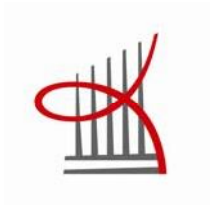


IFM-NÄYTÖN SOVITTAMINEN CAN-VÄYLÄLLÄ DIESELMOOTTORIIN CODESYS-KEHITYSYMPÄRISTÖN AVULLA

Leo Suhonen

Opinnäytetyö
Toukokuu 2011
Kone- ja tuotantotekniikka
Kone- ja laiteautomaatio
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

Tampereen ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma, Kone- ja laiteautomaatio

Tekijä	Leo Suhonen
Työn nimi	IFM-näytön sovittaminen CAN-väylällä dieselmoottoriin Codesys-kehitysympäristön avulla
Sivumäärä	39 sivua, 15 liitesivua
Valmistumisaika	5/2011
Työn ohjaaja	Risto Myllymäki
Työn teettäjä	Insinööritoimisto Comatec Oy, ohjaajana Ilkka Lanne

TIIVISTELMÄ

Uusissa moottoreissa tarvitaan jatkuvasti lisää diagnostiikkaa. Tämä mahdollistaa helpomman ja nopeamman huollettavuuden ja mahdollistaa käyttäjien diagnostiikan avulla tekemän vian paikallistamisen tai huoltotarpeen. Diagnostiikan tarvetta lisää myös uusien koneiden kiristyvät päästömääräykset ja vaatimukset. Autoissa sekä liikkuvissa työkoneissa elektroniikan ja diagnostiikan pohjana on usein käytössä CAN-väylä. Väylä välittää tietoa ja mahdollistaa eri laitteiden välisen kommunikaation.

Työssä dieselmoottorin CAN-väylään liitettiin IFM:n valmistama CR0451 mallinen perusnäyttö. Näyttö on suunniteltu juuri tällaiseen diagnostiikkakäyttöön, jossa se esittää moottorin anturien mittaamat suureet sekä ilmoittaa mahdollisista vioista järjestelmässä. Jotta näyttö ymmärtää väylällä liikkuvia viestejä se täytyy ohjelmoida kyseisen moottorin sekä käyttäjän haluamalla tavalla. Ohjelmointi tehtiin yleisesti liikkuvien työkoneiden logiikkaohjauksessa käytetyllä Codesys-kehitysympäristöllä.

CAN-väylän toiminta perustuu siitä laadittuihin standardeihin. Työssä on perehdytty väylän toiminnan perusteisiin, jotka ovat välttämättömät tietää työskennellessä väyläohjattujen laitteiden parissa. Standardissa asiat ovat esitelty hyvin perusteellisesti, mutta ovat myös hyvin raskaita tulkita. Tässä työssä asiat on pyritty esittämään selkeästi ja loogisessa järjestyksessä. Työssä on esitelty myös SAE J1939 raskaan kaluston väyläprotokollan ominaisuuksia ja perusteita.

Työ sisältää luottamuksellista tieto ja siitä syystä sitä ei voida julkaista julkiseen käyttöön kaikessa laajuudessaan.

Avainsanat CAN, SAE J1939, Codesys

TAMK University of Applied Sciences
Mechanical and Production Engineering, Machine Automation

Writer	Leo Suhonen
Thesis	Fitting IFM-display in CAN-bus to diesel engine with using Codesys-software
Pages	39 pages, 15 appendices
Graduation time	5/2011
Thesis Supervisor	Risto Myllymäki
Co-operating Company	Comatec Oy, Supervisor Ilkka Lanne

ABSTRACT

In the new generation engines more diagnostics is needed. This allows easier and faster serviceability and also enable users to locate the fault or need of maintenance with the help of diagnostics. Tighten emission requirements and standards add the need of diagnostics. In the mobile vehicles and machines base of the electronics and diagnostics is often CAN bus. Bus transfers data and enable communication between different devices.

In this final thesis IFM manufactured CR0451 basic display was connected to diesel engine via CAN bus. The display is designed for this kind of diagnostic use in which it presents the engine sensors measured parameters and will notify detected faults in the system. In order the display to understand messages from the bus, it must be programmed in the way the engine requires. Programming was done with CoDeSys development environment which is widely use in PLC programming.

CAN bus is based on established standards. The thesis presents the basic operations for bus activity, which are necessary to know when working with bus controlled devices. In standards things are thoroughly processed, but are also very heavy to interpret. In this work, cases are presented in a clear and logical order. The characteristics and criteria of the SAE J1939 protocol are also presented in this work.

The thesis contains confidential information and therefore, it can not be published for public use in all its context.

Keywords CAN, SAE J1939, Codesys

Lyhenteiden selitykset

ACK	Acknowledgement
BAM	Broadcast Announce Message
CA	Controller Application
CAN	Controller Area Network
CFC	Continuous Function Chart
CM	Connection Management
CRC	Cyclic Redundancy Check
CTS	Clear to Send
DA	Destination Address
DLC	Data Length Code
DP	Data Page
DT	Data Transfer
ECU	Electronic Control Unit
EDP	Extended Data Page
EOF	End of Frame
FBD	Function Block Diagram
GE	Group Extension
ID	Identifier
IDE	Identifier Extension Bit
IL	Instruction List
LD	Ladder Diagram
LLC	Logical Link Control
LSB	Least Significant Bit or Byte
MSB	Most Significant Bit or Byte
NA	Not Allowed
NACK	Negative Acknowledgement
P	Priority
PDU	Protocol Data Unit
PF	PDU Format
PG	Parameter Group
PGN	Parameter Group Number
PS	PDU Specific
RTR	Remote Transmission Request
RTS	Request to Send
SA	Source Address
SAE	Society of Automotive Engineers
SFC	Sequential Function Chart
SOF	Start of Frame
SPN	Suspect Parameter Number
SRR	Substitute Remote Request
ST	Structured Text
TP	Transport Protocol

SISÄLLYS

1 JOHDANTO

Työn tarkoituksena oli tutkia IFM Electronic:sin valmistamaa näyttöpäätettä dieselmoottorin diagnostiikkakäytössä. Näyttö liitetään moottoriin CAN-väylällä. Jotta näyttö osaisi keskustella väylän kanssa se ohjelmoitiin käyttäen Codesys-ohjelmistoa. Näytöltä voidaan tarkastella moottorilta mitattavia suureita sekä moottorissa mahdollisesti esiintyviä häiriö- ja vikatiloja. Näyttöön oli myös tarkoituksena ohjelmoida moottorin ohjaussovellus, jolla moottorin kierroslukua ja vääntömomenttia voidaan säätää näyttöpäätteen kautta.

Työn voi jakaa karkeasti kahteen eri osaan. CAN-väylätekniikan teoreettinen tuntemus ja näytön ohjelmoiminen Codesys-ohjelmalla. Teoreettinen perustietämys asiasta on välttämätöntä ennen varsinaiseen suorittamiseen siirtymistä. Tavoitteena oli toimivan sovelluksen luominen näyttöpäätteeseen sekä uuden tiedon omaksuminen työn kohteen alueesta.

2 CAN-VÄYLÄTEKNIIKKA

CAN-väylällä voidaan yhdistää lukuisia eri älyä sisältäviä laitteita. Väylään voidaan liittää myös antureita ja muita tiedonkeruulaitteita. CAN-väylässä ei ole yhtä pääkeskusta joka komentaisi muita tekemään haluttuja asioita, vaan useat komponentit jakavat tietoa väylää pitkin ja tekevät päätöksiä muiden lähettämiin tietoihin perustuen. Väylällä toimivia laitteita kutsutaan solmuiksi (node). Kun solmu jakaa tietoa muiden kanssa, se ei osoita lähetystä erityisesti toiselle solmulle, vaan viestittää tiedon eetteriin, kuten radiolähetin tekee. Väylän kaikki laitteet kuulevat tiedon ja päättävät tarvitsevatko ne kyseistä tietoa vai ei. Näin kaikki laitteet ovat selvillä mitä järjestelmässä tapahtuu ja pystyvät toimimaan kokonaisuutena. CAN-väylä on suunniteltu nopeaan tiedonsiirtoon ja siinä liikkuvat tietomäärät ovat verrattaen pieniä. Jos järjestelmässä on tarkoitus liikutella suuria tietomääriä, on syytä harkita jotain muuta kuin CAN-väylää. (CAN in Automation 2011)

2.1 Historia ja CiA

CAN-väylä (Controler Area Network) on alun perin suunniteltu autoteollisuuden käyttöön, mutta nykyään se on laajasti käytössä lähes kaikilla teollisuuden osa-alueilla. Bosch aloitti CAN-väylän suunnitteluprojektin 80-luvun alkupuolella ja se esiteltiin yleisölle 1986 SAE-tapahtumassa (Society of Automotive Engineers) Robert Bosch GmbH toimesta. Alkuperäisessä suunnittelussa tehtiin ilmeisen hyvää työtä, koska protokolla on säilynyt muuttumattomana CAN-väylän 25-vuotisen historian ajan. Muita tärkeitä asioita CAN-väylän kehityksen kannalta on ollut CiA (CAN in Automation) perustaminen vuonna 1992 ja CAN-väylän stardartointi, ISO 11898 mukaan, vuonna 1993. (CAN in Automation 2011)

CiA on kansainvälinen käyttäjien ja valmistajien voittoa tavoittelematon järjestö. Se perustettiin muutaman yhtiön toimesta tarkoituksenaan tehdä CAN-väylä tunnetuksi maailman teollisuudessa ja lisäämään sen käyttöä. Nykyään noin 540 yritystä ovat CiA jäseniä. CiA järjestää jatkuvasti koulutuksia ja seminaareja lisätäkseen CAN-väylän tunnettavuutta ja sen osaajien määrää. CiA tarjoaa tietoa teknistä, tuote- ja markkinointi tietoa CAN-väylästä. (CAN in Automation 2011)

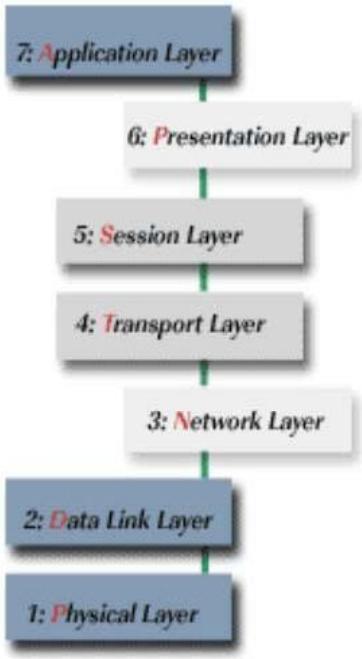
CAN-väylä on hyvin joustava järjestelmä, joten sillä on paikkansa monissa eri sovelluskohteissa. Järjestelmä voidaan toteuttaa halvalla tai kalliilla kohteen vaativuudesta riippuen. Järjestelmän kaapelointi on hyvin yksinkertainen: yksi parikaapeli joka kiertää solmujen kautta. Järjestelmän hintaan voidaan vaikuttaa kaapeloinnin laadusta tinkimällä. Yksinkertainen kaapelointi on suuri etu konetta suunnitellessa ja järjestelmän toimilaitteet voidaan sijoittaa helposti eri puolille konetta. Järjestelmä on modulaarinen eli solmuja voidaan lisätä ja poistaa ilman että se häiritsee muiden solmujen toimintaa. CAN-väylään on olemassa valmiiksi määriteltyjä fyysisen kerroksen standardeja sekä korkeamman kerroksen sovellusprotokollia joista varmasti löytyy valmiiksi pureskeltu ratkaisu käyttökohteeseen. (Wilfried 2005 CAN, 1-3)

