddelande så återsänds meddelandet efter en timeout har gått.

NACK (Negative Acknowledgement)

Detta meddelande indikerar ett negativt svar på en förfrågan. Skickas ifall en förfrågan sker specifikt till en annan enhet och denna enhet inte stödjer informationen som begärs. NACK-meddelanden skickas aldrig om sändaren broadcastar sin förfrågan.

MessageSender

MessageSender är den klass som ansvarar för sändning av meddelanden. MessageSendern använder sig av DISMessageClassInterface för att få tillgång till det underliggande subsystemets funktionalitet att skicka meddelanden. Klassen innehåller en buffert där de meddelanden som byggs ihop av

MessageBuilder-klassen lagras. Vid varje exekvering av J1939Transport-subsystemet kontrolleras ifall det finns meddelanden i bufferten att sända. Om det existerar meddelanden i bufferten skickar MessageSendern alla dessa meddelanden till DISMessageClass för vidare sändning ut på CAN-bussen.

MessageBuffer

MessageBuffer-klassen representerar bufferten som används för att lagra inkommande J1939-meddelanden som ska levereras till överliggande subsystem. Varje subsystem som allokerar en MessageBuffer hos J1939Transport subsystemet erhåller ett unikt id för att kunna nå sin specifika buffert och hämta meddelanden från denna.

6.3.2 J1939SignalValues

Eftersom Hägglunds egenutvecklade signaler, SIL-signaler, följer specifikationen för signaler över CAN, kunde deras signalhantering till stor del återanvändas för att skapa ett subsystem för att hantera signaler över J1939. Den största skillnaden i hanteringen av de olika signaltyperna var att SIL-signaler har en statisk längd som aldrig överstiger åtta bytes, medan J1939-signaler kan ha signaler av variabel längd upp till 1785 bytes. Vidare skickas SIL-signaler alltid automatiskt medan vissa J1939-signaler enbart skickas när de begärs från någon nod i systemet.

Subsystemet i DIS som sköter hantering av SIL-signaler, SILSignalValues läser vid uppstart in en xml-fil där alla signaler som hanteras finns införda. Denna xml-fil kan genereras med hjälp av den tidigare nämnda fordonsdatabasen. När sedan signaler strömmar in från CAN-bussen jämförs deras identifierare med de som lästs in från xml-filen. Ifall signalen hittas i filen (vilket betyder att den hanteras) tolkas signalens värden med hjälp av den information som lästs in från xml-filen.

I och med att vi var tvungna att hantera signaler av variabel längd samt signaler med en storlek större än åtta bytes behövdes den befintliga signalhanteringen modifieras. Olika sätt att anpassa den information som hämtades ut ur fordonsdatabasen undersöktes men vi fann till slut att den smidigaste lösningen var att utesluta meddelanden av variabel längd från signalhanteringen och istället sköta dessa separat. Motiveringen till detta vara att bandvagnen vi arbetat mot endast behövde stöd för tre meddelanden med variabel längd.

Hantering av signaler med fix längd men med en storlek större än åtta bytes samt hantering av signaler som måste begäras kunde dock implementeras med några mindre förändringar av den befintliga signalhanteringen.

Eftersom signalerna kan ha en storlek större än åtta bytes måste de kunna delas upp i flera paket. Eftersom isärplockning och hopsättning av dessa ska ske hos transportlagret krävdes en viss strukturell förändring för att hantera dessa signaler. I den befintliga lösningen är signalhanteringen kopplad direkt mot länklaget enligt OSI-modellen (SILSignalValues är kopplat mot DISMessage, se figur 4). Vi var tvungna att koppla in transportlagret mellan dessa två för att hantera samtliga signaler på ett korrekt sätt (se figur 7).

Systemet garanterar att begärda signaler besvaras genom att en begäran skickas igen ifall svaret ej inkommer inom ett visst tidsintervall.

Eftersom vi återanvänt stora delar av den befintliga signalhanteringen har det inneburit att vår lösning är kompatibel med några av de mer avancerade signalhanteringsalternativ som finns i DIS, grafitning samt datainspelning och export, dock med vissa begränsningar (Se kapitlet lösningens begränsningar för mer information kring detta).

6.3.3 J1939Diagnosis

Subsystemet J1939Diagnosis är vår motsvarighet till subsystemet DTCValues i DIS applikationen. J1939Diagnosis kan mappas mot applikations och presentationslagren enligt OSI-modellen. J1939Diagnosis huvuduppgifter är att översätta J1939-meddelanden till diagnosinformation som sedan kan levereras till användargränssnittet för presentation. Subsystemet ansvarar även för att ta emot requests från användargränssnittet och skicka dessa vidare till J1939Transport systemet.

J1939Diagnosis har en egen buffert allokerad i J1939Transport där meddelanden på J1939 format lagras. Vid varje exekvering kontrolleras ifall det finns meddelanden att hämta ifrån bufferten. Om meddelanden finns att hämta levereras dessa från transporlagret upp till J1939Diagnosis. Subsystemet analyserar sedan dessa meddelanden och beroende på typen av meddelande översätts innehållet till motsvarande diagnosinformation. De typer av diagnosinformation som hanteras av J1939Diagnosis är:

- Freezeframe information
- Aktiva felkoder
- Tidigare aktiva felkoder
- Identifikation av fordonet
- Identifikation av mjukvara i fordonet
- Identifikation av komponenter i fordonet
- Information om diagnosstatus för fordonet.

Dessa typer av information översätts sedan till klasser (se figur 9). Överliggande lager kan sedan använda sig av diagnosinterfacet för att få tillgång till diagnosinformation som är översatt från J1939-meddelanden till klasser.

De klasser som ingår i J1939Diagnosis subsystemet är följande:

- J1939DiagnosisInterface
- J1939Diagnosis

- J1939freezeFrame
- J1939ActiveDTC
- J1939PreviouslyActiveDTC
- J1939TroubleCode
- J1939VehicleIdentification
- J1939ComponentIdentification
- J1939SoftwareIdentification
- J1939DiagnosticReadiness
- J1939SignalConverter

J1939DiagnosisInterface

Detta är det interface som specificerar vilka metoder som finns tillgängliga i J1939Diagnosis för överliggande subsystem. De viktigaste funktionerna är de som tillåter ett subsystem att hämta de olika typerna av diagnosinformation samt att skicka request-meddelanden.

J1939Diagnosis

J1939Diagnosis är den centrala klassen i J1939Diagnosis subsystemet. Huvuduppgifterna är att lyssna på anrop ifrån överliggande subsystem (i denna implementation, användargränssnittet) samt att hämta J1939-meddelanden från J1939Transport-subsystemet och översätta dessa till motsvarande diagnosinformation. Subsystemet har en buffert allokerad i J1939Transport-subsystemet där inkommande meddelanden på J1939 format lagras och sedan kan hämtas upp.

När ett anrop från ett överliggande subsystem sker för att begära diagnos information skickar J1939Diagnosis detta anrop vidare till J1939Transport systemet som bygger ihop request-meddelandet och sänder det till fordonet via CAN-bussen. När ett request-meddelande har sänts till transportlagret sätter J1939Diagnosis klassen upp parametrar för att indikera för det överliggande subsystemet när ett svar på förfrågan har inkommit och finns tillgängligt för leverans.

Vid varje exekvering av J1939Diagnosis-subsystemet kontrolleras ifall det finns meddelanden i bufferten hos transportlagret. Ifall meddelanden finns tillgängliga hämtas dessa upp till

J1939Diagnosis och översätts till diagnosinformation som lagras i klasser som motsvarar typen av information. Efter att meddelandet översatts så sätts parametrar för att indikera för det överliggande subsystemet att informationen som begärdes nu finns tillgänglig. Inget direkt anrop sker för att indikera för överliggande lagren att information inkommit och översatts. De subsystem som är intresserade av diagnosinformation ställer frågor med jämna intervall och kontrollerar ifall informationen inkommit och översatts.

Ifall informationen inkommit kan överliggande subsystem använda de olika metoder som finns tillgängliga i J1939Diagnosis för att läsa ut informationen i form av strängar som sedan presenteras i användargränssnittet.

J1939SignalConverter

Denna klass ansvarar för att översätta signaler till värden som ska presenteras i användargränssnittet. Olika signaler använder olika formler för översättning till värden som representerar t.ex. temperatur, hastighet eller ett varvtal.

J1939SignalConverter hanterar även översättningen av felmeddelanden för signaler som ligger utanför deras giltiga gränsvärden. En signal på en t.ex. en byte kan ta värden mellan 0-255, men signalen kanske bara är giltig för värden mellan 0-250. De övriga fem värdena som signalen kan anta specificerar då olika typer av felmeddelanden. J1939SignalConverter kontrollerar ifall signalen ligger inom sina gränsvärden och ifall värdet är utanför översätts signalen till ett felmeddelande.

J1939DiagnosticReadiness

Lagrar all information rörande fordonets diagnostiska status. Här presenteras hur många aktiva och tidigare aktiva felkoder som finns lagrade i fordonet. Information om vilken kompatibilitet fordonet har mot OBD-II standarden går även att utläsa härifrån. Slutligen lagras också statusen från fordonets självtester och vilka olika självtestsystem som är implementerade och stöds.

J1939freezeFrame

Denna klass motsvarar en freezeFrame. Innehåller all information som ingår i en freezeFrame som t.ex. vilken felkod freezeFramen gäller, hur många gånger felet inträffat, varför felet inträffat samt även en liten mängd realtidsdata som avlästes i precis det ögonblick felet inträffade. Erbjuder

metoder för att översätta denna information till strängar som överliggande system (användargränssnittet) kan presentera.

J1939ActiveDTC

Lagrar information om de aktiva felkoderna som finns i fordonet. Här lagras statusen på felindikatorlamporna samt även en lista över de felkoder som existerar (J1939TroubleCode).

J1939PreviouslyActiveDTC

Lagrar information om de tidigare aktiva felkoderna som finns i fordonet. Förutom statusen på indikatorlamporna lagras även de tidigare aktiva felkoderna (J1939TroubleCode) i en lista

J1939TroubleCode

Denna klass motsvarar felkoden som finns lagrad i fordonet. Innehåller felkoden, hur många gånger felet upptäcks samt vad som orsakat felet. Klassen används av J1939ActiveDTC och J1939PreviouslyActiveDTC.

J1939VehicleIdentification

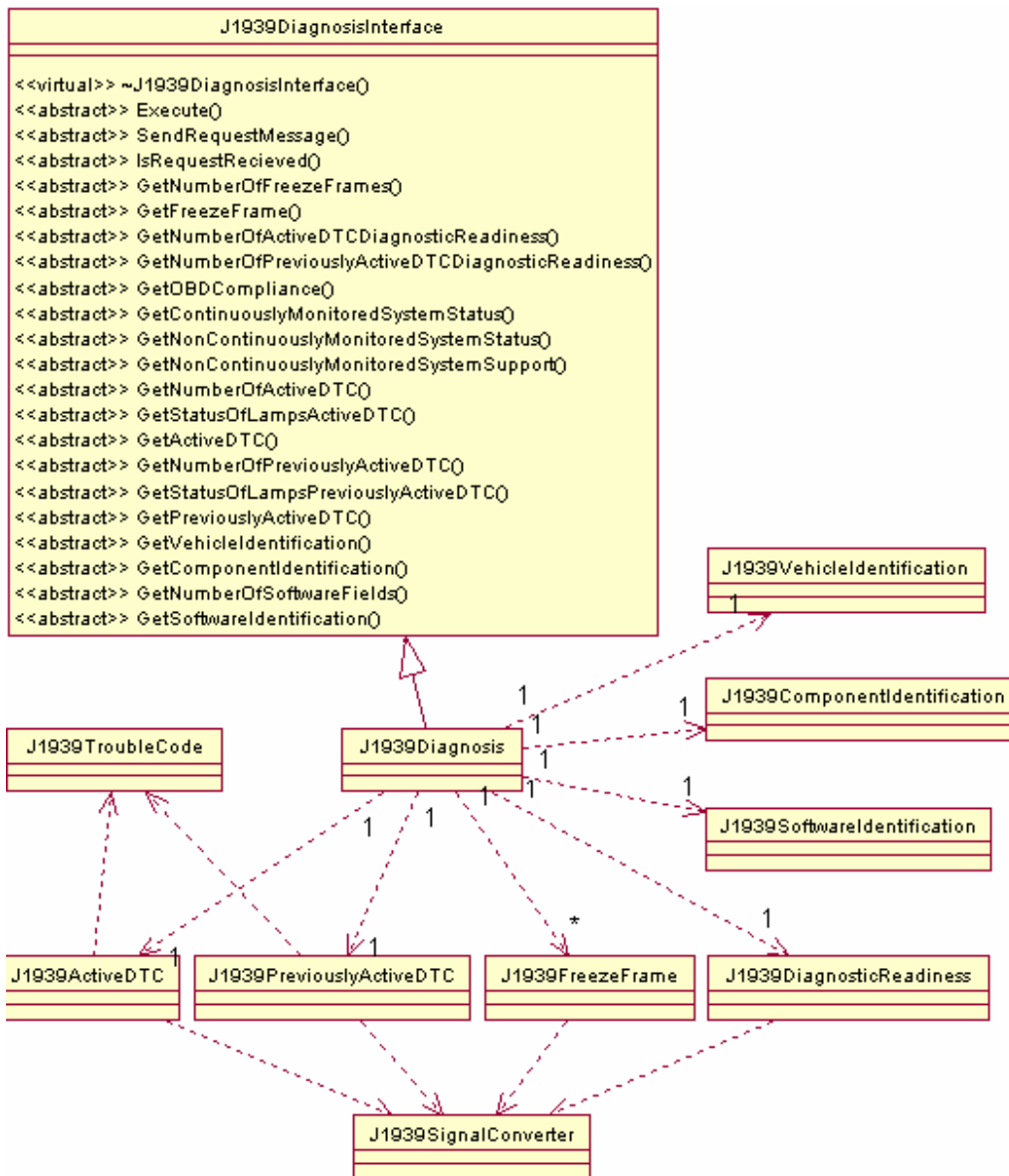
Lagrar information om fordonets identifiering.

