Vesa Norrbacka

CAN-väylärajapinnat taksiautoissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

29.11.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Vesa Norrbacka CAN-väylärajapinnat taksiautoissa 99 sivua 29.11.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	sulautettu tietotekniikka
Ohjaajat	Tuomo Roivainen lehtori Keijo Länsikunnas

Insinöörityössä tutkittiin Suomessa yleisimmin taksiautoina käytettävien henkilöautojen CAN-väylärajapintoja ja suunniteltiin Semel Oy:n TM6000-taksamittarille ohjelmointirajapinta, jonka avulla auton lisälaiteväylään liitettyjä laitteita voidaan ohjata ja niistä voidaan lukea tietoa. Tutkimuksessa keskityttiin CAN in Automation -standardointijärjestön CANopen-standardiin ja CiA447-sovellusprofiiliin, jota useat eurooppalaiset autovalmistajat ovat olleet mukana kehittämässä.

Ohjelmointirajapinta suunniteltiin C-kielellä IAR Embedded Workbench -kehitysympäristöä käyttäen. Ohjelmisto sisältää CAN-ajurin, CANopen-pinon ja CiA447-yhteensopivan IVN-yhdyskäytävän testiohjelman. Asiakkaan vähimmäisvaatimukset ohjelmistolle olivat, että sen avulla tulisi pystyä ohjaamaan sisävaloja ja lukemaan matkamittarin lukema ja polttoaineen kulutus.

Ohjelmistoa testattiin ensin Embedded Systems Academyn CiA447 Gateway Simulation -ohjelman avulla ja sen jälkeen taksivarustellun E-sarjan Mercedes Benzin lisälaiteväylään liitettynä. Testauksen tuloksena huomattiin, että simulaatio-ohjelman ja auton IVN-yhdyskäytävät eivät toimi täysin CiA447-sovellusprofiilin mukaisesti. Simulaatio-ohjelman versio 1.63 jättää käynnistyksen yhteydessä yhden kyselyviestin välistä, eikä tue toimintojen ohjaamista ohjausobjekteilla. Mercedes Benzin yhdyskäytävän objektikirjasto sekä tila- ja ohjausobjektien rakenne poikkeaa CiA447-profiilista. Tämä saattaa johtua siitä, että auton yhdyskäytävä tukee CiA447-profiilin versiota 1.0, kun ohjelmistoa suunnitellessa ja tätä dokumenttia kirjoitettaessa saatavilla oli vain profiilin versio 2.0.

Asiakkaan vähimmäisvaatimuksista saatiin Mercedes Benzissä toteutettua ainoastaan matkamittarilukeman lukeminen IVN-yhdyskäytävästä. Polttoaineenkulutus-arvo saatiin luettua, mutta se ei pitänyt paikkaansa ja pysyi aina samana. Sisävalojen ohjaaminen ei myöskään onnistunut. Tässä työssä saatujen tietojen avulla se tosin tulee onnistumaan pienellä lisäselvitystyöllä.

CAN-ajuria ja CANopen-pinoa voidaan myöhemmin käyttää CAN-väyläominaisuuksien lisäämiseksi taksamittarin tuotantoversioon ja CiA447-testiohjelmalla voidaan kartoittaa auton CiA447-yhteensopivan IVN-yhdyskäytävän objektikirjasto ja testata ohjausobjektien toimintaa.

Avainsanat	CAN, väylä, CANopen, CiA447, taksi

Author Title Number of Pages Date	Vesa Norrbacka CAN bus interfaces in taxicabs 99 pages 29 November 2013
Degree	Bachelor of Engineering
Degree Programme	Information technology
Specialisation option	Embedded engineering
Instructors	Tuomo Roivainen Keijo Länsikunnas, Senior Lecturer

In the project described in this thesis, CAN bus interfaces in taxicabs were studied. An application programming interface for controlling and reading from devices connected to the vehicle internal bus was also developed for the TM6000 taximeter of Semel Oy.

The API was designed in C language and includes a CAN driver, CANopen stack and a CiA447-compliant program for testing IVN gateways. The customer requirements included that it should be possible to control interior lights and read the odometer and fuel consumption.

The software was tested with CiA447 Gateway Simulation by ESAcademy and in a Mercedes Benz taxicab. Test logs indicate that the simulator and Mercedes gateways do not fully conform to CiA447 specification. The simulator skips initial polling that should take place before the beginning of LSS Fastscan, and it jumps directly to sending the LSS Fastscan start message. The Mercedes gateway object dictionary structure and the structures of some status and control objects in the dictionary do not comply with CiA447.

From the customer requirements, only reading of the odometer value was fulfilled. Fuel consumption could be read, but the value was constant. Further, the interior lights could not be controlled in the Mercedes, but a way to do this can be found with some more work.