2.2 Yleiset ominaisuudet

CAN-väylä on usean isännän väylä (multi master bus), joka tarkoittaa että kaikki laitteet voivat lähettää viestejä väylälle. Viestejä ei ole osoitettu kenellekään vaan niihin on merkitty mitä tietoa ne sisältävät. Periaatteessa solmut lähettävät viestejä väylälle vapaassa järjestyksessä, mutta viesteihin on sisällytetty myös niiden tärkeysaste. Jos jokin solmu lähettää tärkeämpää viestiä kuin muut, niin solmut hiljenevät kuuntelemaan ja antavat tilaa väylälle tärkeämpiä viestejä varten. Periaatteessa solmuja voi olla rajaton määrä, mutta yleensä niiden määrästä on annettu ohjearvo tai rajoitus. (Wilfried 2005 CAN, 13)

Viestit liikkuvat kehyksissä, joissa viestin pituus on rajoitettu 8 tavuun. Kehykseen on sisällytetty lukuisia hyödyllisiä ominaisuuksia kuten kilpavaraus, virheentarkistus sekä tiedon paketin sisällöstä. Nopeimmillaan väylä pystyy 1Mbit/s nopeuteen 40m kaapelipituudella, mutta nopeudesta tinkimällä päästään jopa 5km väylänpituuteen. (CAN in Automation 2011)

Controller Area Network (CAN) protokolla määrittelee fyysisen kerroksen ja tiedonsiirto kerroksen OSI-mallissa. CAN-väylä standardi ISO 11989 määrittää CAN-protokollaa mukaillen säädökset näistä kahdesta alimmaisesta kerroksesta. (CAN in Automation 2011)



Kuvio 1: OSI-malli (CAN in Automation 2011)

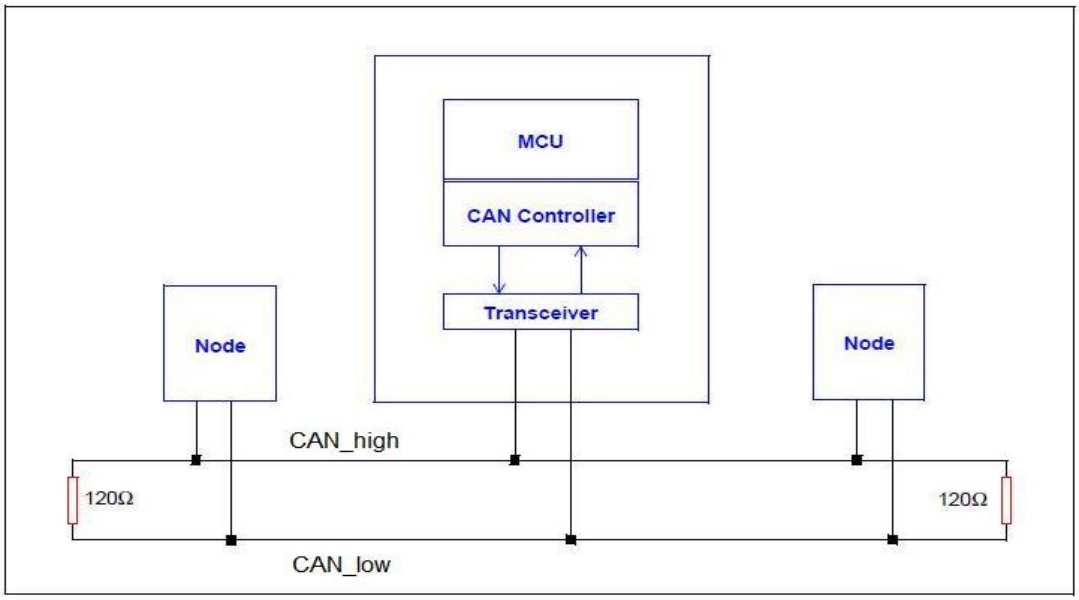
2.3 Fyysiset ominaisuudet

Väylän fyysinen muoto on kierretty parikaapeli. Suuremmilla tiedonsiirtonopeuksilla kaapelin tulee olla laadukas ja suojattu. Liittimien laatuun tulee kiinnittää erityistä huomiota, koska ne voivat aiheuttaa kapasitanssia ja resistanssia johtimiin. Ne loiventavat signaalin muotoa ja siten häiritsevät sitä. Väylä on topologialtaan bus-tyyppinen eli kaapeli kiertää kaikkien päätelaitteiden kautta ja se päätetään molemmista päistä 120 ohmin päätevastuksilla. Suositeltava johdotustapa yksi pitkä suora johdin johon solmut liitetään T-liitännällä lyhyellä johtimella. Päätevastuksien tehtävä estää signaalin heijastuminen takasin. Signaalin takaisin heijastuminen on yleinen kaikilla korkeataajuisilla signaaleilla. Tyhjä johdonpää toimii kuten peili, joka heijastaa takaisin. Päätevastukset on tärkeä olla, ettei heijastus pääse häiritsemään väylän liikennettä. (CAN in Automation 2011)

Kaapelin enimmäismitta eli väylän fyysinen pituus määräytyy siirtonopeuden vaatimusten mukaisesti. Broadcast-tyyppisessä lähetyksessä kaikki solmut ottavat kilpavaraus-periaatteen mukaisesti näytteen viestistä. Jotta ongelmilta vältytään, täytyy kaikkien solmujen ottaa näyte yhtä aikaa tietyllä toleranssilla. Kun väylän pituutta lisätään, viive lisääntyy ja siirtonopeus laskee. Tämä johtuu signaalin hitaudesta eli sähkömagneettisen aallon kulkunopeudesta. (CAN in Automation 2011)

2.3.1 CAN-solmun rakenne

Solmu koostuu kolmesta eri komponentista: Lähetin / vastaanotin, CAN-ohjain ja mikrokontrolleri. Solmun osat voivat olla erillisiä tai yhteen integroituja. Monet automaatiokomponenttien valmistajat tarjoavat integroituja pakettiratkaisuja. Niiden etuna on varmempi toimivuus, helppous ja nopeus, varjopuolena näissä on luonnollisesti hinta. Valmiista ratkaisuista joutuu yleensä maksamaan enemmän. (Alanen 2000, 11)

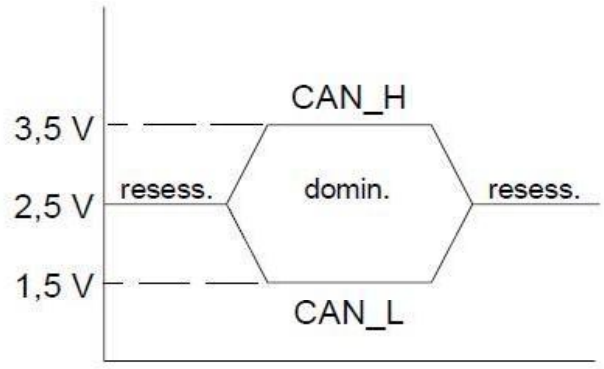


Kuvio 2: Väylä topologia ja solmun rakenne (Wilfried 2005 CAN, 133)

2.3.2 Tiedonsiirto

Tiedonsiirto väylässä kuuluu fyysiseen kerrokseen. Käytetyin fyysinen kerros on High speed CAN, joka on määritelty ISO 11898-2 standardissa. Tiedonsiirtäminen väylässä perustuu CAN_high ja CAN_low väliseen jännite-eroon. Dominantilla tarkoitetaan yleensä että bitti on 1. Resessiivisellä tilasta puhuttaessa tarkoitetaan yleensä 0, mutta jos nähdään jokin erityinen syy, on väylä mahdollista muuttaa ohjelmoimalla päinvastaiseksi. Resessiivisessä tilassa molempien johtimien jännite on 2.5V. Dominantissa CAN_high jännite arvo on 3.5V ja CAN_low arvoksi tulee 1.5V. Jännite ero on siis 2V. (CAN in Automation 2011)

Vaikka johtimeen pääsisi vaikuttamaan sähkömagneettinen säteily, se ei välttämättä pysäytä tiedonsiirtoa väylällä. Sähkömagneettisen säteilyn aiheuttama jännitteen heittely vaikuttaa molempiin johtimiin samalla lailla, joten niiden välinen jännite-ero pysyy ennallaan. Tämä on merkittävä etu tämän tyyppisellä tiedonsiirtokaapeloinnilla. (CAN in Automation 2011)



Kuvio 3: Jännite-ero tiedonsiirrossa (Wilfried 2005 CAN, 135)

Edellä esitellyn fyysisen kerroksen high speed CAN:in lisäksi on olemassa muita vähemmän käytettyjä vaihtoehtoja.

Low speed CAN on nimensä mukaisesti tarkoitettu vaatimattomampaan tiedonsiirtoon, eikä se koskaan ole saavuttanut suosiota sen käytöstä ollaankin lähes luovuttu. Low speed CAN on määritelty ISO 11519-2 standardissa. (CAN in Automation 2011)

Fault-tolerant CAN on käytössä lähinnä autoteollisuuden korielektroniikassa. Se on määritelty ISO 11898-3 standardissa ja on suunniteltu käytettäväksi lähinnä lyhyillä etäisyyksillä. Koska väylä on niin lyhyt, aikaisemmin mainittu signaalin takaisinheijastumisongelma ei ole niin suuri. Tällöin ei tarvitse käyttää lineaarista topologiaa vaan epäsymmetriset ratkaisut ovat mahdollisia, koska väylää ei välttämättä tarvitse päättää päätevastuksella. Tämän protokollan hyviin puoliin kuuluu myös parempi vikasieto. Jos yksi väylän haara putoaa pelistä pois vaikka sähkövian takia, kommunikaatio muiden haarojen välillä voi jatkua. Tehon kulutus on myös hyvin alhainen ja maksiminopeus on rajoitettu 125kbit/s. (CAN in Automation 2011)

Single wire CAN toimii yhdellä kaapelilla ja vaatimattomalla 33kbit/s nopeudella. Koska tässä järjestelmässä on vain yksi johdin, tiedonsiirto tapahtuu jännitetasoa mittaamalla jännite-eron sijaan. Se on suunniteltu moottoroituihin ajoneuvoihin ja elektroniikkajärjestelmiin. Standardi SAE J2411 määrittelee tätä vähän käytettyä ratkaisua. (CAN in Automation 2011)

On olemassa muitakin sovelluksia esimerkiksi optista kuitua tiedonsiirtoon käyttävä väylä, mutta toistaiseksi niiden suosio on pientä eikä niille ole laadittu omaa standardia.