J1939ComponentIdentification

Lagrar information om komponenterna i fordonet, serienummer på komponenter, motortillverkare mm.

J1939SoftwareIdentificaron

Lagrar information om den mjukvara som finns i fordonet.



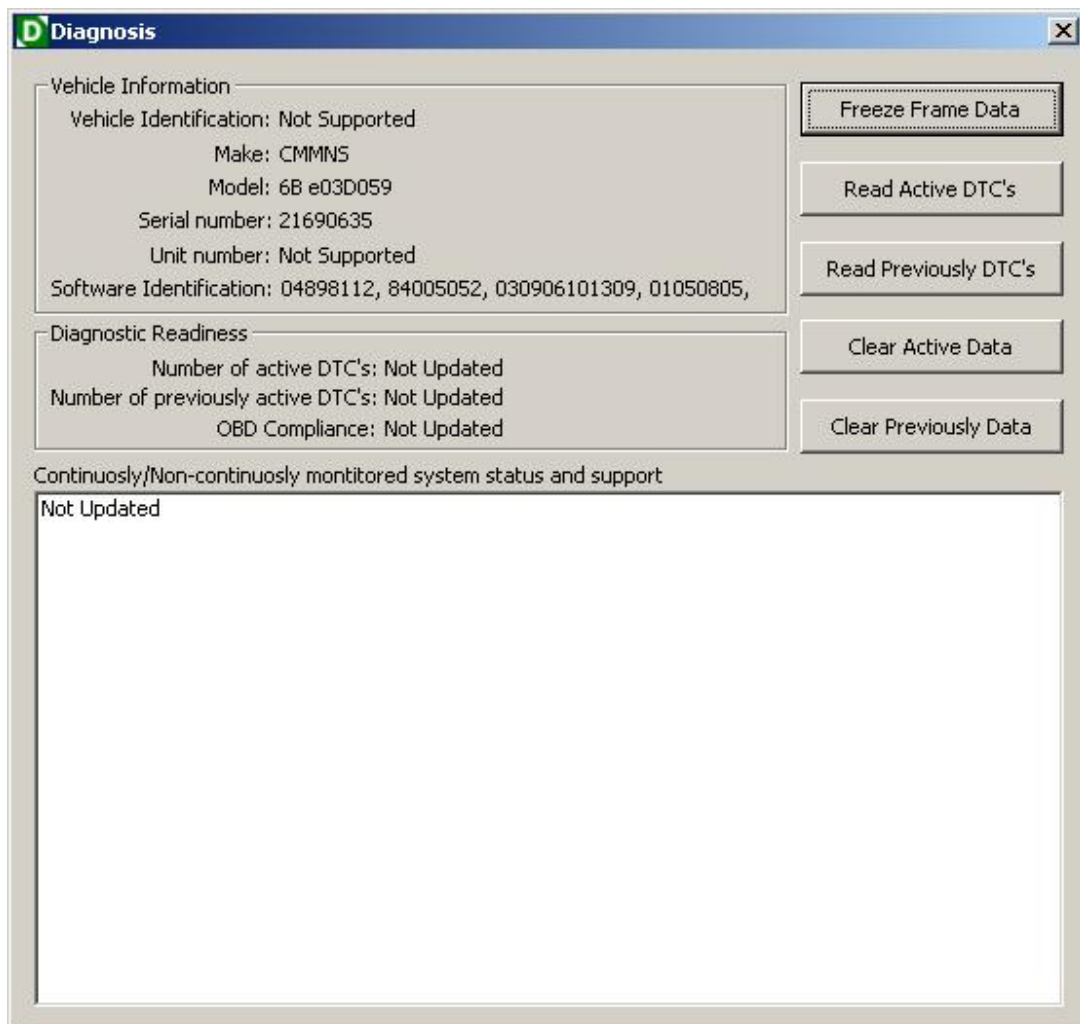
Figur 10. Klasserna i J1939Diagnostics

6.5 Användargränssnitt

Nedan kommer en kortare beskrivning av användargränssnittet och dess funktionalitet att ges.

6.5.1 Diagnostik

Presentationen av diagnosinformation som är specifik för bandvagnen presenteras i en egen modul i DIS applikationen. När modulen startas läses viss information ut ur fordonet och presenteras (se bild). Användaren har här tillgång till information angående fordonet som ska diagnostiseras samt information rörande hur många aktiva och tidigare aktiva felkoder som finns lagrade. Listan längst ner i dialogen presenterar statusen för de kontinuerliga och icke-kontinuerliga övervakningssystemen.



Figur 11. Dialogen som presenteras när diagnos modulen startas.

Längst till höger i dialogen finns funktioner tillgängliga för användaren att begära ytterligare diagnosinformation från fordonet samt att radera/nollställa felkoder och övrig diagnosinformation. Övrig diagnosinformation som kan begäras är, freezeframes, aktiva felkoder samt tidigare aktiva felkoder.

Presentation av freezeframes

Presentationen av freezeframes består av två delar. Den första delen är en lista där alla freezeframes presenteras. Här kan användaren få information om vilken felkod (DTC) som genererat freezeframen, namnet på signalen, hur många gånger felet inträffat samt varför felkoden genererats (Failure Mode Identifier). Namnet på signalen hämtas från Hägglunds befintliga fordonsdatabas, eftersom denna inte är anpassad för diagnos av bandvagnar så finns inte alla namn tillgängliga. Ifall signalen inte stöds av fordonsdatabasen presenteras strängen "Signal name not available" i namnfältet. Failure Mode Identifier beskriver varför felet har inträffat, det finns 32st olika beskrivningar varför ett fel kan uppstå.

I bilden nedan visas ett exempel på hur FMI används. Felkoden 110 återkommer två gånger i listan, det första felet har inträffat en gång och berodde på att det avlästa värdet låg över det specificerade gränsvärdet och att värdet låg så mycket över att det ska tolkas som allvarligt. Andra gången felkoden 110 återkommer i listan så har felet uppstått fyra gånger. Anledningen är att värdet på signalen har överskridit gränsvärdet men inte så mycket att det ska tolkas som allvarligt utan istället som ett fel av moderat grad.

Andra delen i presentationen av freezeframes är utökad information för en given freezeframe. Användaren väljer en freezeframe ur listan för vilken utökad information ska visas. Här presenteras den information som lästes av i det ögonblick som felet upptäcktes och blev aktivt. Här får användaren tillgång till information som t.ex. varvtal, motorns kylvattentemperatur och hur mycket motorn belastades när felkoden blev aktiv.

Previously Active Diagnostic Trouble Codes

Lamp Status

MIL Status: OFF
 Red Stop Lamp Status: OFF
 Amber Warning Lamp Status: OFF
 Protect Lamp Status: OFF

Previously Active Diagnostic Trouble Codes

DTC	Description	Occurrence co...	Failure Mode Identifier
105	ECAN.Intake Manif...	0	Data valid but above normal operating range - Moderately severe level
627	Signal name not av...	2	Data erratic, intermittent or incorrect
1077	Signal name not av...	2	Special instructions
110	ECAN.Engine Coola...	1	Data valid but above normal operational range - severe level
110	ECAN.Engine Coola...	4	Data valid but above normal operating range - Moderately severe level
441	Signal name not av...	12	Data valid but above normal operating range - Least severe level
723	Signal name not av...	1	Data erratic, intermittent or incorrect
168	ECAN.Electrical Pot...	8	Data valid but below normal operating range - Moderately severe level

Figur 13. Bilden som visar tidigare aktiva felkoder.

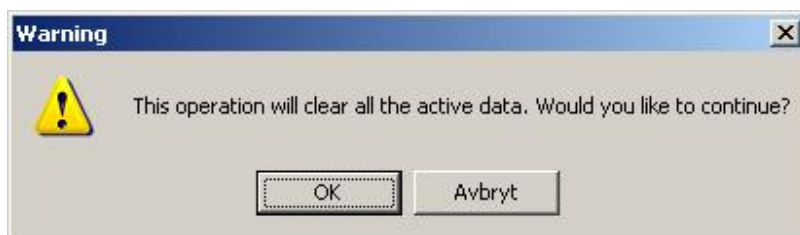
Ifall användaren försöker att begära aktiva eller tidigare aktiva felkoder från fordonet trots att det inte existerar några visas en dialog för användaren som informerar om detta (se bilden nedan).



Figur 14. Den dialog som kommer fram när inga felkoder existerar.

Nollställning av aktiv och tidigare aktiv information

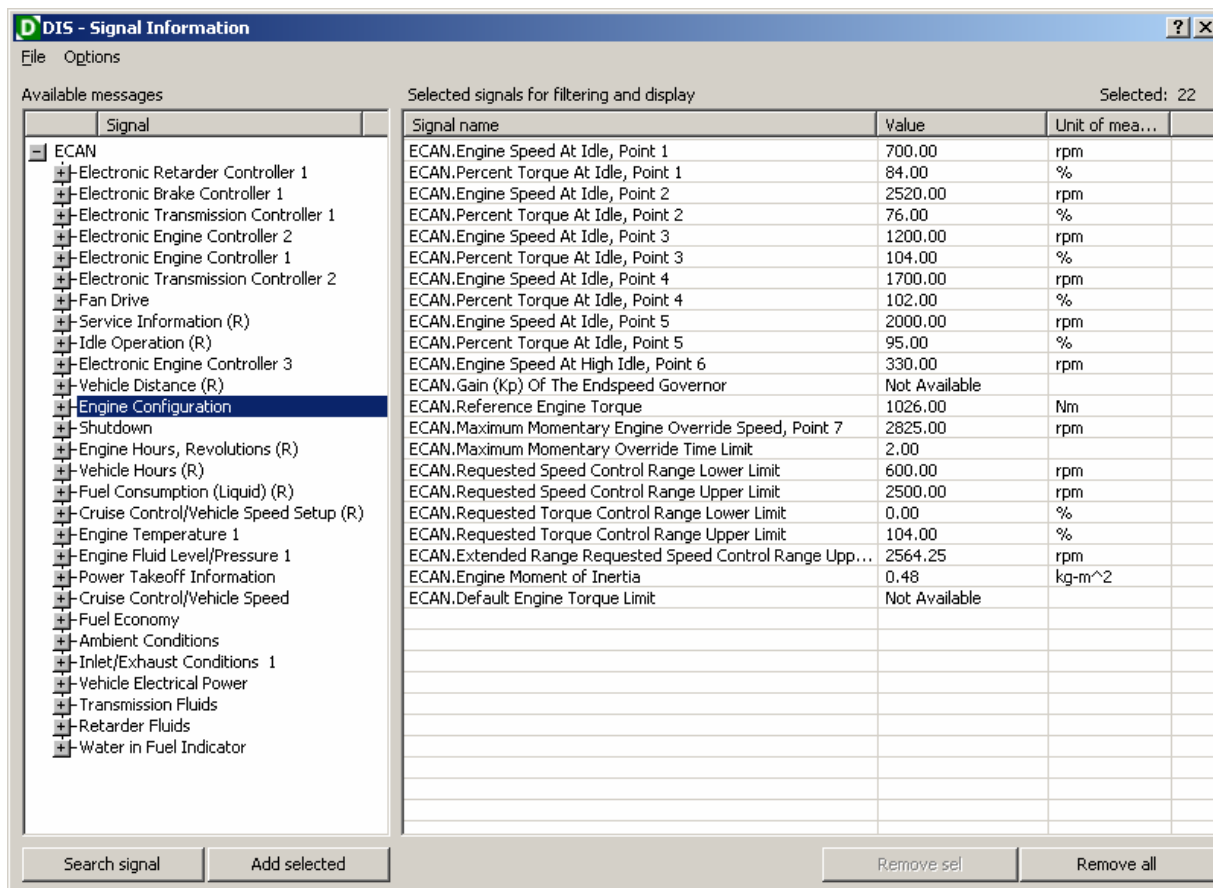
I Huvuddialogen finns det möjlighet för användaren att utföra operationer som raderar den aktiva och tidigare aktiva information som är lagrad i fordonet. Denna operation nollställer motsvarande felkoder, freezeframes, kalibrerings-data samt resultat av fordonets självtester och övervakningssystem. När användaren försöker utföra en sådan operation presenteras en dialog där användaren får bekräfta att operationen ska utföras (se bild 14).