Keywords	CAN, bus, CANopen, CiA447, taxi

Sisällys

1 Johdanto

Tämän insinöörityön tavoite on selvittää miten Suomessa yleisimmin taksiautoina käytettävien henkilöautojen sisäisestä tietoverkosta saataisiin luettua tietoja ja miten verkossa olevia laitteita pystyttäisiin ohjaamaan. Työn tilaaja on Semel Oy, vuonna 1971 perustettu yritys, joka toimittaa mobiilikäyttöön soveltuvia tietojärjestelmiä ja ohjelmistoja sekä toimii ohjelmisto- ja järjestelmäintegraattorina vaativissa mobiilijärjestelmähankkeissa. Semel on Pohjoismaiden johtava tietojärjestelmien toimittaja taksialalla.

Käytännössä jokaisessa uudessa taksiautossa on yksi tai useampia CAN-väyliä, joihin elektroniikkalaitteet on liitetty. Ongelmaksi muodostuu kuitenkin se, että käytetyt protokollat ovat yleensä valmistajakohtaisia, eivätkä valmistajat mielellään anna tietoa niistä.

Eurooppalainen standardointijärjestö Can in Automation (CiA) on kuitenkin määritellyt CANopen-protokollan päälle CiA447-sovellusprofiilin autojen erikoislisälaitteita varten. Profiilissa määritellään muun muassa niin sanottu IVN-yhdyskäytävä (In Vehicle Network), joka erottaa lisälaiteväylän auton sisäisestä väylästä siten, että lisälaitteet saavat yhteyden valittuihin toimintoihin auton sisäisessä väylässä pääsemättä kuitenkaan vikatilanteessakaan häiritsemään väylän laitteita.

Osana insinöörityötä suunnitellaan Semelin taksamittarille ohjelmointirajapinta, jonka avulla voidaan IVN-yhdyskäytävän kautta lukea tietoja ja ohjata auton sisäisessä verkossa olevia laitteita. Vähimmäisvaatimuksina yhdyskäytävän kautta tulisi pystyä ohjaamaan sisävaloja ja sen kautta tulisi pystyä lukemaan matkamittarin lukema ja polttoaineen kulutus.

Ohjelmisto suunnitellaan tukemaan CANopen-protokollan CiA447-sovellusprofiilin niitä osia, jotka ovat välttämättömiä toivotun toiminnallisuuden saavuttamiseksi. Esimerkiksi CiA447-profiilin määrittelemiä virransäästöominaisuuksia ei toteuteta, koska ne eivät ole tarpeellisia taksamittarijärjestelmässä. Ohjelmistoon pitää kuitenkin toteuttaa sellainen toiminnallisuus, että taksamittarisolmu ei häiritse tai estä lisälaiteväylän muiden solmujen toimintaa.

Ohjelmiston toiminta testataan ensin CiA447 Gateway Sim -simulaattori-ohjelmalla ja sen jälkeen varsinaisessa Mercedes Benz -taksiautossa.

Tämän insinöörityön toisessa luvussa määritellään CAN-väyläteknologia, jota autoissa käytetään. Kolmannessa luvussa esitellään lyhyesti standardointijärjestö CAN in Automation ja kuvaillaan CANopen-protokolla, joka on CAN-väylän päälle kehitetty sovellustason protokolla. Neljännessä luvussa käydään läpi CiA447-sovellusprofiili. Viidennessä luvussa esitellään projektissa käytettävä laitteisto ja työkalut. Kuudennessa luvussa käsitellään taksamittarille luotu CiA447-rajapinta ja seitsemännessä luvussa sen testaus.

2 CAN-väylä

Robert Bosch GmbH esitteli vuonna 1986 uuden sarjamuotoisen väyläjärjestelmän henkilöautoihin. CAN-väylän (Controller Area Network) suunnitelun ajavana voimana oli autojen monimutkaistuvat elektroniset järjestelmät. CAN-väylä kehitettiin lisäämään toiminnallisuutta ajoneuvon sisäisiin järjestelmiin ja sivutuotteena myös johdotuksen tarve väheni huomattavasti. [1.]

Vaikka CAN-väylä suunniteltiin käytettäväksi autoissa, ensimmäiset käytännön toteutukset sillä olivat täysin eri aloilla. Muun muassa suomalainen hissivalmistaja Kone käytti CAN-väylää hisseissään ensimmäisten joukossa 80-luvun lopulla. Samaan aikaan Ruotsissa Kvaser suositteli CAN-väylää tekstiililaitteiden sisäiseksi kommunikaatioprotokollaksi. Tästä muodostui myöhemmin korkeamman tason protokollastandardi CAN Kingdom. Hollannissa Philips Medical Systems käytti CAN-väylää röntgenlaitteissaan ja kehitti ensimmäisen sovellustason protokollan CAN-väylälle. [1.]