2.3.3 Liittimet

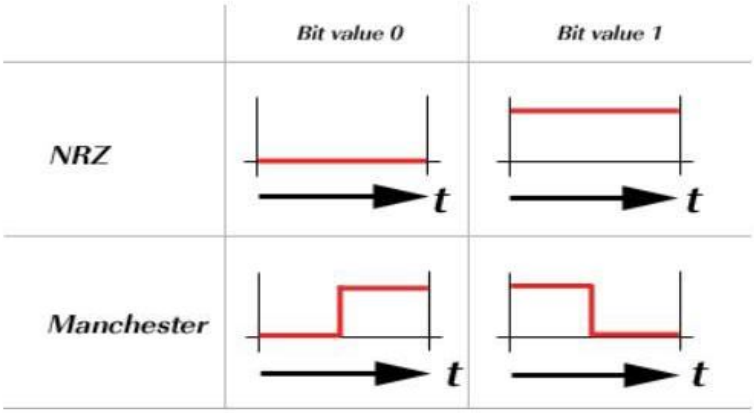
CiA standardi liitin CAN-väylässä on 9-napainen D-liitin. Haluttaessa voidaan käyttää myös muita liittimiä. Liitin on usein valmistajakohtainen. Projektissa käytetyn IFM:n CR0451 näytön takaa löytyy 5-napainen M12 liitin, joka suojattu ympäristön vaikutuksilta hieman paremmin kuin perus D-liitin. (CAN in Automation 2011)

Illustration	Pin	Designation	Note
	1	n.c.	----
	2	VBB	8...32 V DC
	3	GND	terminal 31
	4	CAN_H	
	5	CAN_L	

Kuvio 4: A koodattu M12 CAN liitin (System manual BasicDisplay 2011, 11).

2.3.4 NZR

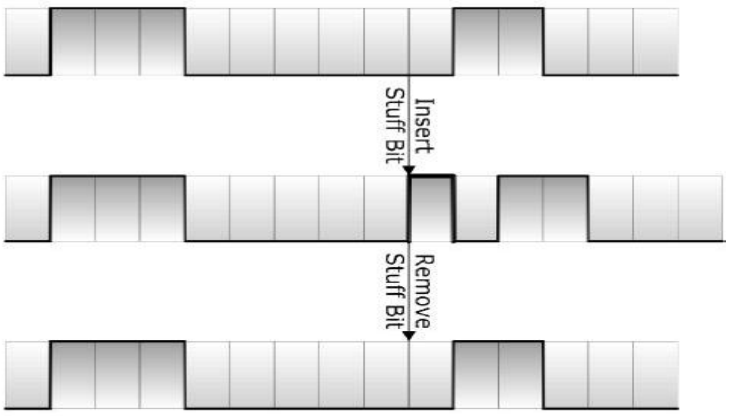
Tieto liikkuu väylällä jännitesignaalina. Se täytyy muuttaa biteiksi, jotta koneet ymmärtävät sitä. Tähän käytetään Non return to zero (NRZ) koodausta. Tässä koodaus menetelmässä dominantissa tilassa bitti on 1 ja resessiivisessä 0. Vertailun vuoksi Manchester koodauksessa signaalin nouseva reuna on 0 ja laskeva 1. (CAN in Automation 2011)



Kuvio 5: NZR ja Manchester koodauksen vertailu (CAN in Automation 2011).

2.3.5 Bit stuffing

Bitit erotetaan toisistaan ajastuksella. Yksi bitti kestää tietyn hetken. Koska NRZ koodauksessa ei ole neutraalia tilaa sen kanssa on käytettävä bit stuffing toimintoa. Se tarkoittaa, että kun koodissa esiintyy viisi samaa peräkkäistä merkkiä, lähetin syöttää automaattisesti väliin yhden stuff bitin. Viestiä vastaanotettaessa vastaanotin poistaa ylimääräiset stuff bitit koodista. Tämä menetelmä helpottaa solmujen välistä synkronointia ja vähentää virheiden mahdollisuutta väylän liikenteessä. Alla oleva kuvio 5 esittää bit stuffing tapahtuman. Ylin koodirivi on lähetettävä koodi, johon lähettäjä laittaa stuff bitin. Keskimmäinen koodirivi on väylällä liikkuva koodi, josta vastaanottaja poistaa stuff bitin jolloin viesti on taas alkuperäisessä muodossaan. (Wilfried 2005 CAN, 96)



Kuvio 6: Bit stuffing tapahtuma (Wilfried 2005 CAN, 98).

Koska bit stuffing toiminto lisää viestipaketissa olevan bittien määrää se hidastaa väylän nopeutta. On kuitenkin todettu parantunut lähetysvarmuus on niin hyvä etu, että nopeudesta voidaan hieman tinkiä. Seuraava taulukko esittää paljonko stuff bittejä yhdessä paketissa voi olla. Taulukko perustuu normaaliin 11 bittisen tunnistekenttäiseen viestiin. Keskimääräinen bit stuffing on määritetty matemaattisesti. (Wilfried 2005 CAN, 100)

Taulukko 1: Viestin pituus bit stuffingista johtuen

	Viestin sisältämä data 0 bittiä	Täysi viesti eli 8 tavua dataa
Ei bit stuffingia	47 bit	111 bit
Maksimaalinen bit stuffing, eli pahin mahdollinen tilanne	55 bit	135 bit
Keskimääräinen bit stuffing	49 bit	114 bit

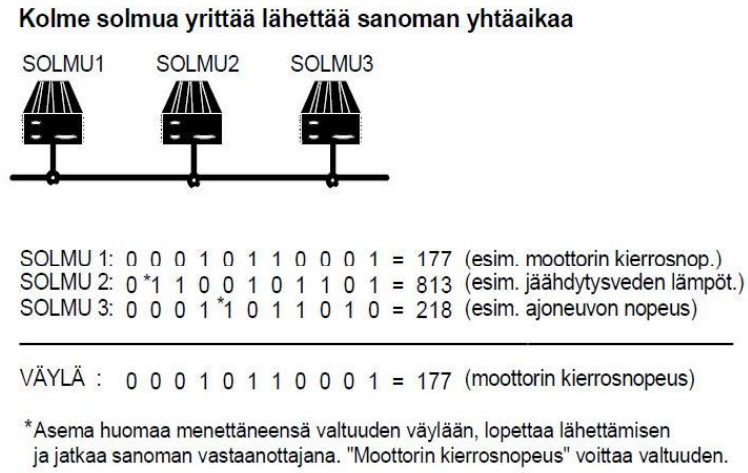
2.4 Siirtoyhteyskerros

Kaikki edellä mainitut asiat kuuluvat fyysiseen kerrokseen. Seuraavat kappaleet käsittelevät kuviossa 1 esitetyn OSI mallin seuraavaa kerrosta, siirtoyhteyskerrosta. Siihen kuuluu viestien priorisointi, niiden kehysrakenne ja virheentarkistus. (CAN in Automation 2011)

2.4.1 Kilpavaraus

Solmut lähettävät viestejä Broadcast-tyyppisesti kaikille väylään liitetyille laitteille. Viestistä löytyvän tunnistenumeron perusteella solmut päättävät tarvitsevatko ne kyseistä tietoa. Tunnistenumero kertoo myös viestin kiireellisyydestä. Mitä pienempi binääriluku sitä tärkeämpi viesti on. Kun useampi solmu lähettää viestiä yhtäaikaisesti väylälle pienimmällä tunnistenumerolla varustettu viesti voittaa kilpavaraus tilanteen.

Kilpavarauksen hävinneet solmut lopettavat lähettämisen ja hiljenevät kuuntelemaan väylää. Alla oleva kuvio 6 selittää hyvin kilpavaraus tilanteen. (Wilfried 2005 CAN, 88-89)



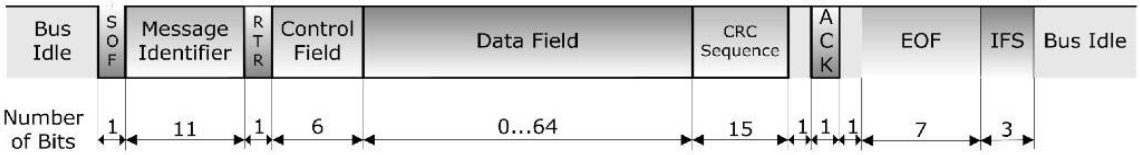
Kuvio 7: Kilpavaraustilanne (Alanen 2000, 7)

2.4.2 Kehysrakenne

Viestit liikkuvat väylällä viesti kehyksissä CAN-prokollassa on määritelty 4 erilaista kehystä. Tilanne voi olla hieman eri käytettäessä ylemmänkerroksen sovellustason protokollaa esimerkiksi SAE J1939, josta lisää myöhemmin. Viestistä on olemassa myös extended malli, jossa tunnistekenttä on pidempi. Myös tähän pureudutaan paremmin SAE J1939 kappaleen alla.

2.4.2.1 Sanomakehys

Sanomakehys tai datakehys kuljettaa viestin väylää pitkin kaikkien solmujen kuultavaksi. Kehyksen aloittaa SOF (start of frame). Se on yksi dominantti bitti se ilmoittaa viestin alkamisesta ja sitä käytetään solmujen keskinäiseen synkronointiin. (Wilfried 2005 CAN, 14)



Kuvio 8: Sanomakehyksen rakenne (Wilfried 2005 CAN, 46)

Message identifier-kenttä sisältää tiedon mitä viesti pitää sisällään. Se määrittää myös viestin tärkeellisyysasteen ja sitä käytetään kilpavaraustilanteessa.

RTR kertoo onko kyseessä sanomakehys(=dominantti) vai kyselykehys(=resessiivinen).

Control field kentän ensimmäinen bitti ilmoittaa onko kyseessä standardimuotoinen vai laajennettu kehys. Toinen kentän bitti on varattu tulevaisuuden mahdollisille lisäkeksinnöille ja se tulee asettaa dominantiksi. Kentän loput neljä bittiä kertovat paljonko tietoa kyseinen kehys sisältää.

Data Field kenttä on itse tieto joka halutaan välittää muille väylällä.

CRC sequence eli tarkistus summa.

ACK on kuittaus kenttä jonka lähettäjä kirjoittaa resessiiviseksi ja sanoman vastaanottajat kirjoittavat sen dominantiksi ilmoittaakseen oikein saapuneesta viestistä. Sen molemmin puolin on yksi bitti, jotka kirjoitetaan resessiivisiksi aina.

EOF (end of frame) on seitsemän resessiivistä bittiä jotka ilmoittavat viestin loppumisesta.