Figur 15. Dialog för att bekräfta att en nollställnings operation ska utföras.

6.5.2 Signalhantering

Presentationen av det inlästa datat sker via samma modul som presenterar SIL-signaler, nämligen Signal Information. När modulen startas får man se vilka signaler som stöds, och för att få ett uppdaterat värde för en signal väljer man den i listan. Signaler som uppdateras automatiskt presenteras också automatiskt i fönstrets högra fält, medan signaler som måste begäras istället begärs när de markeras och presenteras när svaret kommer. Detta sker dock så pass snabbt så man vid användning av programmet inte märker någon skillnad på de olika signaltyperna. För att särskilja de två typerna har vi markerat signaler som måste begäras med "(R)" på slutet.



Figur 16. Bild över hur signaler presenteras i DIS med modulen Signal Information.

6.6 Lösningens begränsningar

De begränsningar som existerar är till den största delen funktionalitet hos J1939 som vi valt bort i vår lösning. Till dessa hör möjligheten att diagnosapplikationen själv ska kunna skicka meddelanden av längd större än åtta bytes, det är ett medvetet designval eftersom de enda meddelanden som behöver skickas från DIS är korta request-meddelanden för de signaler som kräver det samt den diagnostiska informationen. Detta designval innebär även att lösningen saknar möjligheten att öppna egna uppkopplingar enligt RTS/CTS (Se appendix B).

Utöver dessa begränsningar som inte påverkar lösningens tänkta funktionalitet finns även vissa begränsningar i signalhanteringen. Denna begränsning ligger i att den generella signalhanteringen inte hanterar signaler av variabel längd. Vi har dock löst detta genom att hårdkoda in hantering av dessa

signaler i systemet. Detta är inget användare av programmet uppfattar, men ändå en klar brist som i designen. Alternativa lösningar diskuteras i kapitlet Framtida arbete.

Det var från början tänkt att lösningen skulle användas för att utläsa information både från bandvagnens motor och växellåda. När det gäller data från motorn stöds den mesta informationen som skickas, all realtidsdata stöds via fordonsdatabasen och de flesta diagnostiska meddelanden stöds via diagnoslagret. Vi valde dock vid designen av systemet att inte implementera alla diagnostiska meddelanden då detta skulle vara för tidskrävande samt att vissa av dessa inte föll inom ramen för examensarbetet. Under implementationen av programmet har den tekniska specifikationen från motortillverkaren Cummins spelat en central roll, då den tydligt visat vilken typ av data som skickas på CAN-bussen och hur den ska hanteras. Den informationen har sedan tillsammans med J1939-specifikationen stått som grund för designen och implementationen.

När det gäller data från växellådan har vi dessvärre inte haft tillgång till samma dokumentation och detta har inneburit att vi enbart kan läsa ut den information som automatiskt skickas på fordonets CAN-buss, eftersom vi helt enkelt inte känner till vilka övriga meddelanden som stöds.

Eftersom Hägglunds befintliga diagnosapplikation DIS levereras till en mängd olika länder har applikationen inbyggt språkstöd. I dagsläget har DIS inbyggt stöd för Svenska, Engelska, Finska samt Holländska. Vår implementation saknar idag stöd för några av språken och stödjer endast Engelska. Denna begränsning måste lösas ifall systemet ska kunna levereras till kunder med krav på andra språk än engelska.

7 Framtida arbete

Detta kapitel presenterar förslag på framtida arbete för att förbättra programvaran samt att lösa några av begränsningarna som presenterats i lösningens begränsningar (kap 6.6)

Framförallt är det briserna i det begränsade stödet av J1939:s alla typer av diagnosmeddelanden samt lösningen av signalhanteringen som bör ses över. För att främst uppnå den funktionalitet som eftersträvats samt för att nå målen med examensarbetet så har den först nämnda begränsningen inte haft hög prioritet. För att systemet ska stödja alla typer av diagnosinformation som specificeras i J1939 krävs att transportlagret byggs ut för att hantera sändning av meddelanden med en längd större än åtta bytes. Framför allt är det metoderna för sändning av meddelanden som bör ses över för att erbjuda sändning av BAM/RTS från transportlagret. I dagsläget klarar transportlagret i applikationen endast av att skicka meddelanden innehållande åtta bytes data, dvs. Signal CAN Frames.

För att åtgärda signalhanteringen skulle det dock behövas större förändringar. Som det är nu återvinns den mesta av funktionaliteten hos den befintliga signalhanteringen av SIL-signaler som bygger på information som hämtas ur fordonsdatabasen. Fordonsdatabasen har stöd för J1939-signaler, men stödet är endast en mindre anpassning av vanliga SIL-signaler och den erbjuder inte full hantering av alla J1939-signaler. Dels saknas det stöd för meddelanden och signaler av variabel längd, databasen innehåller även dubbelinformation. Vid skapandet av meddelanden i fordonsdatabasen anges bland annat meddelandets identifierare, dess ursprung och destination.

Enligt uppbyggnaden av identifieraren för ett J1939-meddelande (Se appendix B) innehåller identifieraren redan den informationen. Det finns fler exempel som indikerar att fordonsdatabasen inte är fullt optimerad för J1939-signaler och en bättre anpassad hantering skulle behövas för att erbjuda full hantering av dessa signaler.

Eftersom ingen teknisk specifikation från växellådetillverkaren Allison funnits har inte möjligheten att bygga in fullt stöd för kommunikation med växellådan genomförts. Vi har dock konstruerat programvaran så att enbart små förändringar krävs för att hantering av diagnostisk data ska stödjas även från växellådan. Hantering av signaler från växellåda kan dock enkelt läggas till genom

användning av fordonsdatabasen och kräver inga förändringar i den nuvarande lösningens konstruktion.

För att lösa problemet med språkstöd (se lösningens begränsningar kap 6.6) kan vår diagnosmodul använda sig av samma lösning som DIS. DIS använder ett subsystem som heter LanguageDb (Language Database) för att kunna erbjuda språkstöd. Vid start av programmet väljer användaren språk och all information som presenteras i användargränssnittet hämtas från LanguageDb. I vår lösning är den information som presenteras för användaren hårdkodad i applikationen i form av strängar och erbjuder i dagsläget endast engelska. Eftersom vår lösning integrerats i DIS kan vi använda oss av samma metod för att lösa problemet med språkstöd. Arbetet med att lösa detta är inte avancerat utan till största del tidskrävande eftersom alla strängar ska översättas till motsvarande språk och inkluderas i LanguageDb.

8 Diskussion

Som det framgår av rapporten så råder en viss förvirring rörande de olika formaten och standarderna. Dels finns en mängd olika organisationer som standardiserar OBD (ARB, EPA, WWH etc.), dessutom finns standarden i olika tappningar för olika typer av fordon. När examensarbetet påbörjades var meningen att fokus skulle ligga på OBD-II och huruvida det gick att implementera stöd för denna standard i DIS för att erbjuda diagnos av bandvagnar. Allt eftersom examensarbetet fortflutit har fokus dock skiftat till att mer handla om diagnostik enligt J1939 standarden istället för OBD-II som främst gäller lättare fordon, såsom bilar och lättare lastbilar.

Denna förändring av examensarbetets fokus har inneburit svårigheter i efterforskningarna då diagnostik mot tyngre fordon inte är lika dokumenterad som OBD-II. Den största mängden information kring ämnet har hämtats från ARB:s hemsida samt i de ISO/SAE-specifikationer som vi arbetat med. Informationen från ARB:s sida har mestadels bestått av de preliminära bestämmelser man tagit fram i arbetet med att skapa en ny standard för tyngre fordon.

Denna standard, kallad HD OBD , har utvecklats av ARB parallellt med arbetet med examensarbetet och vi har haft den i åtanke under utvecklingen av vår lösning. Samtidigt har det inte gått att fokusera helt på den eftersom när examensarbetet påbörjades befann sig standarden enbart utvecklingsstadiet samt att det fortfarande är oklart vilket genomslag den kommer få. Som det sett ut tidigare har industrin anammat ARB:s standarder men en viss förvirring råder numera pga. den parallella utvecklingen av WWH OBD, som är tänkt som en generell standard gällande samtliga fordon, oavsett vikt. HD OBD har dock ett rejält försprång i och med att den redan klubbats igenom samt att både aktörer i branschen och organisationer som EPA uttryckt sitt stöd. EPA har dessutom påbörjat arbete med en nationell bestämmelse som ska baseras på HD OBD.

Efter inledande svårigheter innan arbetet mot den nyare bandvagnen BVS10 påbörjades så har arbetet med att implementera programvaran gått friktionsfritt. Kommunikationen på den nyare bandvagnen var bättre dokumenterad än på den äldre BV206 med Steyr-motorn. Tack vare motortillverkarens (Cummins) tekniska specifikation konstaterades att J1939 används som kommunikationsprotokoll i den nya bandvagnen. Detta protokoll stöds i OBD-II genom en undantagsregel samt var det förmodade valet av kommunikationsprotokoll för den nya HD OBD standarden. När det blev klart att vi skulle jobba mot BVS10 så användes en dryg vecka till

fördjupning inom J1939 samtidigt som möjligheterna att läsa information från motor och växellåda utreddes. Det visade sig att det fanns färdiga hårdvarulösningar på marknaden för att koppla in sig mot hårdvara som kan kommunicera enligt J1939 och efter klartecken från Hägglunds började sökandet efter ett hårdvaruinterface som bäst lämpade sig för vårt arbete. Vi fann att motortillverkaren till BVS10:an, Cummins, hade en egen hårdvarulösning kallad Inline som erbjöd den önskade funktionaliteten. Inline finns dock i flera olika versioner och när vi diskuterade vilken som var bäst lämpad upptäcktes ytterligare en möjlighet, nämligen att använda Hägglunds befintliga hårdvara, testboxen kallad DTB. Denna används i dagsläget för att läsa diagnostisk information från stridsfordonen. Efter att ha konsulterat sakkunniga inom området visade det sig att detta var möjligt och än en gång hade examensarbetet tagit en ny skepnad. Denna lösning visade sig vara den lämpligaste eftersom det innebär att ingen extern hårdvara behövdes köpas in.

Under de regelbundna testkörningar som gjordes under arbetets gång upptäcktes en del avvikelser mellan informationen som skickades från bandvagnen och J1939-specifikationen. Dessa avvikelser förbryllades oss till en början men efter att ha analyserat informationen upptäcktes var avvikelserna lokaliserad. Det visade sig att tolkningen av så kallade freezeframes gjordes enligt standarden, eftersom en formel i J1939-specifikationen inte följdes av motortillverkaren Cummins. Vi vet ej om detta är en allmän feltolkning av specifikationen eller om den är Cummins-specifik, men den måste beaktas för att kunna läsa ut information på korrekt sätt.

Felet låg i tolkningen av freezeframe information, enligt J1939 ska denna tolkas på formatet $A=B+C$. Där A (totala längden på meddelandet) är summan av längden av freezeframen (B) plus längden av tillverkarspecifik data (C). Denna formel upprepas för varje freezeframe och datan skickas på formen ABCABCABC etc.