Autoissa CAN-väylä yleistyi eurooppalaisissa autoissa vasta 90-luvun alkupuolella ja Kaukoidän autoissa 90-luvun loppupuolella. Aluksi CAN-väylää käytettiin autoissa lähinnä moottorinohjausjärjestelmissä, mutta myöhemmin väylän käyttöaluetta laajennettiin myös korielektroniikkaan. Nykyisissä autoissa, varsinkin eurooppalaisissa, lähes kaikki auton elektroniikkalaitteet on liitetty CAN-väylään tai johonkin muuhun vastaavaan väylään. Moottorinohjausjärjestelmässä käytetty väylä on yleensä erotettu muihin tarpeisiin käytetystä väylästä. Uusissa autoissa onkin siksi yleensä kaksi tai useampia toisistaan erotettuja CAN-väyliä. Väylät on usein yhdistetty yhdyskäytävillä (gateway), joiden avulla tiedon siirtäminen väylästä toiseen onnistuu ilman vaaraa siitä, että väylät häiritsisivät toisiaan. [1; 2, s. xv.]

Koska CAN-väylästandardi määrittelee ainoastaan tiedonsiirtoverkon fyysisen kerroksen ja siirtokerroksen, eikä ota kantaa datan sisältöön, pitää toimivaa järjestelmää varten aina määritellä jokin korkeamman tason protokolla. Aluksi laitevalmistajat loivat aina omat protokollansa, mutta myöhemmin on eri yritysten ja järjestöjen toimesta luotu useita standardisoituja ja yleisessä käytössä olevia protokollia. Näistä yleisimpiä ovat CAN in Automation -järjestön kehittämä CANopen, joka sopii moninaisten laite- ja sovellusprofiiliensa vuoksi hyvin moneen käyttöön, Allen Bradleyn kehittämä DeviceNet, jota käytetään tehdasautomaatioon, SAE J1939, jota käytetään hyvin laajasti raskaassa ajoneuvokalustossa. Muita yleisiä korkeamman tason protokollia ovat ASAM-järjestön CCP, joka on tarkoitettu lähinnä autojen elektronisten ohjausyksiköiden väliseen tiedonsiirtoon, OSEK/VDX, jota myös käytetään autoissa, sekä jo mainittu CAN Kingdom. [1.]

Taulukko 1. CAN-solmun tyypillinen rakenne OSI-mallin mukaisesti [3, s. 7].

7. Sovelluskerros • Sovellusten toiminnallisuus • Sovellustason protokolla	Prosessori (Sovellus)
2. Siirtokerros LLC-alikerros (Logical Link Control) • Viestikehysten suodatus • Ylikuormituksen ilmaisu • Virheistä palautuminen MAC-alikerros (Medium Access Control) • Tiedon pakkaus viestikehyksiin/-kehyksistä • Viestikehysten koodaus/purku (bit-stuffing) • Virheiden tunnistaminen • Virhetiedon välitys • Viestikehysten kuittaus • Sarja-/rinnakkaismuunnokset **1. Fyysinen kerros** PLS-alikerros (Physical Signalling) • Bittitason koodaus/purku • Bittitason ajoitukset • Synkronointi	CAN-ohjain
PMA-alikerros (Physical Medium Attachment) • Lähettimen ja vastaanottimen ominaisuudet	CAN-sovitin
MDI-alikerros (Medium Dependent Interface) • Liittimen tyypit ja pinnijärjestykset **0. Siirtomedia** • Kaapelityyppi	Kaapelointi

CAN-väylästandardi ISO 11898 määrittelee OSI-mallin (Open Systems Interconnection Reference Model) kerrokset 1 ja 2, eli fyysisen kerroksen ja siirtokerroksen. Toimiva järjestelmä tarvitsee näiden lisäksi jonkinlaisen sovelluskerroksen protokollan, eli tavan

käyttää CAN-viestikehyksiä. Myöskään liitintyyppiä ei ole määritelty standardissa vaan se määritellään yleensä sovelluskerroksen protokollan yhteydessä. [4.] Taulukko 1 esittää CAN-solmun tyypillisen rakenteen ja eri lohkojen toiminnot OSI-mallin mukaisesti.

Can-ohjaimen tehtävä on hoitaa kaikki CAN-kommunikointiin liittyvä, kuten viestien lähetykset vastaanotot, viestien kuittaukset sekä törmäys- ja eheyskäsittelyt. CAN-sovitinpiiri muuttaa CAN-ohjaimen lähetys- ja vastaanottosignaalit CAN-väyläsignaaleiksi. [3, s. 6.]