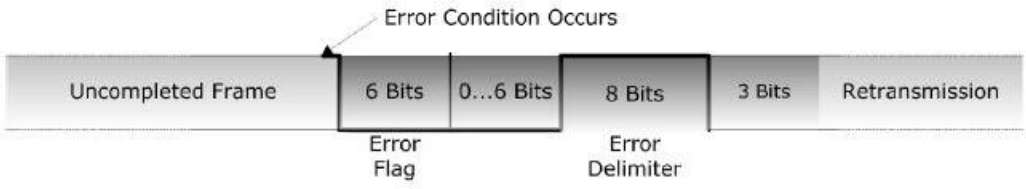
IFS kenttä on kolme resessiivistä bittiä. Ne ovat viestien välinen odotusaika ja niiden tehtävä on rauhoittaa väylän liikennettä.
(CAN in Automation 2011)

2.4.2.2 Kyselykehys

Kyselykehys on rakenteeltaan samanlainen kuin viestikehys. Eroina on että se ei sisällä mitään informaatiota eli datakenttä puuttuu kyselykehyksestä ja RTR kentässä on resessiivinen bitti ilmoittamassa kyseessä olevan kyselykehys. Esimerkkinä jos jonkin solmu tarvitsee tietoa moottorin kierrosnopeudesta se lähettää siitä kyselyn väylälle. Kaikki solmut saavat tämän tiedon ja kierroslukuanturi ymmärtää viestin sille tarkoitetuksi, joten se lähettää sanomakehyksen joka sisältää kierrosluvun arvon takaisin väylälle. (Wilfried 2005 CAN, 60)

2.4.2.3 Virhekehys

Virheellisen viestin osuessa väylälle kaikki virheen havaitsemat solmut keskeyttävät viestin vastaanottamisen ja lähettävät väylälle virhekehyksen, joka keskeyttää virheellisen viestin lähettämisen. Virhekehyksessä on 6 peräkkäistä dominanttia bittiä ja 8 peräkkäistä resessiivistä bittiä. Tämä rikkoo bit stuffing sääntöä, joten solmut ymmärtävät heti kyseessä olevan virhekehysviesti. Tästä solmut viimeistään tajuavat lähetyksessä olleen viestin olleen virheellinen ja hylkäävät sen. Virheellisen viestin lähettänyt solmu yrittää uudestaan lähettää saman viestin. Jos virhe toistuu seuraavissa viesteissä riittävän monta kertaa peräkkäin sanomaa lähettävä solmu luovuttaa ja sulkee itsensä pois väylältä, jotta se ei häiritsisi enempää väylän toimintaa. (Wilfried 2005 CAN, 64)



Kuvio 9: Virhekehys väylällä (Wilfried 2005 CAN, 66)

2.4.2.4 Ylikuormituskehys

Ylikuormituskehys tai viivekehys on tarkoitettu hitaille solmuille tilanteessa jossa solmu tarvitsee lisää aikaa vastaanotetun sanoman käsittelyyn. Käytännössä uusissa CAN järjestelmissä tälle ei ole tarvetta vaan kaikki solmut suoriutuvat yksittäisistä viesteistä ilman ongelmia. Jos viivekehystä tarvitsee käyttää, se lähetetään sanomien välisenä aikana eli viestikehyksessä mainitussa INT kentässä. Viivekehys on muodoltaan samankaltainen virhekehyksen kanssa. (Wilfried 2005 CAN, 67)

2.4.3 Virheentarkistus

CAN-protokolla on melko varmatoiminen, koska siinä on useita virheentarkistustoimintoja. Niillä varmistetaan että viestit siirtyvät väylällä muuttumattomana. (CAN in Automation 2011)

Solmut tarkkailevat väylällä liikkuvien kehysten rakennetta ja niiden pituutta. Kehyksessä havaittua virhettä kutsutaan nimellä format error. (CAN in Automation 2011)

Viestikehyksestä löytyvä CRC (Cyclic Redundancy Check) tarkistussummaa käytetään virheiden havaitsemiseen. Solmu suorittaa CRC luvulle laskutoimituksen ja vertaa sitä viestin mukana tulleeseen avaimeen.. Jos summa ei täsmää on tapahtunut virhe. (CAN in Automation 2011)

Lisäksi solmut tarkkailevat jatkuvasti bit stuffing säännön mukaisesti ettei koodissa ole peräkkäin yli viittä samanmuotoista bittiä. (CAN in Automation 2011)

Virheettömästi vastaan otetusta viestistä ilmoitetaan kuittauksella lähettäjä solmulle viestikehyksestä löytyvällä ACK kuittaus bitillä. Näin solmu tietää että ainakin yksi on vastaanottanut viestin virheettömästi. (CAN in Automation 2011)

Kun solmu huomaa lähettäneensä virheellisen viestin, se katkaisee lähetyksen ja yrittää uudestaan. Usein virheet ovat ohimeneviä häiriöitä, mutta jos virhe toistuu se lisää virhelaskurin summaa. Solmuilla on virhelaskuri lähetysvirheille ja vastaanottovirheille.

Jatkuva virheellisten viestien lähettäminen tukkii väylää sen takia kun virhelaskuri saavuttaa arvon 127 solmu siirtyy passiiviseen virhetilaan. Passiivisessa virhetilassa oleva solmu yrittää vielä kuunnella väylällä liikkuvia viestejä sekä lähettää virhekehyksessä vain resessiivisiä bittejä ja näin ei häiritse muun väylän liikennettä. Tilanteessa jossa virhelaskuri saavuttaa arvon 255, solmu siirtyy kokonaan pois väylältä. Se voidaan palauttaa toimintaan manuaalisesti resetoimalla solmu. Virhelaskurien arvot pienenevät automaattisesti onnistuneista lähetyksistä, kun väylän liikenne palaa normaaliksi. (Alanen 2000, 8-9)

2.5 Ylemmän kerroksen arkkitehtuurit ja SAE J1939

Sovelluskerros on kuviossa 1 esitetyn OSI mallin ylin kerros ja se määrittelee tarvittavat väliin jääneet kerrokset. Sovelluskerrokseen on olemassa useita vaihtoehtoja. Niistä käytetyimpiä on CANopen, jota käytetään kaikkialla sulautetuissa järjestelmissä. Se on erittäin joustava ja siinä on monipuoliset ohjelmointimahdollisuudet. Toinen paljon käytetty ylempi protokolla kerros on SAE J1939.

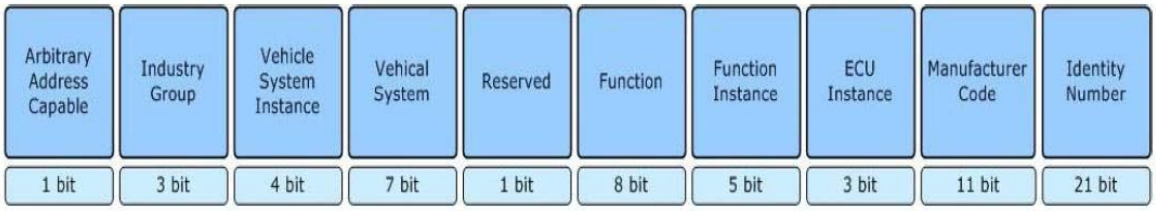
SAE J1939 on suunniteltu käytettäväksi raskaissa työkoneissa kuten kuorma-autoissa ynnä muissa dieselmoottorikäyttöisissä liikkuvissa työkoneissa. SAE (Society of Automotive Engineers) julkaisi standardin vuonna 1998. Se on suunniteltu varmatoimiseksi, koska alun perin pääasiallisia sovelluskohteita olivat moottorin, vaihteiston ja jarrujen viestintä. Myöhemmin viestintäkohteita ja valikoimaa on laajennettu, mutta pää idea eli mahdollisimman suuri toimivuusluotettavuus ja siihen liittyvä vikojen tunnistus ja hallinta on edelleen sama. J1939 pohjalta on tehty sovellukseen sopivaksi muokattuja standardeja kuten: ISO 11783 (ns. ISOBUS) – maatalouskoneet, MilCAN – Sotilas käyttö ja NMEA 2000 – Merenkulun sovellukset. (CAN in Automation 2011)

Standardi määrittelee osittain väylän fyysisiä ominaisuuksia. Maksimi väylän pituus on 40m ja siinä on käytettävä häiriösuojattua kierrettyä parikaapelia. Solmujen määrä väylällä on rajoitettu 30 kappaleeseen. Normaali siirtonopeus väylällä on 250kBit/s. Kaikilla näillä rajoituksilla pyritään väylän maksimaaliseen luotettavuuteen.

SAE J1939 käyttää extended-viestikehystä, jossa viestin tunnistekenttä on 29 bittinen. Standardi ei tue solmujen tarkkailu ominaisuutta, joten väylälle voi liittää tai poistaa laitteita lennosta. Myös isompien kuin 8 tavuisten tietopakettien lähettäminen on mahdollista multi-packet toiminnolla. (Wilfried 2005 J1939, 1-3)

2.5.1 Address claiming

Kun järjestelmä käynnistetään se suorittaa address claiming toiminnon jossa jokainen väylällä oleva solmu saa yksilöllisen tunnisteen ja nimen. Näin voidaan lähettää viestejä osoitettuna tietylle solmulle broadcast lähetyksen sijaan. Jokaisen viestin ID kentän lopusta löytyy kuka viestin on lähettänyt, kohdistetuissa sekä broadcast lähetyksissä. Kun väylälle liitetään solmu myöhemmin myös se suorittaa address claiming toiminnon. Laitekohtaisilla viesteillä solmut pystyvät myös kyselemään toisiltaan vieläkö ne ovat väylällä. Tosin edellä mainittu kysely ei ole automaattisesti standardissa määritelty vaan se on jätetty sovellus tasolle ohjelmoijan päätettäväksi halutaanko sitä tehdä. Alla olevassa kuviossa on esitetty mistä solmujen nimet muodostuvat. (Wilfried 2005 J1939, 82-84)



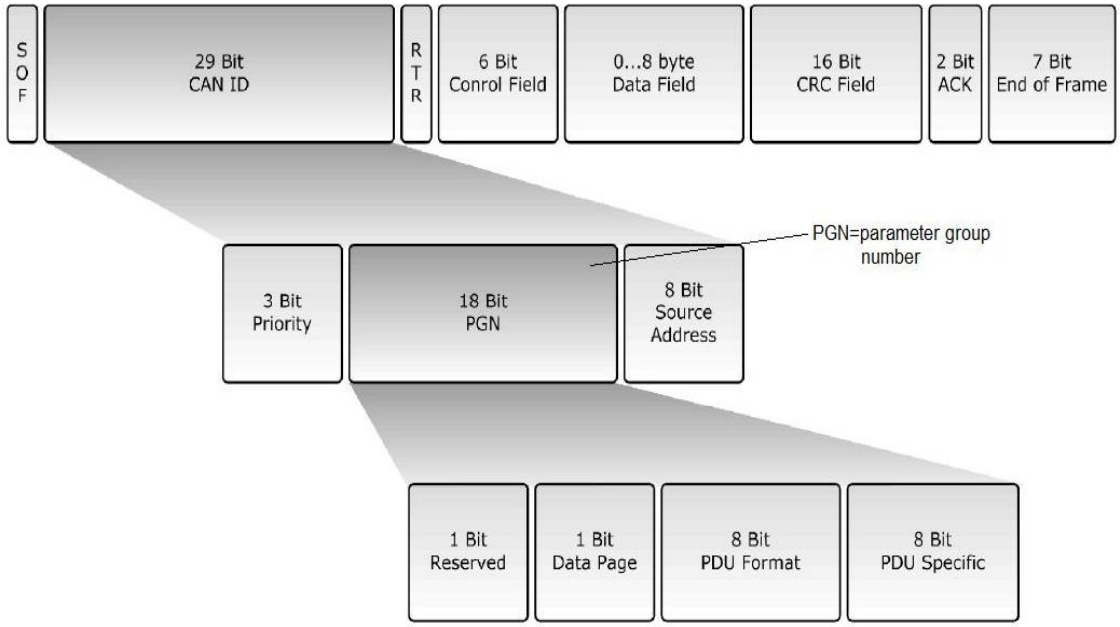
Kuvio 10: Solmun nimi ja osoite (Wilfried 2005 J1939, 83).