Vid användandet av formeln ovan för att läsa ut freezeframes visade det sig att dessa blev omöjliga att tolka. Längden på den första freezeframen fick en orimlig längd och den resterade informationen var omöjlig att tolka. Efter att delat upp datan efter det format vi antog att den skulle vara op upptäcktes att formeln man istället använt sig av var $A = A+B+C$. Cummins hade alltså i sin implementation inkluderat byten som specificerar längden av en freezeframe i längden för hela meddelandet. Detta innebär att vid tolkning av den första freezeframen uppstod en förskjutning med

en byte som innebar att den efterföljande informationen blev obegriplig, och när efterföljande freezeframe skulle läsas ut blev längden felaktig.

Vi har anpassat vår lösning så den hanterar längden enligt den formel som Cummins använder, men det bör noteras att detta är en avvikelse som innebär att programvaran inte fungerar för att läsa ut freezeframes ur motorer som använder formeln som existerar i J1939-specifikationen.

9 Slutsats

Om vi ska återkoppla till de mål som satts upp kan man konstatera att stöd för OBD-II i den form vi beskriver i kapitel 4.2 inte finns fullt ut i bandvagnarna då standarden inte är utformad mot tyngre fordon som BVS10. Istället är det snarare standarden HD OBD som är av intresse och allt tyder på att bandvagnarna i dagsläget har stöd för denna standard alternativt enkelt kommer att kunna modifieras för att stödja den. Dessa modifikationer kommer främst röra mjukvaran eftersom hårdvaran har stöd för J1939, vilket är ett godkänt protokoll enligt HD OBD. När det gäller DIS och dess stöd för OBD-II så kan man genom att studera kraven för extern diagnosenhets gällande OBD-II (Se Appendix F) konstatera att stöd inte existerade vid examensarbetets start. Efter avslutat examensarbete har dock DIS tagit ett stort steg i riktning mot att uppfylla kraven på externdiagnosenhet enligt OBD-II standarden, detta genom att uppfylla punkt 2-6 i kravlistan gällande de funktioner som måste uppfyllas. Den första punkten, automatisk kontroll och verifiering av vilket kommunikationsprotokoll som används i fordonet, finns inte implementerad av den anledningen att vi enbart haft möjlighet att jobba mot J1939 under examensarbetets gång.

Sammanfattningsvis kan man säga bandvagnarna inte uppfyller kraven på ett OBD-II system eftersom de saknar de komponenter som krävs för att övervaka avgasutsläppen hos fordonet. Något som är ett av huvudsyftena med OBD-II systemen. Rimligare är det att anta att bandvagnarna istället kommer att vara HD OBD kompatibla eftersom de grundläggande funktionerna finns. Vissa små förändringar kan komma att krävas för fullt stöd. ARB har gjort en ungefärlig beräkning att ingen motortillverkare behöver bygga om sina motorer till en kostnad större än 132\$ för att stödja HD OBD fullt ut. Denna summa är obetydlig i proportion till utvecklingskostnaden och tillverkningskostnaden för en motor (och en bandvagn). Dessutom kommer standarden att införas i etapper, där varje motortillverkare ska kunna erbjuda en motorfamilj med stöd för HD OBD först 2010, och ha stöd för standarden i samtliga motorer först 2016.

Referenser

- [1] Görans Bilsida. http://home.swipnet.se/g_gson/, Februari 13 2006.
- [2] ISO 15031-6 Diagnostic trouble code definitions, 2005-11-22
- [3] Control of Air Pollution From New Motor Vehicles and New Motor Vehicle Engines; Modification of Federal On-Board Diagnostic Regulations for: Light-Duty Vehicles, Light-Duty Trucks, Medium Duty Passenger Vehicles, Complete Heavy Duty Vehicles and Engines Intended for Use in Heavy Duty Vehicles Weighing 14,000 Pounds GVWR or Less (EPA), 2005-12-20
- [4] Wikipedia <http://www.wikipedia.org/>, Februari 12 2006.
- [5] ISO 9141. Road vehicles -- Diagnostic systems -- Requirements for interchange of digital information.
- [6] ISO 9141-2 Road vehicles -- Diagnostic systems -- Part 2: ARB requirements for interchange of digital information.
- [7] SAE J1939 The Real-Time CAN Solution for Heavy Duty Application
- [8] ISO 14230 Diagnostic systems -- Keyword Protocol 2000
- [9] ISO 15765 Road vehicles -- Diagnostics on Controller Area Networks (CAN)
- [10] SAE J1979 E/E Diagnostic Test Modes -- Equivalent to ISO/DIS 15031-5; Vehicle E E Systems Diagnostic Standards Committee, April 30, 2002
- [11] SAE J1850 Class B Data Communications Network Interface, Vehicle Architecture For Data Communications Standards
- [12] The SAE J1939 set of profiles, <http://www.can-cia.org/j1939based/j1939/>, 2006-02-22
- [13] SAE J1939/71 Vehicle Application Layer, Truck Bus Control And Communications Network Subcommittee, USA, December 2004
- [14] SAE J1939/73 Application Layer—Diagnostics, Truck Bus Control And Communications Network Subcommittee, USA, March 2004
- [15] California Code Regulations, Section 1968.2, Malfunction and Diagnostic System Requirements for 2004 and Subsequent Model-Year Passenger Cars, Light Duty Trucks, and Medium-Duty Vehicles and Engines (OBD II), ARB, 3 November 2005.
- [16] California Code of Regulations, Section 1968.5, Enforcement of Malfunction and Diagnostic System Requirements for 2004 and Subsequent Model-Year Passenger Cars, Light-Duty Trucks, and Medium-Duty Vehicles and Engines, ARB, 3 November 2005.
- [17] California Code Regulations, Section 1971.1, On-Board Diagnostic

System Requirements for 2010 and Subsequent Model-Year Heavy-Duty Engines (HD OBD), ARB, 21 Juli 2005.

[18] Final Statement of Reasons for Rulemaking, Including Summary of Comments and Agency Response, ARB, 21 Juli 2005.

[19] California Code Regulations, Section 1971.1, On-Board Diagnostic System Requirements for 2010 and Subsequent Model-Year Heavy-Duty Engines (HD OBD), ARB, 3 Juni 2005.

[20] World-Wide Harmonised Heavy-Duty OBD (WWH-OBD) Proposed terms of reference for a GRPE ad hoc working group, WWH-OBD, Maj 2004.

[21] ISO 15031-3 Diagnostic connector and related electrical circuits, specification and use, 2004-07-16

[22] ISO 15031-4 External test equipment, 2005-06-15

[23] ISO 15031-5 Emissions-related diagnostic services, 2006-01-09

Appendix A – ISO 11898 (CAN)

ISO 11898 är ett protokoll som utvecklades i Europa för att användas i personbilar. Tack vare de funktioner den erbjuder har standarden spritt sig till andra användningsområden såsom maskiner i industrin och tyngre fordon. Protokollet finns standardiserat som ISO 11898. Anledningen till att CAN utvecklades var för att bilarna fylldes med mer och mer avancerad elektronik, och att det växte fram ett behov för ett standardiserat sätt att kommunicera bland dessa olika enheter. CAN erbjuder en komplett lösning som sammanbinder all elektronik i t.ex. en bil till ett enda nätverk. Hastigheten utlovas till upp till 1Mbit/s över seriell databuss.

Standarden kan mappas mot OSI-modellen där man beskriver det fysiska samt datalänklaget.

Dokumentstruktur

Standarden är fördelat på fyra olika dokument, dessa är:

1. 11898-1

Beskriver datalänklaget enligt OSI-modellen.

2. 11898-2

Beskriver det fysiska lagret enligt OSI-modellen. Detta är det vanligaste fysiska lagret vid användning av CAN och det garanterar en överföringshastighet upp till 1Mbit/s över en maximal busslängd av 40m.

3. 11898-3

Beskriver ett alternativt fysiskt lager med en lägre överföringshastighet, 125kbit/s, men som istället är feltolerant.

Datalänklaget

CAN baseras på broadcast-mekanismen, det vill säga meddelanden skickas till alla på nätverket. Varje meddelande har en identifierare som är unik över hela nätverket och som utgör en beskrivning av meddelandets innehåll och prioritet. Meddelandets prioritet är viktig när flera noder på nätverket försöker sända samtidigt.

- Cyclic Redundancy Check (CRC)
- Frame check
- ACK errors

CAN implementerar även två mekanismer för felhantering på bitnivå:

- Monitoring
- Bit stuffing

Om ett eller flera fel upptäcks vid åtminstone en nod enligt de fem ovan beskrivna mekanismerna, avbryts sändningen genom att man skickar en så kallad error frame. Detta felmeddelande förebygger att andra noder på nätverket tar emot det felaktiga meddelande och garanterar därför nätverkets konsistens. Den sändande noden skickar sedan om det felaktiga meddelandet.

Det fysiska lagret

CAN använder en bitkodnings teknik kallad Non-Return-to-Zero (NRZ) som innebär att signalnivån är konstant över en bitlängd och det endast krävs en bitlängd för att representera en bit (till skillnad mot så kallad Manchester-kodning eller Pulse-width-modulation). Signalnivån kan vara konstant över en längre tidsperiod, därför krävs mätningar som säkerställer att det tillåtna intervallet mellan två signaler inte överskrids. Detta är viktigt för att kunna synkronisera sändningen av data och för att garantera detta använder man en metod kallad Bit stuffing. Bit stuffing innebär att man sätter in en extra bit efter varje fembitarssekvens där alla bitarna har samma värde. Självklart måste mottagaren kunna packa upp denna data och i samma process kontrollera att den är på rätt format.

Appendix B – SAE J1939

J1939 är ett höghastighets kommunikationsprotokoll som är tänkt att användas till att låta ECU:s på ett fordon att kommunicera med varandra. J1939 är en vidareutveckling av ett äldre protokoll kallat J1708/J1587. J1939 uppfyller allt hos det äldre J1708/J1587. Protokollet kan mappas mot OSI-modellen men man definierar inte alla sju lager.

Dokumentstruktur

Protokollet specificeras i en mängd dokument, som namnges enligt följande:

1. J1939

Generell beskrivning av protokollet, mappning mot OSI modellen etc.

2. J1939/0X

Beskriver specifika versioner av nätverket. Två utkast existerar, 01 beskriver protokollet för lastbilar och bussar och 02 för jordbruksmaskiner.

3. J1939/1X

Mappas mot det fysiska lagret i OSI-modellen. Tre dokument existerar, 11,12 och 13. De två förstnämnda beskriver fysisk överföring av data med två olika typer av kablage och den sistnämnda beskriver den diagnostiska kontakten som används vid överföring av data.

4. J1939/2X

Beskriver datalänklaget i OSI-modellen. Det finns bara ett dokument i denna kategori (21), och inga alternativ är tillåtna.

5. J1939/3X

Beskriver nätverkslaget i OSI -modellen. Det enda nu existerande dokumentet är 31.

6. J1939/4X

Beskriver transportlaget i OSI -modellen.

7. J1939/5X

Beskriver sessionslaget i OSI -modellen.

8. J1939/6X

Beskriver presentationslaget i OSI-modellen.

9. J1939/7X

Beskriver applikationslagret i OSI-modellen. Tre dokument finns i denna kategori, 71 beskriver ett generellt applikationslager, 72 (bara ett utkast ännu) beskriver ett applikationslager för en virtuell terminal och 73 beskriver applikationslagret för diagnostik.

10. J1939/8X

Mappas ej mot OSI-modellen. Dokument 81 beskriver nätverkshantering, adresstilldelning och liknande.

Meddelandeformat

J1939 använder det allmänna CAN-protokollet för att skicka meddelanden på nätverket. Varje meddelande innehåller en identifierare som definierar meddelandets prioritet, avsändare och vilken typ av data meddelandet innehåller. Kollisioner undviks genom en avskiljningsprocess som sker när meddelandets identifierare skickas. Om två eller flera meddelanden skulle skickas samtidigt kommer endast meddelandet med högst prioritet sändas i sin helhet.

CAN-protokollet som används för signalöverföring definierar två typer av identifierare, en 11 bitar lång (CAN Standard Frame) och en 29 bitar lång (CAN Extended Frame). J1939 använder sig av det senare, längre formatet men man tillåter samtidigt att meddelanden med den kortare identifieraren skickas på samma nätverk.

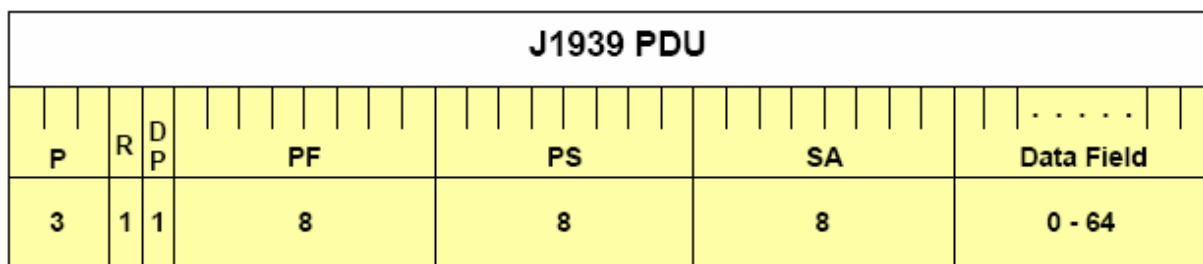
J1939-PDU

De första 3 bitarna av dess 29-bitars identifierare beskriver meddelandets prioritet, där ett lågt värde har högre prioritet än ett högt. Nästa bit i identifieraren är reserverad för framtida användning och skall alltid vara satt till 0. Efter den kommer 9 bitar där den första anger Data Page (DP) och de efterföljande PDU Format (PF). PDU står för Protocol Data Unit och motsvarar hela meddelandet. DP används som en sidväljare för meddelandet. Alla nu definierade meddelanden använder sida 0, och sida 1 är för att ge plats för fler meddelanden i framtiden. PF fältet anger vilken typ meddelanden har samt hur de nästkommande 8 bitarna ska tolkas. Om PF har ett värde under 240 så är meddelandet av PDU1-typ och då ska nästa 8 bitars tolkas som destinationsadress (DA) för meddelandet. Om värdet är 240 eller större så är meddelandet på PDU2-format och då ska nästa 8 bitar ses som en förlängning av meddelandes format (Group

Extension, GE). PF och PS tillsammans kallas Parameter Group Number (PGN). Det kan existera 8672 olika PGN:s per datasida.

De flesta meddelanden som är definierade i J1939 är tänkta att skickas till samtliga noder i nätverket (broadcast) enligt PDU2-formatet. I och med meddelandets uppbyggnad kan meddelanden av PDU2-typ helt enkelt inte skickas till specifika destinationer. När ett meddelande skickas till en specifik destination måste följaktligen PDU1 användas.

De sista 8 bitarna av meddelandet anger dess avsändare (Source Address, SA). Alla enheter i nätverket måste ha en unik adress.



Figur 19. Uppbyggnaden hos en J1939-PDU.

Meddelandetyper

J1939 definierar fem olika typer av meddelanden, dessa är commands, requests, responses, acknowledgements och group functions. Meddelandetyperna känns igen på dess PGN. De olika typerna används enligt beskrivning:

Command

Skickar ett kommando, antingen till samtliga eller en specifik nod, både PDU1 och PDU2 kan alltså användas.

Request

Används för att begära information från samtliga eller från en specifik nod. Om begäran skickas globalt måste alla noder som har stödjer den typen av request svara, övriga ska bara ignorera meddelandet.

Response

Antingen ett svar på en förfrågan eller data som skickas regelbundet. Detta är den vanligaste formen av data i J1939.

Acknowledgement

Används för att verifiera att en överföring gått rätt till, alternativt för att indikera att något gick fel.

Kommunikationstyper

Kommunikationen inom J1939 kan ske på tre olika sätt, dessa är:

1. Destinationsspecifik kommunikation med PDU1
2. Broadcast-kommunikation med PDU2
3. Tillverkardefinierad (Proprietary) kommunikation med antingen PDU1 eller PDU2

Samtliga kommunikationsätt har sitt användningsområde. Destinationsspecifik kommunikation behövs när ett meddelande måste riktas till en viss destination. J1939 definierar till exempel ett vridmomentsmeddelande som ska sändas till motorn. Om det finns mer än en motor, måste meddelandet skickas till rätt motor och destinationsspecifik kommunikation är nödvändig.

Broadcast-kommunikation har flera användningsområden, bland annat:

- Meddelanden sänds från en eller flera noder till en nod
- Meddelanden sänds från en eller flera noder till flera noder

Den tredje formen av kommunikation, den tillverkardefinierade, finns definierad genom två PGN:s. En för broadcast och en för destinationsspecifik kommunikation. Tillverkarna är fria att använda denna kommunikationsform enligt eget behov.

Ta emot meddelanden

Det finns flera tekniker för hur man ska ta emot meddelanden från nätverket, men följande generella beskrivning kan sägas alltid gälla:

1. Om meddelandet är en destinationsspecifik begäran eller kommando, måste noden som tog emot det jämföra sin egen adress med meddelandes DA. Om de är lika, måste meddelandet hanteras på rätt sätt och skicka ett kvitto på detta genomförts.
2. Om meddelandet är en begäran skickat till alla noder (broadcast) måste alla, även avsändaren svara om data finns tillgängligt.
3. Om meddelandet är skickat till alla, måste varje nod avgöra om det är relevant eller inte.

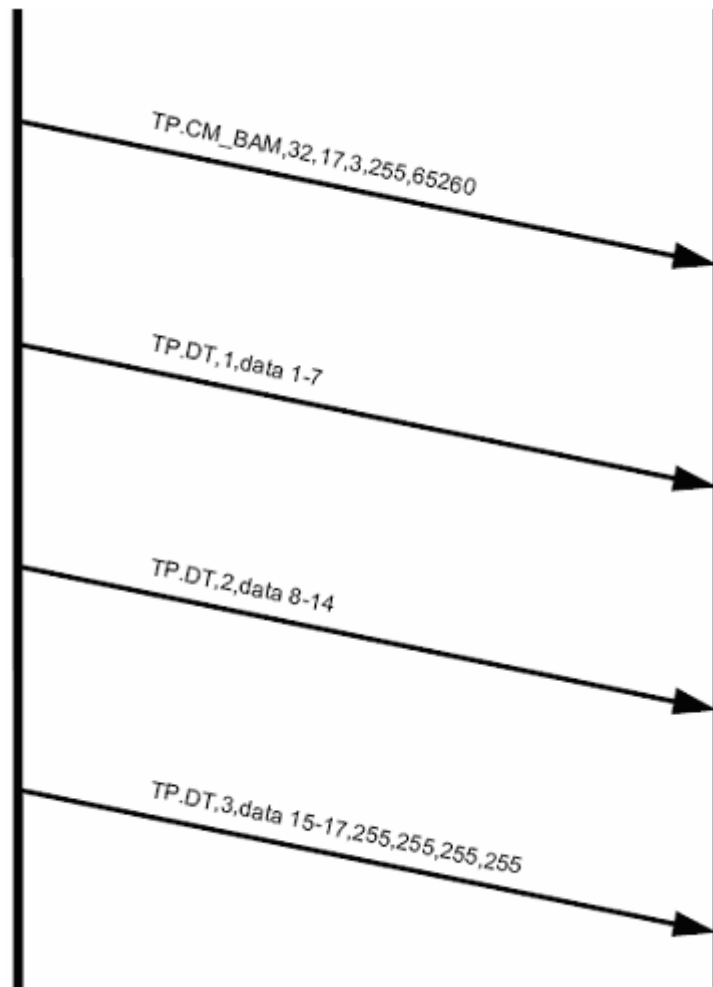
Transport

Eftersom ett CAN-meddelande bara kan hålla 8 bytes data men ett J1939 meddelande kan vara 1785 bytes stort så krävs funktioner för att skicka fragmenterad data i flera CAN-meddelanden. Detta hanteras av ett enkelt transportlager definierat i J1939/21. Transport-protokollet har mer eller mindre hand om två funktioner, att plocka isär meddelande med storlek större än 8 bytes till flera olika CAN-meddelanden samt att sköta hopsättning av dessa meddelanden.

Transportlagret hanterar meddelanden med stor data (större än 8 bytes) på två olika sätt, beroende på vilken kommunikationstyp som används. Om meddelandet har specifik destination skapas en uppkoppling mellan den sändande och den mottagande noden, och om meddelandet är en broadcast används något som kallas flerpakets broadcast. Transportlagret sköter enbart hantering av stora meddelanden.

Flerpakets broadcast

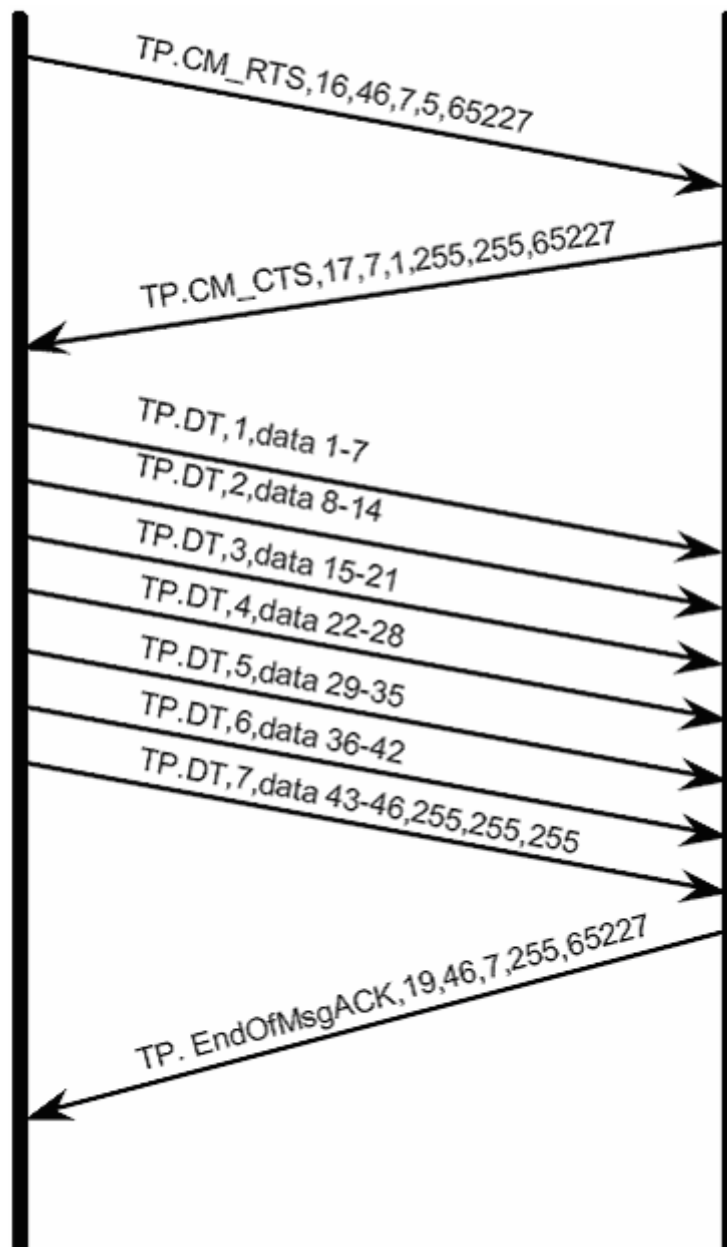
Stora meddelanden kan skickas till som en broadcast, och för att hantera detta används något som kallas BAM (Broadcast Announce Message). Ett BAM-meddelande skickas av den som vill sända det stora meddelandet och kan ses som en varning till alla noderna på nätverket. Denna varning säger att det kommer ett stort meddelande och att de mottagande noderna måste frigöra resurser för att kunna ta emot och sätta ihop meddelandet.



Figur 20. Dataflödet vid flerpakets broadcast. Först skickas en BAM för att varna alla noder att ett stort meddelande är på väg, sedan skickas meddelandet uppdelat i tre.

Uppkoppling

När ett stort meddelande skickas till en specifik destination sker detta på det vis att man först skapar en uppkoppling mellan noderna. Detta görs genom att avsändaren först skickar en RTS (Request To Send) till mottagarnoden. Denna RTS innehåller vilken typ av meddelande som ska skickas, storleken och hur många paket information kommer att delas upp i. Mottagarnoden svarar på denna RTS genom att skicka en CTS (Clear To Send) där man anger hur många paket man är beredd att ta emot. Avsändarnoden tar emot denna CTS och skickar så många paket som mottagarnoden sagt sig kunna ta emot. Detta upprepas till dess att alla paket är överförda, då mottagarnoden skickar en ett meddelande som säger att alla paket togs emot korrekt till avsändarnoden.



Figur 21. Dataflödet vid uppkopplad sändning. Först skickas en RTS till en viss nod, denna nod svarar med en CTS och den första noden skickar sedan all data. Den mottagande noden bekräftar att all data kommit fram genom att skicka ett ACK.

Fördefinierade värden

Applikationsspecifika parametrar och parametergrupper finns definierade i J1939/7X-dokumenten. Parametergrupper som används för kontroll och skötsel av nätverket finns definierade i J1939/21, J1939/31 och J1939/81. Tilldelningar för adresser och parametergruppnummer finns sammanfattade i J1939-dokumentet.