Järjestelmän käynnistyessä solmu lähettää väylälle address claim viestin, jossa se vaatii nimeä ja osoitetta. Vastauksen saatuaan, solmu tarkistaa nimensä vertaamalla sitä sisäiseen nimikylttiinsä. Ristiriita voi syntyä jos kaksi solmua vaatii samaa osoitetta. Tällöin pienemmällä nimellä varustettu voittaa. Hävinneen solmun täytyy vaatia uutta osoitetta, kunnes saa sen tai sulkea itsensä pois väylältä. (Wilfried 2005 J1939, 82-84)

2.5.2 Extended CAN viestikehys

Datakehykset SAE J1939 ovat CAN standardin laajennettua muotoa. Ne ovat muilta osin samanlaisia kuin normaali sanomakehys, mutta kehyksen ID kenttä on 29 bittiä pitkä normaalin 11 bitin sijaan. (Wilfried 2005 J1939, 40-42)

Alla olevassa kuvio 10:ssä on selvitetty ID kentän rakennetta. Kolme ensimmäistä bittiä kertoo viestin tärkeydestä väylälle. Viesteille on sovittu tärkeys luokitus nollasta seitsemään. Oletusarvona kaikille hallintaviesteille on 3. Mitä pienempi arvo sitä tärkeämpi on viestin luokitus, 000 ollessa tärkein mahdollinen. (Wilfried 2005 J1939, 40-42)



Kuvio 11: Extended viestikehys (Wilfried 2005 J1939, 41).

PGN osuuden 2 ensimmäistä bittiä on varattu tulevaisuuden käyttöön ja tulee aina asettaa arvoon nolla. Jos PDU format kentän arvo on väliltä 0 ja 239 viesti on solmulta solmulle (peer to peer) tyyppinen. Tällöin PDU specific kenttään tulee vastaan ottavan solmun osoite. Tilanteessa jossa PDU format kentän arvo on välillä 240 ja 255 viestillä on useampi vastaan ottaja. Tällöin PDU specific kenttään tulee vastaanottaja ryhmän tunnus. Vastaan ottaja ryhmä voi olla esimerkiksi kaikki väylältä löytyvät paineanturit. Ryhmät ovat etukäteen määriteltyjä ja arvolla 255 viesti on broadcast tyyppiä eli kaikki väylän solmut kuulevat sen. Source address Kenttä kertoo kuka viestin on lähettänyt, se on yksilöllinen jokaisella väylän solmulla. (Wilfried 2005 J1939, 43-45)

2.5.3 Parametrit

SAE J1939 käyttää valmiiksi määriteltyjä parametrejä viestimiseen. Ne ovat myös luokiteltu parametriryhmiin. Valmiit ryhmät ovat lueteltu SAE J1939 standardissa ja ne ovat määritelty SAE J1939/71:ssä. Lähes kaikki mahdolliset moottoreihin ja liikkuviin työkoneisiin liittyvät parametrit on listattu valmiiksi, mutta tarvittaessa niitä on myös mahdollista luoda itse lisää. Parametrit ovat tärkeä työkalu väylän viestintä ohjelmaa tehtäessä. Alla on esimerkki valmiiksi määritellystä parametrista. (Wilfried 2005 J1939, 45-46)

Taulukko 2: Parametri

SPN 110	**Engine Coolant Temperature**
Temperature of liquid engine cooling system	
Data Length	1 Byte
Resolution	1 deg C / Bit
Data Range	-40 to 210 deg C
Type	Measured
Reference	PGN 65262

Alla on selitetty parametritaulukon suureita:

- SPN (Suspect Parameter Number) on parametrin tunnistenumero.
- Data Length kertoo kuinka monta tavua tai bittiä lähetettävä tieto tarvitsee.
- Resolution kertoo viestittävän suureen tarkkuuden eli kuinka monta astetta tai sen osaa yksi bitti vastaa.
- Data range kertoo millä välillä parametri voi saada arvoja.
- Type on mistä parametri on saatu. Suureet kuten paine ja lämpötila tyypillisesti saadaan mittaamalla. Viesti voi olla myös status eli tila tyyppinen on / off arvo.
- Reference kertoo mihin parametriryhmää kyseinen parametri kuuluu.

2.4.4 Parametriryhmät

Parametrien ryhmittely on tehty useimmiten tyypin mukaan kuten öljy, jäähdytysneste, polttoaine jne. Väylällä viestit liikkuvat aina parametriryhmissä eivätkö yksittäisinä parametreina. Jotta väylän liikenne voidaan määrittää, täytyy tuntea speksi jonka mukaan väyläliikenteen parametriryhmät on laadittu. Alla on esimerkki parametriryhmästä. (Wilfried 2005 J1939, 46)

Taulukko 3: Parametriryhmä

PGN 65262	**Engine Temperature**
Transmission Rate	1 sec
Data Length	8 bytes
Data Page	0
PDU Format (PF)	254
PDU Specific (PS)	238
Default Priority	6
PG Number	65262 (FEEE hex)

Description of Data		
Byte		**SPN**
1	Engine Coolant Temperature	110
2	Fuel Temperature	174
3, 4	Turbocharger Oil Temperature	175
5, 6	Turbocharger Oil Temperature	176
7	Engine Intercooler Temperature	52
8	Engine Intercooler Thermostat Opening	1134

Parametri ryhmä numero 65262 on moottorilämpötiloille. Taulukos selviää viestin tärkeysaste väylällä, kuinka monta tavua se vie kaikkiaan ja viestin lähetysmuoto väylälle (PDU). Taulukon alaosassa on mitä arvoja ryhmään kuuluu ja minkä tavun paikalle ne kuuluvat. Viestin tärkeysaste väylällä ja kuinka monta tavua se vie kaikkiaan.

Transmission rate kertoo kuinka usein tieto lähetetään väylälle. Sen arvo voi olla myös vain pyynnöstä tai riippuen jostain muusta muuttujasta. Säännöllinen ajasta riippuva arvo on tosin yleinen.

2.5.5 Multi packet lähetys

SAE J1939 tukee myös isompia viestejä kuin 8 tavua. Kun lähetetään isompia tietomääriä solmu aloittaa lähetyksen BAM (broadcast announce message) viestillä. Tässä viestissä solmu ilmoittaa kenelle viesti lähtee, paljonko siinä on tietoa ja montako 8 tavun pakettia lähetys sisältää. Tieto liikkuu edelleen 8 tavun paketeissa mutta näitä kehyksiä on peräkkäin niin monta kuin tarvitaan, pisimmän mahdollisen viestin ollessa 1785 tavua. (Wilfried 2005 J1939, 71-73)

3 CODESYS

Codesys on saksalaisen automaatioyrityksen, 3S-Smart Software Solutions, tuote. Se on suunniteltu kattavaan logiikkaohjelmointiin ja käytettäväksi kaikissa logiikoita sisältävässä automaatiossa. Codesys ohjelmoitavia logiikoita on käytössä lukuisissa suomalaisissa yrityksissä. Pääasiassa sitä on käytössä liikkuvien työkoneiden ohjelmoinnissa. Codesys täyttää IEC 61131 standardin, joka on International Electrotechnical Comission:in laatima tardardi ohjelmoitavista logiikoista. (CoDeSys 2011)

Ohjelmointi Codesyssillä oli minulle mielenkiintoinen haaste, koske se oli ennalta tuntematon ohjelmisto. Minulla on aikaisempaa kokemusta muiden valmistajien logiikka ohjelmoinnista. Codesys osoittautui hyväksi työkaluksi ja se vakuuttaa haasteellisimmissa ohjelmointi kohteissa sen monipuolisuudellaan. Siinä on lukuisia ominaisuuksia ja työkaluja jotka helpottavat ja nopeuttavat ohjelmoijan työtä.

3.1 Ohjelmointikieli

Codesys:issä on käytössä lukuisia eri vaihtoehtoja ohjelmointikieleksi tai tavaksi. Ohjelmoija pysyy valitsemaan ennestään tutun ohjelmointityylin. Tämä tekee ohjelman helpoksi sisäistää. Jos kyseessä ei ole hyvin yksinkertainen sovellus niin ohjelma koostuu lukuisista aliohjelmista. Aliohjelmat voivat olla tehty eri ohjelmointikielillä ja silti kokonaisuus toimii ongelmitta. Tämä on kätevä ominaisuus, koska erilaiset toiminnot ovat helpompia ja selkeämpiä ohjelmoida toisella kielellä kuin toisella. Esimerkiksi ohjelman tietokantamainen osuus on helpointa tehdä Strucktured Text tyylillä, kun taas loogiset vertailut ja if-lauseet hoituvat kävevimmin Function Block Diagram muodossa. Itse käytin ohjelmoimiseen juuri kahta edellämainittua ohjelmointityyliä. Seuraavissa kappaleissa on esitelty hieman kaikkia Codesysin kuutta eri mahdollista ohjelmointityyliä.