Parametergruppnummer (PGN)

Parametergruppnummer är tilldelade specifikt för att använda antingen PDU1-formatet eller PDU2-formatet. När väl tilldelningen skett kan den ena formatet inte använda sig utav det andra, det vill säga, ett parametergruppnummer för specifika destinationer kan ej skickas till en global adress och vice versa. Tilldelningen för ett parametergruppnummer skall innehålla dess prioritet, uppdateringsfrekvens, typ och längd av data.

Mycket av kommunikationen mellan noder tillverkade av en enda tillverkare behöver inte standardiseras då informationen oftast inte är intressant för andra noder på nätverket. I dessa fall kan den tillverkarspecifika (Proprietary) metoden användas.

Datafältsgruppering

För att minimera overhead bör olika meddelanden grupperas så att hela det 8 bytes stora datafältet fylls. Detta gäller för allt utom extremt tidskritiska meddelanden. Grupperingen bör ske på följande sätt:

1. Enligt gemensamt subsystem (Den nod som mäter och skickar data)
2. Med liknande uppdateringsfrekvenser
3. Enligt funktion (olja, bränsle etc.)

De tre punkterna ovan ska ses som riktlinjer då det är omöjligt att gruppera alla parametrar utan att bryta åtminstone en av punkterna ovan. Då alla parametrar i J1939 har en teknik för att informera om de är tillgängliga eller ej är det inte alltid nödvändigt att de kommer från samma nod.

Industrigrupp

För att möjliggöra att många olika industrier ska kunna använda J1939, används en industrigruppkod för att identifiera vilken industri en nod tillhör. Industrigrupp 0 är en speciell kategori som gäller alla typer av industrier, Alla noder som kan härröras till fler än en industrigrupp, t.ex. dieselmotorer ska tillhöra denna globala industrigrupp. Om en nod endast existerar inom en industrigrupp, är det önskvärt att den tilldelas den industrigruppens kod och inte den globala industrigruppen.

Industrigrupp	Industri
0	Global, gäller alla
1	Trafikfordon
2	Jordbruksfordon
3	Byggfordon
4	Marina fordon
5	Industri-process Control
6	Reserverad
7	Ej tillgänglig

Tabell 4, J1939 Industrigrupper.

Adresser

Eftersom noderna på nätverket adresseras med 1 byte kan man inte adressera fler än 254 enheter (254 och 255 är reserverade). De flesta noder i ett J1939-nätverk har en adress att föredra, och om denna redan är upptagen blir den tilldelad en annan ledig adress enligt procedurer beskrivna i J1939/81.

Adress	Industri
0 - 127	Reserverad för vedertagna noder
128 - 247	Reserverad för industrispecifika tilldelningar
248-253	Reserverad för speciella noder
254	Nulladress
255	Global adress

Tabell 5. J1939 Industrigrupper.

Appendix C – ISO 15765

ISO 15765 är en standard för fordonsdiagnostik som implementeras över ett CAN-nätverk specificerat enligt ISO 11898. Standarden har upprättats för att definiera gemensamma krav för fordonsdiagnostik över CAN. Även om standarden främst är tänkt att användas till diagnostik, har den även utvecklats för att stödja krav från andra CAN-baserade system. För att möta dessa krav har man konstruerat standarden enligt OSI-modellens sju lager.

Dokumentstruktur

Standarden definieras i en mängd dokument, dessa namnges enligt följande:

1. 15765-1

En översikt över standarden och de olika dokument som ingår.

2. 15765-2

Mappas mot nätverkslagret i OSI-modellen. Nätverkslagret som definieras är skräddarsytt för att användas med CAN som underliggande lager.

3. 15765-3

Beskriver en del av applikationslagret i OSI-modellen. Dokumentet täcker diagnostiska tjänster som ska hanteras av standarden.

4. 15765-4

Beskriver det fysiska-, datalänk- och nätverkslagrets i OSI-modellen. Dokumentet gör inskränkningar i ISO 15765-2 och ISO 11898 (CAN) i syftet att möta lagstiftade OBD-krav.

Det sker en viss överlappning mellan dokumenten. Anledning till detta är att standarden kan användas på två olika sätt, för att ställa upp krav för att möta tillverkarspecifik diagnostik, eller för att möta de lagstiftade OBD-kraven. I det tidigare fallet används ISO 11898 som datalänklagret, det underliggande fysiska lagret kan vara vilket CAN-baserat fysiskt lager som helst. Nätverkslagret för att möta dessa krav är 15765-2 och applikationslagret 15765-3. Tillverkaren är sedan fri att utveckla den diagnostiska applikationen enligt egna önskemål.

Om man istället vill möta den hårdare, lagstiftade OBD-standarden erbjuder även 15765 en inskränkning av den tidigare specifikationen där 15765-4 definierar ett fysiskt lager samt begränsningar till ISO 11898-1 och 15765-2. Applikationslagret och den diagnostiska applikationen definieras i ISO 15031-5 (Se tabell 1).

Detta appendix kommer att fokusera på tillverkardefinierad diagnostik. Detta för att det i huvudsak är den som används inom Hägglunds befintliga diagnostiska applikation som utvecklats mot stridsfordon 90.

Lager i OSI-modellen	Tillverkarspecifik diagnostik	Lagstiftade OBD-krav
Diagnostik	Tillverkarspecifikt	ISO 15031-5
Applikationslagret	ISO 15765-3	ISO 15031-5
Presentationslagret	E/A	E/A
Sessionslagret	E/A	E/A
Transportlagret	E/A	E/A
Nätverkslagret	ISO 15765-2	ISO 15765-4
Datalänklagret	ISO 11898-1	ISO 15765-4
Fysiska lagret	Tillverkarspecifikt	ISO 15765-4

Tabell 6. Hur ISO 15765 mappas mot OSI-modellen.

Nätverkslagret

Nätverkslagret är som tidigare nämnt främst definierat i 15765-2, och sedan vidare i 15765-4 med inskränkningar för att möta lagstiftningskrav. Nätverkslagret är utformat för att användas mot CAN enligt ISO 11898 men kan även användas mot andra CAN-lösningar.

15765-2 beskriver den grundläggande funktionaliteten hos nätverkslagret, ett lager för att kommunicera mellan noder på nätverket, t.ex. mellan två ECU:s eller mellan en ECU och en extern testenheter. Om meddelandena som ska skickas inte får plats i ett CAN-meddelande ska en segmenteringsmetod användas.

Tjänster till de övre lagren

Nätverkslagret erbjuder några tjänster till det överliggande lagret, dessa är överföring/mottagande av data samt möjlighet att sätta några av protokollets parametrar. Man delar upp dessa tjänster i två kategorier:

1. Kommunikationstjänster

Dessa tjänster möjliggör överföring av upp till 4095 bytes data. Följande tjänster finns definierade

- **N_USData.request:** Begär överföring av data, om det är nödvändigt segmenteras det.
- **N_USDATA_FF.indication:** Används för att signalera att början av ett segmenterat meddelande ska levereras till det övre lagret.
- **N_USData.indication:** Används för att leverera data till de övre lagren.
- **N_USData.confirm:** Bekräftar till det övre lagret att den begärda tjänsten utförts, samt om resultat blev korrekt eller ej.

2. Tjänster för att sätta protokollparametrar

Dessa tjänster används till att kunna sätta protokollets parametrar

- **N_ChangeParameter.request:** Denna tjänst används för att sätta en specifik parameter
- **N_ChangeParameter.confirm:** Bekräftar till det övre lagret att begäran att sätta en parameter utförts, samt om det genomförts korrekt eller ej.

Interna operationer

Nätverkslagret erbjuder metoder för att hantera segmentering, flödeshantering och hopsättning av fragmenterad data. Det huvudsakliga syftet hos nätverkslagret är att överföra data som ibland kan vara fragmenterad. Meddelanden som inte får plats i ett CAN-meddelande ska segmenteras till flera delar, där varje del sänds som ett CAN-meddelande.

Applikationslagret

Den tredje delen av ISO 15765, täcker tjänster knutna till applikationslagret. Mer specifikt specificerar man diagnostiska tjänster skräddarsydda för att möta kraven hos CAN-baserade fordonsnätverkssystem. De har blivit definierade i enlighet med diagnostiska tjänster definierade i ISO 14230-3 och ISO 15031-5 (Se appendix E).

Applikationslagret tillhandahåller tjänster för att kontrollera och ta emot information CAN-nätverket. Det erbjuder även riktlinjer för hur man ska implementera ett tryggt, säkert och kraftfullt system för fordonsdiagnostik kompatibelt med de flesta tillverkares befintliga lösningar.

15765-3 specificerar vidare hur tjänster redan definierade i ISO 14230-3 ska återanvändas för diagnostik över CAN.

Tjänster till det övre lagret

Applikationslagrets tjänster kallas oftast diagnostiska tjänster. Dessa fungerar enligt klient/servermodellen där klienten oftast är en extern testutrustning och servern är en ECU hos fordonet. Testutrustningen skickar en förfrågan till ECU:n som i sin tur skickar någon form av svar. För varje tjänst som tillhandahålls av applikationslagret är följande metoder specificerade:

- **Begäran:** Används av klienten för att initiera en tjänst hos servern
- **Indikation:** Används av applikationslagret för att indikera att en intern händelse av betydelse för den diagnostiska applikationen skett samt att leverera data om den begärda tjänsten.
- **Svar:** Används av servern för att skicka svarsdata till applikationslagret
- **Bekräftelse:** Används av applikationslagret för att indikera att en intern händelse av betydelse för testapparaten skett samt leverera data till klienten.

Dessa tjänster kräver att man skickar med parametrar för att de ska utföras korrekt, några av dessa är obligatoriska, och dessa är:

- **Ursprungsadress (SA):** Anger klientens adress för att servern ska veta vars svaret skall skickas.
- **Måladress (TA):** Anger adressen till servern som klienten försöker nå.

- **Måladresstyp (TA_type):** Anger om adressen som anges är fysisk eller funktionell.
- **Resultat:** Används av svar/bekräftelse för att indikera huruvida en tjänst utförts korrekt eller ej.

Utöver de metoder som beskrivits ovan finns även en möjlighet för en klient att fjärransluta till en server som inte befinner sig på samma nätverk. För detta ändamål finns tillägg till de befintliga tjänsterna där man anger ytterligare en parameter, nämligen fjärradress (RA). Denna parameter anger serverns adress på det andra nätverket, samtidigt som DA då anger adressen till den gateway man använder för att kommunicera mellan de två nätverken.

När det gäller de faktiska tjänster som erbjuds använder sig ISO 15765-3 utav Keyword Protocol 2000 som finns definierat i ISO 14230. Alla diagnostiska tjänster som definieras i ISO 14230 är tillämpliga för diagnostik enligt ISO 15765-3. Exempel på tjänster som kan finnas är ReadfreezeFrameData som hämtar en freezeFrame från serverns minne, eller ClearDiagnosticInformation som rensar felkoderna hos servern tjänsten begärs hos.