3.1.1 Structured Text (ST)

Structured text on C-ohjelmointikielien kaltainen syntaksirakenteinen ohjelmointityyli. Valmiiksi määriteltyjä komentoja on lukuisia ja tällä tyylillä on mahdollista tehdä hyvin korkealuokkaista koodia ja monimutkaisia ohjelmistorakenteita. Aikaisemmin C:llä, Pascal:illa tai Java:lla ohjelmoineet sisäistävät ST ohjelmoinnin hyvin nopeasti. Minulla kyseiset kielet ovat hieman oudompia. ST-kieleen tutustuminen ja sillä ohjelmoiminen osoittautui erittäin hyväksi oppimiskokemukseksi. (CoDeSys 2011)

3.1.2 Function Block Diagram (FBD)

Function Block Diagram on erittäin havainnollinen ja graafinen tyyli ohjelmoida. Se sisältää kaikki IEC 61131-3 standardin määrittelemät toiminnot, joka käsittelee logiikan ohjelmointi kieliä. Valmiiksi määriteltyjä toimilohkoja lisäämällä saadaan helposti ja nopeasti tehtyä boolen sääntöjä noudattavia toimintoja. Codesys sisältää laajan kirjaston valmiiksi määriteltyjä toimilohkoja, jotka sisältävät erittäin monimutkaisiakin toimintoja. Lisäksi on mahdollisuus luoda omia toimilohkoja, jos valmiista kirjastosta puuttuu jotain. Useimmat algebraaliset toiminnot ovat myös nopeita tehdä lohkojen avulla. Näytön valmistaja, IFM electronics, on laatinut valmiita toimilohkoja näytön ohjelmoimista helpottamaan. Osa niistä liittyy suoraan kyseisen näytön ohjelmoimiseen ja osa helpottaa CAN-väylällä kommunikointia. (CoDeSys 2011)

FBD on minulle tutuin logiikkaohjelmointityyli lukuisten sitä käsittelevien koulussa käytyjen kurssien takia. Aikaisemmin olen ohjelmoinut Siemensin Simatic step 7 logiikka ohjelmointi työkalulla. Codesys:in vastaava toimilohko tyyli on hyvin samankaltainen, joten pääsin nopeasti sisälle siihen. Tämä vahvisti ajatustani useiden rinnakaisten ohjelmointitapojen hyödyllisyydestä. Matala oppimiskynnys aikaisemman tietämyksen perusteella on iso plussa kun ohjelmistoja valitaan työkäyttöön.

3.1.3 Ladder Diagram (LD)

Tikapuukaavio on tuttu sähköpuolen osaajille. Se on graafinen ohjelmointityyli, joka muistuttaa hyvin pitkälle relekytkentöjen piirrustuksia. Se toimii erittäin hyvin yksinkertaisissa sovelluksissa, mutta itse valitsen toimilohkotyylin tikapuukaavion sijaan. Minulla on jonkin verran kokemusta tikapuukaavio-ohjelmoinnista. Olen ohjelmoinut logiikoita Feston FST työkalulla ja havaintoni oli että function blokeilla ohjelmoiminen on huomattavasti loogisempaa kuin tikapuurakennetta käytettäessä. Codesys:in versiossa on mahdollista liittää mukaan myös toimilohkoja ja käyttää valmiiksi määriteltyjä monimutkaisempia toimintoja. (CoDeSys 2011)

3.1.4 Instruction List (IL)

Komentolistassa on toimintoja allekkain ja ne luetaan yksi kerrallaan. Se on ohjelmointikielten alaluokkaa eli symbolinen konekieli. Symbolinen konekieli on havainnollisempi muoto tietokoneiden ymmärtämästä binäärimuotoisista numerojonoista. Se on määritelty IEC 61131-3 standardissa yhtenä logiikkaohjelmointikielenä. Logiikoiden ohjelmoimisessa tämä on minulle tuntematon muoto, joten en käyttänyt sitä myöskään tässä projektissa. (CoDeSys 2011)

3.1.5 Sequential Function Chart (SFC)

Sequential function chart on kehitetty ohjelman rakenteen helppoon hallintaan. Toiminnot listataan askeleina ja toiminnot suoritetaan kun kyseinen askel on aktiivisena. Askelten välillä siirrytään boolen algebraa noudattavien ehtojen mukaisesti. Tämä on vähemmän käytetty ohjelmointimalli logiikoissa ja myös minulle oudompi ratkaisu. (CoDeSys 2011)

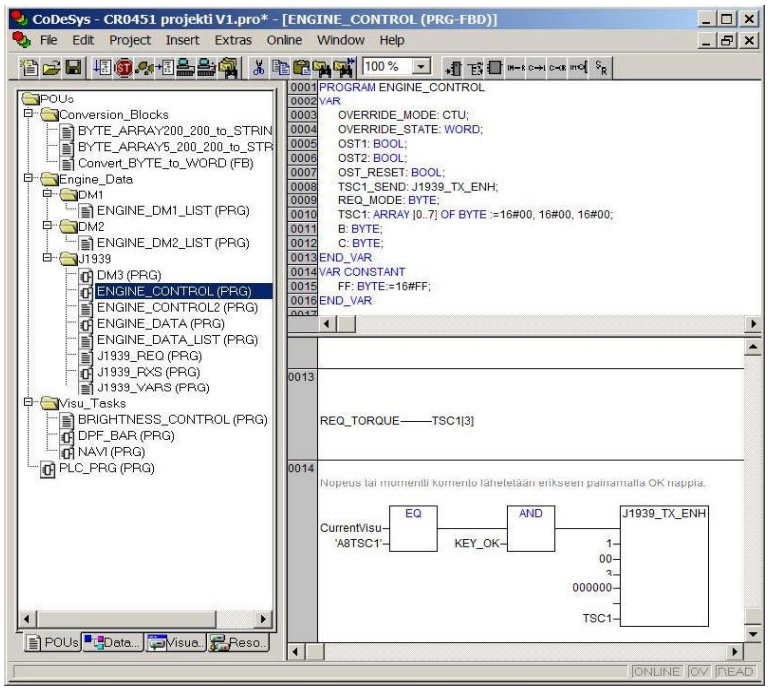
3.1.6 Continous Function Chart (CFC)

Tämä on ainoa mainituista logiikka ohjelmointitavoista jota ei mainita IEC 61131-3 standardissa. Continous function chart on muunnnelma Function block-diagramista. Eroina on esimerkiksi CFC sallii takaisinkytkentä silmukoiden käytön, inputtien väliset yhteyden, omien makrojen luonnin sekä outputit eivät ilmesty automaattisesti vaan ohjelmoijan täytyy piirtää ne itse. En käyttänyt ohjelmoinnissa CFC tyyliä, mutta sen mahdollisuudet vaikuttavat mielenkiintoisilta ja sillä on varmasti paikkansa joissakin sovelluksissa. (CoDeSys 2011)

3.2 Codesys ominaisuudet

Seuraavissa kappaleissa on käsitelty Codesys:in ominaisuuksia oman työni kannalta. Siitä löytyy erittäin paljon sovellusmahdollisuuksia, koska sillä on yritetty kattaa kaikki logiikkaohjelmoinnin ongelmat. Tämän työn parissa pystyin tutustumaan vain osaan niistä, joten tässä on käsitelty vain oleellisempia.

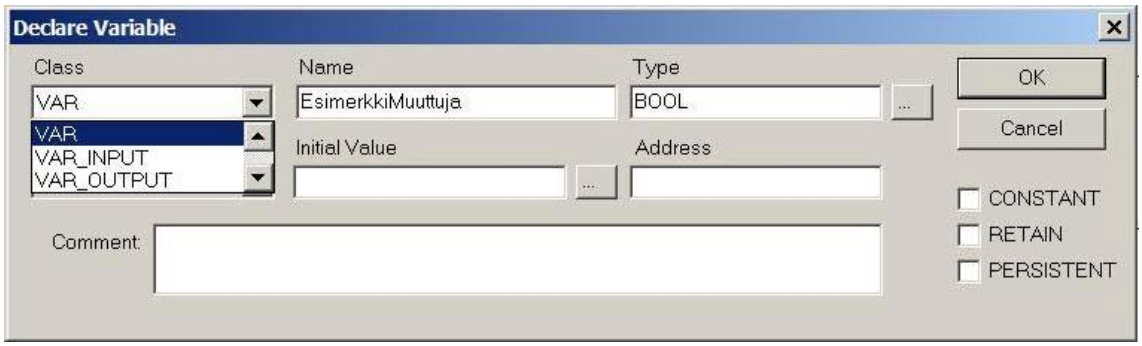
Codesys:in normaali työskentely näkymä on jaettu kolmeen eri laatikkoon. Ääri vasemmalla on ohjelman rakennepuu, Program organization unit (POU). Aliohjelmia voi lisätä tarpeen mukaan ja järjestää kansioihin ohjelman rakenteen hahmottamisen helpottamiseksi. Uusia ohjelmia lisättäessä täytyy valita ohjelmointikieli tarpeen mukaan. Oikealla alhaalla on ohjelmointiikkuna valitulla ohjelmointityylillä. Alla olevassa kuvassa on valittuna FBD. Ohjelmaa tehtäessä muuttujille täytyy antaa nimi, määrittää niiden data tyyppi sekä mihin ryhmään ne kuuluvat. Määritetyt muuttujat näkyvät yläoikealla olevassa ikkunassa. (User Manual for PLC Programming with CoDeSys 2.3 2004)



Kuvio 12: Codesys ikkuna

3.2.1 Auto declare

Muuttujien määrittäminen on pakollista, jotta muuttujia voin käyttää outputteina tai niihin voidaan viitata myöhemmin ohjelmassa. Määrittäminen voidaan tehdä suoraa yläoikealla olevaan muuttuja ikkunaan tai ohjelman rakentamisen yhteydessä käyttäen auto declare toimintoa apuna, joka on huomattavasti nopeampi ja helpompi tapa. Muuttujille on määritettävä ovatko ne paikallisia eli kyseisen ohjelman sisäisiä. Tällöin ne lisätään listaan VAR niin kuin variable. Tälläin ne näkyvät yläoikealla olevassa ikkunassa. Jos muuttujiin on tarkoitus viitata muissa aliohjelmissa niiden pitää olla luokkaa VAR_GLOBAL eli global variable. Näille on mahdollista luoda alaluokkia tai ryhmiä niiden jäsentelyä helpottamaan. Muita mahdollisia määrittely luokkia on VAR_INPUT ja VAR_OUTPUT. Näitä käytetään kun halutaan ohjelmoida suoraa logiikan sisään ja ulostuloja. (CoDeSys 2011)



Kuvio 13: Auto Declare ikkuna

Muuttujalle on myös määritettävä datatyyppi. Niitä on olemassa useita ja ohjelmoija täytyy tietää mikä on kyseisen muuttujan tehtävä ja kuinka paljon tilaa sen mahdollinen tietomäärä tarvitsee. Alla esimerkkinä omassa ohjelmassani käytetyt datatyypit.