Appendix D– ISO 9141

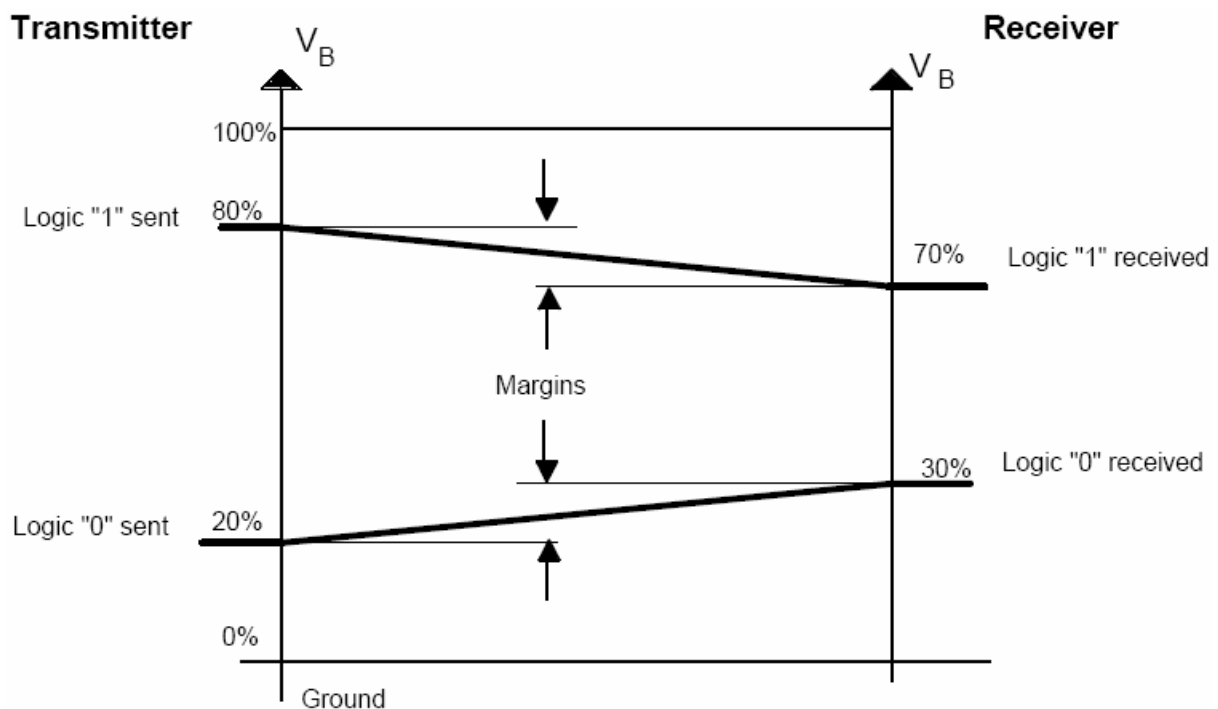
ISO 9141 är en internationell standard som togs fram 1989. Den första versionen av standarden skapades för att sätta upp krav som måste uppfyllas för utbyte av information mellan ECU:s på vägfordon och hur lämpliga diagnostiska testapparater skulle vara utformade. Kommunikationen som specificeras är främst till för att sköta inspektion, diagnos och möjligheter att göra inställningar hos fordonen.

ISO 9141 börjar med att specificera det fysiska lagret hos överföringsprotokollet. Specifikationen säger att varje ECU ska ha minst en (K) eller två (K och L) anslutningar för kommunikation. Utöver denna/dessa ska man även kunna koppla in testapparaten till batteriet (Vb) samt en signaljord (G). Om K eller L-linan från flera ECU:s är sammankopplade, har man ett s.k. bussystem.

K-linan definieras som linan där data sänds i seriell digital form från ECU till testapparaten. K kan vara bidirektionell, i vilket fall man kan skicka kommandon eller data från testapparaten till ECU:n. K-linan ska även kunna användas till att initiera kommunikationen.

L-linan å andra sidan definieras som en endast endirektionell lina från testapparaten till ECU:n. När den existerar kan den användas till att initialisera kommunikationen och/eller bära kommandon och/eller data.

Vidare definierar ISO 9141 att en logisk 0:a för seriell kommunikation är ekvivalent med en spänningsnivå på mindre än 20% av Vb hos sändaren och mindre än 30% av Vb hos mottagaren. En logisk etta är ekvivalent med en spänningsnivå på över 80% av Vb hos sändaren och över 70% av Vb hos mottagaren. Vid sändning av data tillämpar man en metod kallad NRZ vilket innebär att man inte ändrar signalnivån vid successiv sändning av två bitar av samma logiska nivå.



Tabell 7. Bild över hur spänningarna tolkas i systemet.

På grund av dåvarande ekonomiska anledningar begränsade man det maximala överföringshastigheten till 10 kbps, den minimala till 10 bps. Överföringshastigheten kan dock variera mellan dessa två värden beroende på kapacitansen hos ECU:s, kablage och testapparat. Följande samband råder:

$$BR < \frac{10^{-4}}{\sum_{i=1}^n C_{ECU_i} + C_{OBW} + C_{TE}}$$

Formel 1. Formeln för överföringshastigheten.

BR är överföringshastigheten, C_{ECU} är kapacitansen för ECU:n, C_{OBW} för kablaget och C_{TE} för testapparaten.

Vidare specificerar ISO 9141 hur kommunikation skall initieras. Detta kan ske på tre olika sätt:

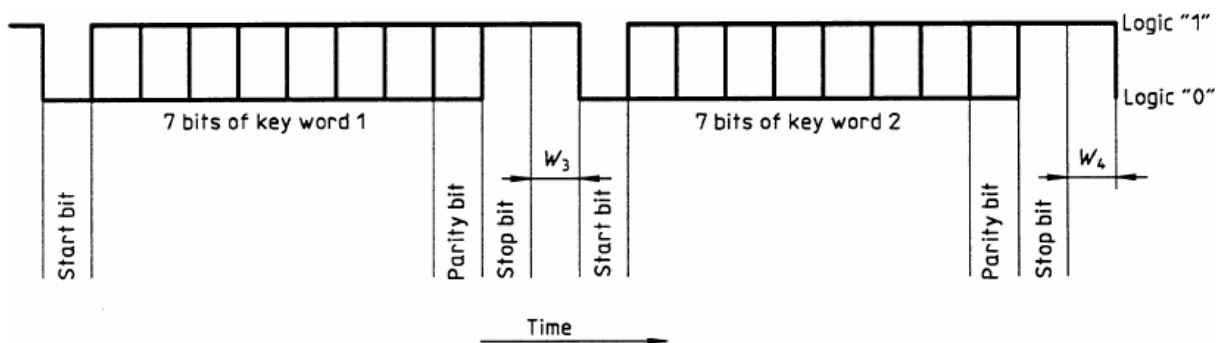
1. Genom externa medel (dvs utan användning av K eller L-linan). T.ex. att man startar bilen.
2. Genom en initieringssignal från testapparaten, vilket kan vara någon av följande:

- a. En logisk "0" under 1.8s (+/- 0,01) på både K och L-linan, detta för att skilja signalen från den maximala logiska "0" på den efterföljande adressen:
 - b. En 8 bit lång adresskod som skickas med hastigheten 5bps.
3. Genom att K och/eller L kopplas mot jord under minst 2s.

När initieringen är utförd kan kommunikationen påbörjas. För att testapparaten ska kunna kommunicera med fordonets ECU:s behöver den veta formen på kommunikationen med den initierade ECU:n. Denna information ges av den första informationen som skickas av ECU:n och denna består av:

- Ett hastighetssynkroniseringsmönster som definierar överföringshastigheten hos den efterföljande informationen.
- Minst två nyckelord som bildar en identifikationskod.

Tillverkare av fordon, system och ECU:s som behöver använda sig av nyckelord ska kontrollera samtliga nyckelord man önskar använda hos det internationella organet FAKRA som kontrollerar nyckelordet mot sin databas och sedan antingen ger sitt godkännande eller föreslår ett alternativ. Att flera tillverkare använder samma nyckelord är tillåtet om den efterföljande kommunikationen är identisk.

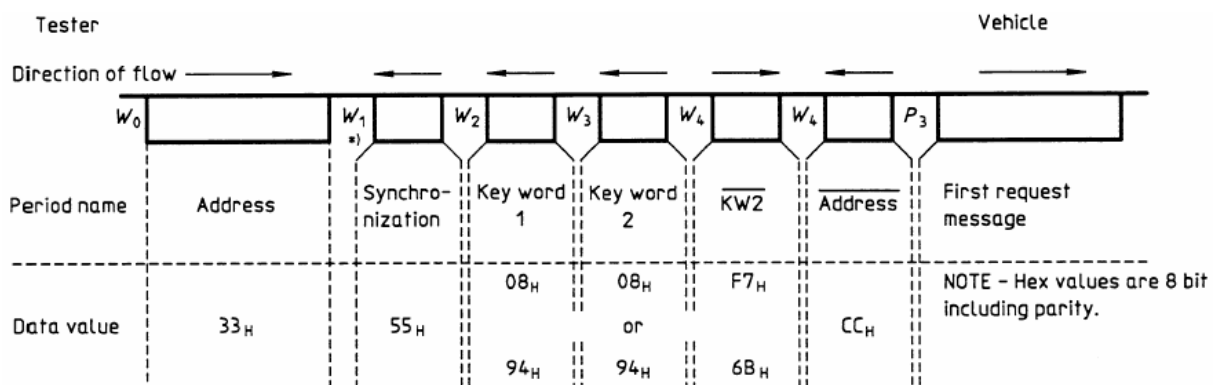


Figur 22. Strukturen hos nyckelorden.

ISO 9141 beskriver även hur kontakten för kommunikation ska vara utformat. Kontakten skall ha 16 anslutningar (se bild) och minst fyra av dessa ska vara inkopplade. Dessa fyra anslutningar är K, L, V_b samt G. Specifikationen ställer även krav på testapparaten så att den ska klara av de kommunikationssekvens som beskrevs tidigare.

1994 utökades ISO 9141 för att möta kraven hos OBD-II. Denna uppdaterade specifikation kallas ISO 9141-2. Angående initiering av data är endast alternativ två från ISO 9141 tillämpligt, med skillnaden att ISO 9141-2 definierar att initieringen ska inledas med att K-linan ska vara logisk "1" till skillnad från ISO 9141 där man istället säger att K-linan ska vara logisk "0" under en period av 1.8s. Det efterföljande initieringshuvudet är dock utformat för att vara kompatibel med ISO 9141, både då det gäller synkroniseringen samt utformningen av nyckelorden. Man har dock lagt till att testapparaten skall skicka tillbaka den logiska inversen av det andra nyckelordet till ECU:n av handskakningskäl. Man har även begränsat sig till att endast stödja 12V system, till skillnad från ISO 9141 där även 24V-system stöddes.

För att möta kraven för OBD-II är även kontakten utbytt från ISO 9141. Man använder nu istället den OBD-II kompatibla SAE J1962-kontakten som även finns specificerad i ISO 15031-3. Testapparaten måste vidare ha stöd för meddelandeprotokollet SAE J1979 och överföringshastigheten är satt till att alltid vara 10.4kbps, till skillnad från ISO 9141 där den kunde variera mellan 10 och 10kbps. Även ECU:n måste klara av kommunikation i 10.4kbps. Man väljer vilket meddelandeprotokoll man vill använda i överföringen i adresskoden som används vid initieringen. För att välja J1979-protokollet skall 33_H anges. Man definierar vidare att de två nyckelorden som skickas efter synkroniseringen väljer timingen på meddelandena. Om synkroniseringen av meddelanden beror enbart på timing skall nyckelorden sättas till 08_H 08_H, men om en annan typ av synkronisering används skall nyckelorden sättas till 94_H 94_H.



Figur 23. En schematisk bild över hur kommunikationen fungerar.

Nedan finns ett exempel på en giltig kommunikation enligt ISO 9141-2 tagen från specifikationen:

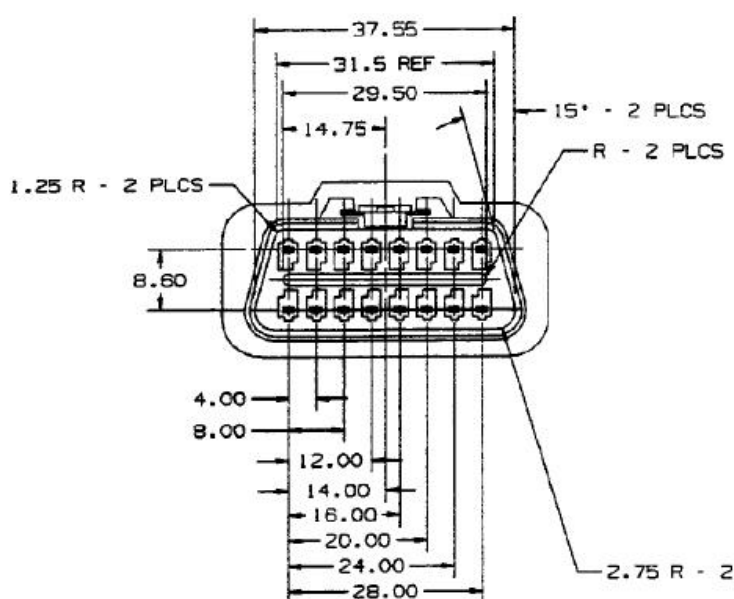
1. Testapparaten skickar 33_H med hastigheten 5 bps på både K och L-linan.
2. Både ECU 1 och ECU 2 vaknar upp, ECU 2 lyssnar medan ECU 1 förbereder sig att skicka synkroniseringsmönstret till hastigheten 10.4 kbps, testapparaten förbered sig för att ta emot.
3. ECU 1 sänder 55_H i hastigheten 10.4 kbps, testapparaten tar emot.
4. ECU 1 förbereder sig att skicka det första nyckelordet.
5. ECU 1 skickar det första nyckelordet 08_H , testapparaten tar mot nyckelordet 08_H .
6. ECU 1 förbereder sig att skicka det andra nyckelordet.
7. ECU 1 skickar det andra nyckelordet 08_H , testapparaten tar emot det andra nyckelordet och sätter P_2 min till 25 ms.
8. Testapparaten förbereder sig för att skicka svar i form av inversen av det sista nyckelordet.
9. Testapparaten skickar $F7_H$ (Den 8-bitars logiska inversen av 08_H)
10. ECU 1 förbereder sig att sända ”Klar att kommunicera” i och med mottagandet av den logiska inversen av nyckelordet.
11. ECU 1 sänder CC_H .
12. Testapparaten förbereder sig att sända sin första fråga.
13. Testapparaten sänder sitt första meddelande, en mod(1) PID(0) förfrågan som innebär en förfrågan över tillgängliga PID-värden från alla ARB-relaterade ECU:s.
14. ECU 1 och ECU 2 väntar på time-out för att vara säkra på att hela meddelandet kommit fram. ECU 1 förbereder sitt svar.
15. ECU 1 skickar ett 10 byte mod(1) svar till PID(0) frågan. Svaret säger att dessa PID:s finns tillgängliga: 1,2,5,7,8,10,13,14,16,19,21.
16. ECU 2 väntar på time-out för att vara säker på att ECU 1 svarat färdigt och förbereder sitt eget svar.
17. ECU 2 skickar ett 10 byte mod(1) svar till PID(0) frågan. Svaret säger att dessa PID:s finns tillgängliga: 1,2,4,7,9,12,13,17,20
18. Testapparaten väntar på time-out för att vara säker på den fått hela svaret.
19. Testapparaten sänder sin andra fråga, en mod(1), PID(1) fråga om diagnostisk system status.
20. ECU 1 och ECU 2 väntar på time-out för att vara säkra att hela meddelandet kommit fram.
21. ECU 1 skickar ett 8 byte mod(1) svar på PID(1) frågan. Svaret notifierar testaren om en sparad problemkod och verifierar att samtliga 8 diagnostiska test utförts och avslutats, och att ECU 1 aktiverar MIL.
22. ECU 2 väntar på time-out för att vara säker på att ECU 1 svarat färdigt och förbereder sitt eget svar.
23. ECU 2 skickar ett 8 byte mod(1) svar på PID(1) frågan. Svaret innehåller inga problemkoder, alla diagnostiska test har utförts och ECU 2 har inte aktiverat MIL.

Appendix E– Krav på OBD-II systemets kontakt

ODB2 standarden kräver en standardkontakt för kommunikationen mellan fordonet och den externa diagnosutrustningen. Kontakten kallas även diagnoskontakt och används för att ladda ner information från fordonsdatorn. Standarden kräver att likadana diagnoskontakter finns monterade i fordonen oavsett tillverkare.[19] Nedan presenteras krav som standarden ställer på diagnoskontakten.

Var kontakten i ett OBD2-system ska vara placerad beror på fordonets viktklassificering. I en personbil gäller att kontakten ska vara placerad i passagerarutrymmet i närheten av förarens del av instrumentpanelen. [19] Vanligast är att placera kontakten under instrumentpanelen, mellan ratten och centrumlinjen på fordonet. Kontakten kan sitta t.ex. bakom en lucka på instrumentpanelen eller mittkonsolen alternativt bakom askkoppen. Kontakten ska vara lätt att komma åt och inga specialverktyg ska behövas för att frilägga kontakten. [19] Kraven på placeringen av diagnoskontakten är inte lika strikta på tyngre fordon som t.ex. lastbilar, bussar och långtradare. För dessa typer av fordon är kravet att kontakten ska vara placerad under instrumentpanelen och att den går att nå antingen från förarsätet, passagerarsätet eller utifrån fordonet. Kontakten ska alltid, oavsett fordonsklass vara placerad så att den är utom synfältet för förare och passagerare.

Nedan beskrivs hur kontakten i ett fordon ska vara utformad för att uppfylla standarden.



Figur 24. OBD2 Standardkontakt.

Kontakten ska ha 16 stift och vara "D-formad" (Se bild). Den kontakt som sitter i fordonet ska vara en hona och den externa diagnosutrustningens kontakt ska vara en hane med bladstift.[19] Kontakten ska vara utformad så att den kan sättas ihop med en hand och ha tydliga egenheter som medför att det endast går att sätta ihop den på ett sätt. Vilka av kontaktens 16 stift som används beror på vilket kommunikationsprotokoll som används. Endast tre av stiften är gemensamma för de olika protokollen. Gemensamma stift är, stift 4 (chassijord), stift 5 (signaljord) och stift 16 (batterispänning +).[19] Ur tabellen (tabell över stifttilldelning) framgår att sju stift är odefinierade, dessa stift är lämnade att fritt användas av respektive fordonstillverkare.

Tillverkarna kan använda dessa till att genomföra specialfunktioner för inställning eller uppgradering av fordonsdatorn. Enda kravet som ställs är att användandet av de odefinierade stiften inte stör kommunikationen eller riskerar att skada diagnosutrustning som ej använder sig av stiften.[19] Man ska alltså kunna koppla in sig med en OBD-II standardiserad diagnosenheter som stödjer fordonets kommunikationsprotokoll och fortfarande kunna läsa ut felkoder, freeze frames samt nollställa minnet i fordonsdatorn trots att inte specialfunktionerna på de odefinierade stiften stöds av enheten.

Appendix F – Krav på extern diagnosenheter

Diagnosenheten måste enligt OBD-II standarden ha en automatisk funktion för att kunna avgöra vilket protokoll fordonet använder för kommunikation. [20] Enheten behöver inte kontrollera protokollen i någon speciell ordning men om det är möjligt kan testerna utföras samtidigt. Enheten ska endast köra följande test för att avgöra vilket protokoll som stöds.

1. Test för SAE J1850 41.6 kbps PWM (Pulse Width Modulation)
2. Test för SAE J1850 10.4 kbps VPW (Variable Pulse Width)
3. Testa snabb initiering av ISO 14230-4 (KWP 2000)
4. Testa 5 baud initiering av ISO 14230-4/ISO 9141-2
5. Test för ISO 15765-4 CAN
6. Test för SAE J1939 CAN

Om inget av testen ovan lyckas ska diagnosenheter repetera dessa och informera användaren om följande:

1. Att kommunikationsprotokollet för fordonet inte kunde avgöras
2. Kontrollera att tändningen är på
3. Kontrollera fordonsinformationen för att bekräfta att fordonet är utrustat med ett OBD-II system.
4. Kontrollera att den externa diagnosenheter är ansluten korrekt.

Den externa diagnosenheter fortsätter att testa alla protokoll tills den går igenom ett av testen eller tills det att användaren avbryter.

Diagnosenhetererna finns i en mängd varierande utföranden, från väldigt enkla handhållna enheter som bara visar information i form av text, till diagnosenheter som kopplas till en PC och använder sig av ett grafiskt gränssnitt för att presentera informationen. Enligt standarden ställs några krav på funktionaliteten hos diagnosenheter. Enheten ska vara konstruerad för att kunna ta emot, omvandla och visa felkoder samt "freeze-frames" från fordonets system. Antigen ska en beskrivande text över problemet, själva felkoden eller bägge visas i diagnosenheter. [20]

Standarden kräver även att det ska vara möjligt att radera felkoder, freezeframes och övrig information ur systemet via den externa diagnosenheten. Enheten ska kräva att användaren bekräftar en sådan information innan operationen genomförs. [20]

Sammanfattningsvis så ska diagnosenheten minst erbjuda följande funktioner:

1. Automatisk kontroll och verifiering av vilket kommunikationsprotokoll som används i fordonet.
2. Ta emot och presentera resultatet av OBD-II systemets beräkningar vid en diagnos.
3. Ta emot och presentera felkoder lagrade i OBD-II systemet
4. Ta emot och presentera realtidsdata från OBD-II systemet
5. Ta emot och presentera freezeframe data lagrade i OBD-II systemet
6. Radera felkoder, och freezeframe data som är lagrade i OBD-II systemet.

Appendix G – Förslag till design av J1939 meddelande och signal i FDB

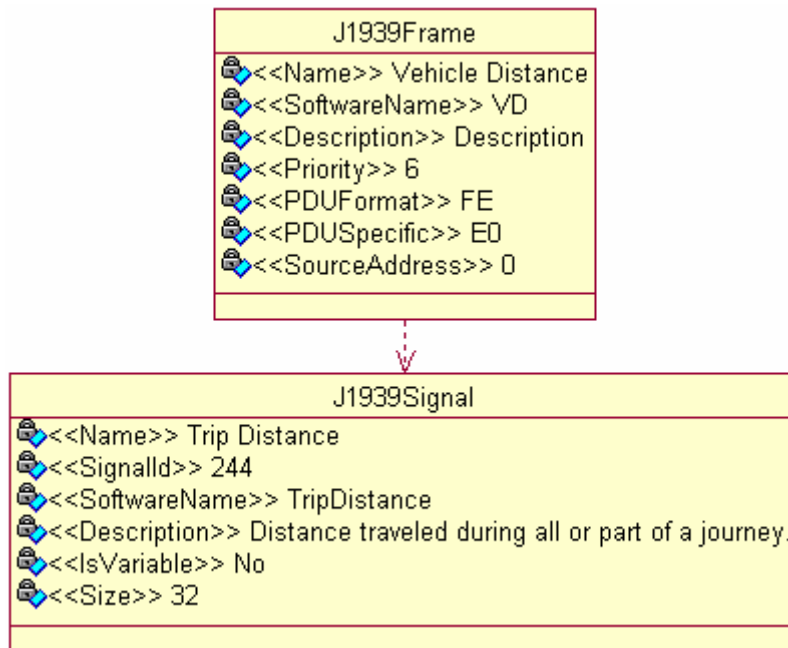
Den befintliga hanteringen av signaler och meddelanden i fordonsdatabasen baseras på att man definierar upp meddelanden var för sig och sedan kopplar man samman dessa genom att i signalen ange vilket meddelande den tillhör. Detta sätt skiljer sig från det sätt man definierar meddelanden och signaler i J1939/71. Där är det istället meddelandet som innehåller referenser till vilka signaler meddelandet ska innehålla. Rent funktionellt går det att använda båda dessa tillvägagångssätt för att definiera J1939-signaler, men rent intuitivt är det lämpligast att välja den modell som man använder i J1939-specifikationen. Detta skulle då innebära att man precis som i dagsläget definierar upp meddelanden och signaler var för sig, men istället låter meddelandet innehålla referenser till de signaler det innehåller.

Vidare innehåller signalerna i fordonsdatabasen information om sin storlek samt offset, signalens förskjutning i meddelandets datalängd. Detta är en tillräcklig lösning för att definiera upp SIL-signaler som alltid är av statisk längd, men problem uppstår när man som i fallet med J1939-signaler vill definiera signaler med variabel längd. Eftersom variabla signaler i J1939 alltid avslutas med en asterisk (*), är en lämplig lösning att lägga till ett attribut till signalen som anger ifall den är av variabel längd eller ej, om svaret på denna fråga är ja, så läses signalen in tills första asterisk hittas och slutet på signalen därmed nåts.

En annan skillnad mellan fordonsdatabasen och specifikationen, är att fordonsdatabasen innehåller en viss dubbelinformation. Detta genom att man först anger meddelandets identifierare och därefter dess avsändare och mottagare. Denna information är dock överflödigt eftersom J1939-meddelandens identifierare är uppbyggda så att de redan innehåller information som avsändare och mottagare (Se appendix B). Detta behöver inte åtgärdas för full hantering av J1939 även om det är önskvärt eftersom det tydligt visar att fordonsdatabasen ej designats med J1939 i åtanke.

Det slutgiltiga förslaget till förändring av fordonsdatabasen är alltså att meddelanden och signaler, precis som den nuvarande lösningen, definieras upp separat. Däremot innehåller signalerna inte längre någon information om vilket meddelande de tillhör eller sin position i meddelandet. Istället innehåller signalen information huruvida den är variabel eller ej, och om signalen är statisk innehåller den även sin storlek. Signalernas inbördes position i fordonsdatabasen anges sedan

istället med att man lägger till referenser till signalerna i meddelandet i den ordning de definieras i J1939-specifikationen. Vidare föreslås att fälten för mottagare och avsändare tas bort eftersom denna information redan finns tillgänglig i identifieraren. Möjligtvis kan även identifieraren delas upp i sina fyra huvudbeståndsdelar, prioritet, meddelandeformat (PF), meddelandespecifik/mottagare (PS) samt avsändare (Se appendix B).



Figur 25. Förenklat förslag på design, med exempelvärden.