Taulukko 4: Data tyypit

data muoto		alaraja	yläraja
BOOL	päälle tai pois	0	1
BYTE	numero esim prosentti	0	255
WORD	luku	0	65535
DWORD	luku	0	4294967295
REAL	desimaali luku	1.17*E-38	3.4*E38
STRING	tekstiä	1 merkki	255 merkkiä
INT	negatiivinen luku	-32768	32767

3.2.2 Syntax Coloring

Avainsanat koodista on väritetty automaattisesti. Tämä selkeyttää koodin rakennetta ja näin helpottaa koodin lukemista. IEC 61131-3 standardin mukaiset käskyt: VAR, IF, BYTE jne. ovat korostettu sinisellä värillä. Avainsanan tunnistaessaan väri vaihtuu automaattisesti. Väärin kirjoitetut avainsanat ja virheet ilmoitetaan punaisella värillä. Kommentointi on väritetty vihreäksi. Aikaan liittyvät arvot ja boolen ilmaisut esim. TRUE ovat purppuran värisiä. (CoDeSys 2011)

3.2.3 Muita tärkeitä ominaisuuksia

Erittäin tärkeä ja hyvin toimiva ominaisuus on debugging. Se on ohjelman automaattinen tarkastus, joka ilmoittaa jos jokin ohjelmoitu asia ei ole mahdollista tai on tehty väärin. (CoDeSys 2011)

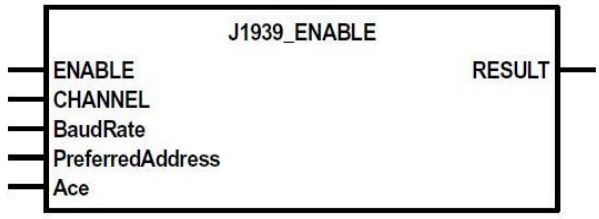
Global search on kätevä työkalu, joka etsii ohjelman läpikotaisin ja näyttää kaikki paikat missä haettu muuttuja vaikuttaa. (User Manual for PLC Programming with CoDeSys 2.3 2004)

Codesys:issä on kattava Help-tietokanta, josta löytyy apua lähes joka ongelmaan. Siellä on selitetty valmiiden toimilohkojen toimintaa sekä seikkaperäistä opastusta ohjelman toiminnasta. (User Manual for PLC Programming with CoDeSys 2.3 2004)

3.3 Valmiit Function Block

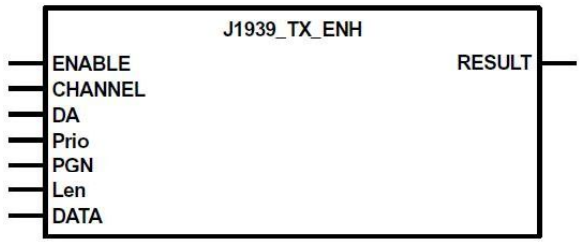
Näytön valmistaja IFM electronics tarjoaa näytön mukana Codesys-kirjastoja joista löytyy mm. valmiita FB:eja jotka helpottavat huomattavasti näytön ohjelmoimista. Näyttö on tarkoitettu liitettäväksi CAN väylään ja näillä valmiilla toimilohkoilla se onnistuu. Projektissa käytettiin SAE J1939 protokollaa mutta muitakin mahdollisuuksia on sisällytetty valmiisiin lohkoihin. (System manual BasicDisplay 2011)

J1939_ENABLE toimilohko luo asetukset ja mahdollisuuden käyttää J1939 protokollaa. Ohjelmoija pystyy tällä lohkolla vaikuttamaan siirtonopeuteen (standardi 250 kbit/s), lähdeosoitteeseen ja kytkemään adress claiming toiminnon päälle tai pois.



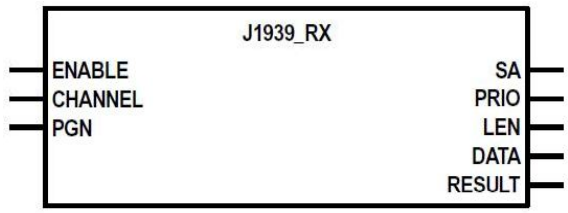
Kuvio 14. J1939 valtuutus toimilohko

J1939_TX_ENH toimilohko on helppo tapa lähettää yksi kehyksinen viesti. ENH=enhanced eli tehostettu, mahdollistaa laajennetulla 29 bitin osoitekentällä varustetun viestikehyksen lähetyksen. Pelkällä TX lohkolla voi lähettää normaalia CAN viestikehystä. Lohkoon voidaan syöttää lähetettävän datan lisäksi viestin tärkeysarvo, PGN ja osoite johon viesti lähetetään.



Kuvio 15. Viestin lähetys toimilohko

J1939_RX toimilohkolla vastaanotetaan väylältä saapuvia viestejä. Lohkon oikealla puolella olevista ulostuloista saadaan vastaanotettu viesti. Niistä käy ilmi viestin lähettäjän osoite, sen tärkeys, viestin tavumäärä sekä tietoa toimilohkon ja väylän sen hetkisestä tilasta.



Kuvio 16. Viestin vastaaotto toimilohko

Nämä ovat kolme tärkeitä ohjelmassa käytettyä toimilohkoa. Muitakin valmiiksi määriteltyjä erikoislohkoja on käytetty, mutta ne ovat osittaisia muunnelmia näistä lohkoista. Valmiiden lohkojen kirjasto on laaja ja sieltä löytyy vastaus lähes kaikkiin näytön ja väylän välisiin tilanteisiin.

4 NÄYTTÖ

Työ rakentui pääasiassa IFM:n BasicDisplay CR0451 mallisen näytön ympärille. Näyttö on kohtalaisen uusi malli IFM:n tuotevalikoimassa ja edustaa malliston halvinta päätä. Siinä on 2.8 tuumainen värinäyttö sekä tukeva silikoninen näppäimistö. Näyttö on hyvin kevyt ja siinä on kovamuovista tehty tukevan oloinen runko. Näyttö ja näppäimistö ovat hyvin vapaasti ohjelmoitavissa ja siihen liitytään näytön takaa löytyvällä M12 CAN liittimellä. Ohjelmoinnin rajoituksena on että se noudattaa kansainvälisen sähköalan standardointiorganisaatio IEC:n laatimaa IEC 61131-3 ohjelmointistandardia. IFM tarjoaa ohjelmointi työkaluksi Codesys-ohjelmistoa, joka on hyvin joustava logiikka ohjelmointi työkalu. (System manual BasicDisplay 2011)

Kuva 1. Basic display CR0451 (System manual BasicDisplay 2011, 11).

4.1 Grafiikan suunnittelu

Codesys:iin on sisällytetty ominaisuus visuaalisten elementtien luontiin ja se mahdollistaa graafisen esityksen ohjelmoitavassa logiikassa. Tässä projektissa ohjelmoitavan näyttölaitteen graafiikan tuottaminen onnistuu helposti, eikä sen integroimiseen ohjelmaan täten tarvita erillisiä ohjelmistoja.

Tilanteen salliessa grafiikka kannattaa luoda ennen varsinaisen ohjelman tekemistä. Ennen visualisaation luomista on hyvä tietää mitä mahdollisia rajoituksia laitteessa on. Tälläisiä voivat olla: resoluutio, värien määrä, sallitut fontit, sallitut fontti koot, sallitut codesys:istä löytyvät visualisaatiotyökalut, sallitut graafiset elementit, kuvan koko yms. Alla olevasta taulukosta selviää osa näytön ohjelmoimista rajoittavista tekijöistä. Lisää olennaista tietoa on liitteessä 1.

Taulukko 5. Näytön rajoitukset (System manual BasicDisplay 2011)

Kuvaus	**Rajoitus**
Yhden merkkijonon (string) pituus	≤ 80
Visualisaatiosivujen määrä	≤ 15
Graafisten kohteiden määrä yhdellä sivulla	≤ 20
Kuvien määrä koko projektissa	≤ 256
POU lukumäärä projektissa	≤ 512

Toinen erittäin tärkeä huomioitava seikka on käyttäjäystävällisyys. Koneen ja ihmisen välinen rajapinta (HMI) pitäisi olla mahdollisimman yksinkertainen ja helposti omaksuttavissa. Vaikka ohjelmoitava koodi olisikin hyvin monimutkainen, itse laitteen käyttäminen pitäisi olla helppoa. Seuraavia asioita pitäisi ottaa huomioon rajapintaa suunniteltaessa:

- Jokaisella toiminnolla selkeä kuvaus.
- Toimintojen tulisi olla olettamien mukaisia eli aikaisemmin opittuja ja tuttuja käyttöliittymiä kannattaa mukailla.
- Hyvä luettavuus. Valaistus- ja ympäristöolosuhteet.
- Intuitiivinen käytettävyys. Valintojen pitäisi olla niin ilmeisiä etteivät ne kaipaa lisäselitystä.
- Laitteen täytyy reagoida nopeasti käyttäjän käskyihin.
- Valinnoista pitää saada palautetta kuittaus viestin muodossa.
- Standardien mukaisuus.

Jos tunnetaan laitteen loppukäyttäjät, on käyttöliittymän suunnittelu huomattavasti helpompaa. Suunniteltaessa käyttöliittymään tulisi tuntea sen loppukäyttäjät ja heidän toiminnalliset tarpeensa. Ainakin seuraavia asioita ja niiden vaikutusta olisi hyvä miettiä käyttöliittymää suunniteltaessa. (System manual BasicDisplay 2011)

- Ikä voi heikentää aisteja kuten näköä. Ikä myös heikentää kognitiivisiä taitoja.
- Sukupuoli
- Yleinen koulutustaso. Tarvitaanko laitetta käyttämään insinööri vai riittääkö peruskoulu käyttöliittymän ymmärtämiseen?
- Laitteen käyttökoulutus tai aiempi käyttökokemus vastaavista laitteista.
- Kieli
- Erilaisten merkkien ymmärrys ja lukusuunta.
- Symbolien ja värien merkitys. Eri kulttuureissa värit symboloivat eri asioita.
- Kuinka usein laitetta käytetään? Useasti käytetyn laitteen toiminnot muistaa helpommin kuin harvoin käytetyn.

Nämä käyttöliittymään liittyvät asiat ovat melko yleispäteviä tuotteita suunniteltaessa. Huomioon täytyy myös ottaa ympäristön vaikutus käyttäjään. Käyttöympäristössä voi olla monia keskittymistä haittaavia tekijöitä kuten pimeys, pöly, tärinä, melu sekä käyttäjän väsymystila. Kaikkia näitä täytyy miettiä käyttöliittymää suunniteltaessa. (System manual BasicDisplay 2011)

5 OHJELMA

Projektini pohjalla oli IFM edustajan tekemä puolitoimiva ohjelma ja tavoitteena oli viilata sitä kuntoon sekä tehdä joitakin lisäyksiä. Sain Comatec:iltä työohjeen (Liite 2) mitä muutoksia ohjelmaan tulisi tehdä. Ongelmien ratkaisuissa sain kuitenkin käyttää omaa harkintaani ja muokata ohjelmaa parhaaksi katsomallani tavalla parempaan suuntaan. Liitteenä 3 on osia ohjelmasta. Ohjelmaa tehdessäni yritin kommentoida toimintoja mahdollisimman paljon, jotta muut sitä lukiessaan ymmärtävät helpommin mitä kyseisessä vaiheessa tapahtuu.

Näytöltä luetaan moottorin diagnostiikkaa jota se lukee väylältä. Ohjelmassa oli valmiina iso joukko näytettäviä suureita mm. kierrosnopeus, lämpötiloja eri pisteissä, moottorinesteiden määriä, paineita yms. Kaikki mitä anturitietoina voi kuvitella saavansa, voidaan esittää näytöltä. Lisäsin ohjelmaan moottorin viimeisestä huollon ajankohdasta ilmoittavan viestin sekä seuraavan huollon ajankohdasta ilmoittavan viestin. Tämän pienen lisäyksen tekemiseksi jouduin perehtymään perusteellisesti ohjelman toimintaan.

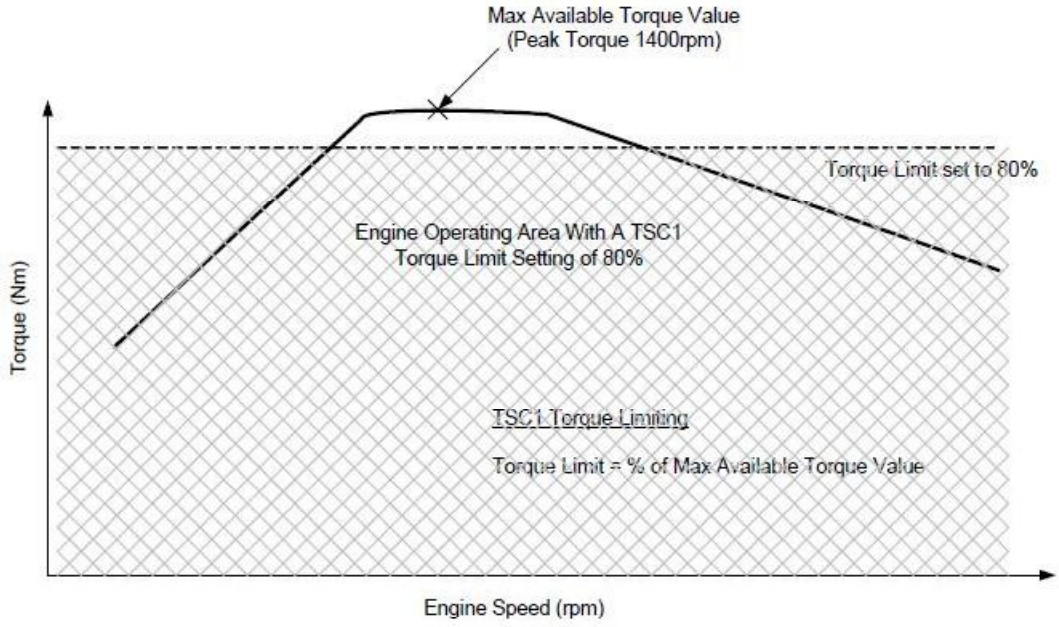
Parametreille on määritetty lähetys taajuus eli kuinka usein sitä tarvitsee lähettää väylälle. Joitakin viestejä lähetetään väylälle 10 ms välein eli todella tiuhaan. Parametreille joille ei ole ilmoitettu lähetys taajuutta täytyy pyytää väylältä. Osana työtäni oli arvioida onko parametrin oletettu lähetystaajuus riittävä vai onko tarvetta hakea sitä pyynnöllä. Näin saadaan tehokkaampi väylä liikenne kun turhaa tiedonsiirtoa vähennetään väylällä.

Näyttö lähettää väylälle pyynnön halutusta parametrista. Parametria mittaava solmu lähettää väylälle parametrin tiedot. Näyttö poimii nämä tiedot väylältä ja esittää ne ruudulla. Näytön käyttäjä valitsee nuoli näppäimillä mitä parametria halutaan esitettävän. Jos mitään erityistä ei haluta tarkastella oletuksena esitetään moottorin kierrosnopeutta. Osana toimeksiantoa suunnittelin etusivulle uuden esitysasun.

Osa väylältä saapuvista viesteistä ovat skaalaltaan epähavainnollisia. Esim. aika seuraavaan huoltoon ei ole järkevää ilmoittaa tunteina vaan päivinä. Nämä viestit piti skaalata mielekkäämmiksi. Saapuvalle suureelle tehdään ohjelmassa matemaattisia toimintoja, jotta näytön näkymä on mahdollisimman havainnollinen.

Moottorin suureiden näyttämisen lisäksi näytön tärkeä tehtävä on ilmoittaa mahdollisista vioista moottorissa. Vikatiedote vastaanotetaan väylältä tarkoitusta varten suunnitellulla DM1 toimilohkolla. Aktiivisesta viasta näyttö hälyttää käyttäjälleen. Vanhoista vioista on vikalogi, jota pystyy myös selaamaan näytöllä. Tätä koskien lisäsin näyttöön uuden ominaisuuden jolla DM2 vikatieto logi saadaan nollattua. Vikatieto logi ei sijaitse näytössä vaan jossain väylän solmuista, joten sen nollaamiseksi näytön pitää lähettää väylälle pyyntö vikahistorian poistamisesta. Loin ohjelmaan oman sivun tätä toimintoa varten, jossa ohjelma varmistaa että käyttäjä varmasi ymmärtää millaista pyyntöä on väylälle lähettämässä.

Suuri ohjelmaan tekemäni lisäys on moottorin ohjaus näytön kautta. Näytöllä voidaan ohjata moottorin kierrosnopeutta sekä momenttia. Tätä varten on olemassa valmis parametri ryhmä TSC1. Se ajaa yli kaikista muista moottorin ohjaustavoista kuten pulssin leveys modulaatio ohjatusta kaasusta ja analogisista ohjaimista. Sen takia on tärkeää ettei näytöllä näennäisen helposti moottoria ohjaa kukaan joka ei ole siihen huolellisesti ensin perehtynyt. Tällä parametriryhmällä voidaan myös asettaa momentti ja nopeus rajoja joita muilla ohjauksilla ei myöskään voida ylittää. Alla olevassa esimerkkitapauksessa momentti jo rajoitettu 80% mutta nopeutta ei ole rajoitettu mitenkään vaan se toimii normaalisti.



Kuvaaja 1. Momentin rajoittaminen

Nopeuden ja momentin säätämistä varten piti suunnitella näyttöön jonkinlainen käyttöliittymä. Luontevasti arvojen säätämiseen sopivat näytön nuoli napit. Hieman ongelmaa ohjelmoinnissa aiheuttaa ensimmäisen tavun asettaminen. Kuten alla olevasta taulukosta näkyy sillä ohjataan haluttu toiminto päälle tai pois. Kuitenkin vain kaksi ensimmäistä bittiä tavusta ohjaavat tätä. Ohjelmoin käyttöliittymän siten että nopeuden tai momentin säätämiseksi toiminto pitää kytkeä päälle, jotta näyttöön ilmestyy säätövalikko. Haluttu arvo täytyy lähettää erillisestä napista, jotta tahattomilta ohjauskomennoilta vältytään. Ohjausta käyttäessä on myös huomioita pieni väylän viive sekä isompi viive itse moottorissa mekaniikasta johtuen.

Taulukko 6. TSC1 Parametriryhmä

Identifier	Rate (msec)	PGN	Default Priority	R1	DP	Source	Destination
0C 00 00 xx	10	000000	3	0	0	See notes	00

Send	Receive	Parameter name	Byte	Bit	Length	State	Units	Resolution (unit/bit)	Range		Note
									Min	Max	
	X	**Override Control Mode (spn 695)**	1	1	2						
	X	Override Disabled				00					
	X	Speed Control				01					
	X	Torque Control				10					
	X	Speed/Torque Limit Control				11					
	X	**Requested Speed Control Conditions (spn 696)**		3	2						
	X	**Override Control Mode Priority (spn 897)**		5	2						A
	X	Highest Priority				00					A
	X	High Priority				01					A
	X	Medium Priority				10					A
	X	Low Priority				11					A
		Not Defined		7..8							
	X	**Requested Speed / Speed Limit (spn 898)**	2	1	16		Rpm	0.125	0	8032	
	X	**Requested Torque / Torque Limit (spn 518)**	4	1	8		%	1	-125	+125	B

6 YHTEENVETO

Työlle asetetut tavoitteet toteutuivat. Näytön käyttöliittymä on havainnollisempi ja monipuolisempi työssä lisättyjen ohjaus ominaisuuksien ansiosta. Oli harmillista että projektin aikataulusyistä ei ollut mahdollista päästä testaamaan ohjelmaa todellisessa toimintaympäristössä moottorin kanssa. Pelkässä CAN-väylässä näyttö vastasi komentoihin moitteettomasti. CAN-väylän perusteiden tunteminen on välttämätöntä, kun ohjelmoidaan laitteita, jotka käyttävät sitä tiedonsiirtoon. Sen takia oli hyvä ensin tutustua hyvin CAN-väylään sekä käytettyyn SAE J1939 protokollaan. Uuden tiedon omaksumista tässä työssä vaikeutti standardien kaupalliset ehdot. Tästä johtuen tässä työssä niitä ei saatu käyttöön. Standardien vapaa saatavuus lisäisi osaavien kehittäjien määrää tällä teknologian saralla, mikä hyödyttäisi kaikkia alan toimijoita.

7 POHDINTA JA JOHTOPÄÄTÖKSET

Aikaisempi kokemus CAN-väylästä sekä Codesys-kehitysympäristöstä oli varsin vähäistä. Tämän takia opinnäytetyö oppimisnäkökulmasta palveli hyvin tarkoitustaan ja opin paljon uutta tämän projektin aikana. Codesys-ohjelmointiin tutustuminen oli mielenkiintoista ja uskon sen tuntemuksesta olevan hyötyä tulevaisuudessa. Comatec Oy tarjosi hyvät puitteet tämän työn tekemiseen ja asiantuntevaa opastusta Codessys-kehitysympäristön käyttöön. Vaikka työ oli haasteellinen ja sisälsi paljon uutta asiaa niiden omaksumista helpotti opintojeni aikana saama koulutus. Täten projekti onnistui hyvin myös näytteenä insinööriosaamisen näkökulmasta erittäin hyvin.

Projektin eteneminen ei sujunut alkuperäisten suunnitelmien mukaan. Matkalla jouduttiin mukautumaan muutoksiin ja tekemään uusia suunnitelmia. Haluankin kiittää työnohjaajaa Ilkka Lannetta joustavasta suhtautumisesta opinnäytetyöprojektissa.

CAN-väylä ei ole enää uusi keksintö, mutta se puolustaa silti paikkaansa ajoneuvojen sisäisessä tiedonsiirrossa eikä lähitulevaisuudessa sille näy konkreettista korvaajaa. Moottorien diagnostiikan laatu ja sovellusten laajuus on kasvussa. Työkoneiden älykkyyttä ja ohjelmistojen osuutta lisätään, jotta tuotteen säilyvät kilpailukykyisinä. Opinnäytetyön aiheen ajankohtaisuus teki siitä entistä mielenkiintoisemman.

LÄHTEET

Alanen J. 2000. CAN – ajoneuvojen ja koneiden sisäinen paikallisväylä. VTT Automaatio.

CAN in Automation. 2011. Luettu 17.4.2011.
http://www.can-cia.org/

CoDeSys. 2011. Luettu 15.5.2011.
http://www.3s-software.com/

System manual BasicDisplay. 2011. IFM electronic.

User Manual for PLC Programming with CoDeSys 2.3. 2004. Saksa: 3S - Smart Software Solutions GmbH.

Wilfried V. 2005. A Comprehensible Guide to Controller Area Network. USA: Copperhill Media Corporation.

Wilfried V. 2005. A Comprehensible Guide to J1939. USA: Copperhill Media Corporation.