Prosessori ja CAN-ohjain on liitetty toisiinsa rinnakkaisella muistiväylällä tai sarjamuotoisella SPI-väylällä. Useissa nykyaikaisissa mikro-ohjaimissa on yksi tai useampi integroitu CAN-ohjain, jota ohjataan konfiguraatiorekistereiden kautta. CAN-ohjain liitetään ulkoiseen sovitinpiiriin vähintään lähetys- ja vastaanottosignaaleilla. Toisinaan ohjain- ja sovitinpiirien välillä käytetään apusignaaleja esimerkiksi lähetyksen estoa, virransäästötilaa tai lähettimen reunojen nopeuksien säätöä varten. [3, s. 7.]

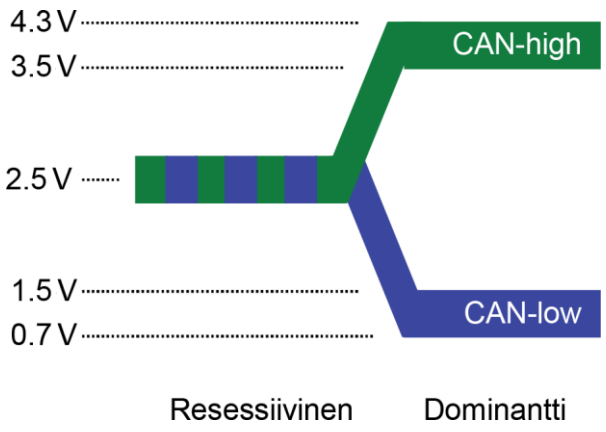
2.1 Fyysinen kerros

CAN-väylälle on määritelty kolme erilaista fyysistä kerrosta:

ISO 11898-2 / high speed CAN

Yleisimmin käytetty fyysinen kerros on niin sanottu high speed CAN, joka on määritelty standardissa ISO 11898-2. Väylän maksiminopeus on 1Mbit/s ja maksimipituus tällä nopeudella on noin 40 metriä. Laskemalla siirtonopeutta on kuitenkin mahdollista käyttää huomattavasti suurempia pituuksia. Signalointi on differentiaalinen ja johdotuksessa käytetään kierrettyä paria. Ominaislinjaimpedanssi on 120 ohmia, ja väylä tulee terminoida molemmista päistä. Solmujen määrää rajoittaa ainoastaan väyläkuorma. [4.]

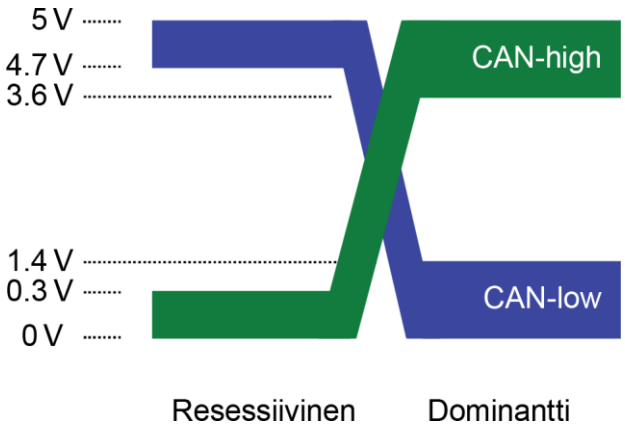
Väylällä on kaksi tilaa: resessiivinen (looginen 1) ja dominantti (looginen 0). Resessiivisessä tilassa jännitepotentiaali molemmissa johtimissa on 2,5 V, eli niiden erotus on 0. Dominantissa tilassa CAN-high on 3,5…4,3 V ja CAN-low on 0,7…1,5 V, eli erotus on 2 voltista 3,5 volttiin. [1, s.8.] Jännitetasot näkyvät kuvassa 1.



Kuva 1. ISO 11898-2 jännitetasot [3, s. 8].

ISO 11898-3 / fault-tolerant CAN

Vikasietoista ISO 11898-3 -standardin mukaista fyysistä kerrosta käytetään enimmäkseen autojen kori-elektroniikassa [4]. Se vastaa toimintaperiaatteltaan high speed CAN:ia, mutta lähetettävien väyläsignaalien välistä jännite-eroa on kasvatettu ja vastaanotettavien tasojen välistä minimijännite-eroa on pienennetty siten, että väylä toimii myös pelkästään CAN-high'n tai CAN-low'n avulla. [3, s. 8.] Jännitetasot näkyvät kuvassa 2. Väylän maksiminopeus on 125 kbit/s johtuen suuremmasta jännite-erosta. Väylän pituus tällä nopeudella on korkeintaan 500 metriä. Vikasietoista väylää ei tarvitse terminoida eikä sen väylätopologian tarvitse olla lineaarinen. Solmujen määrä on rajoitettu 32:een. [4.]



Kuva 2. ISO 11898-3 jännitetasot [3, s. 8].

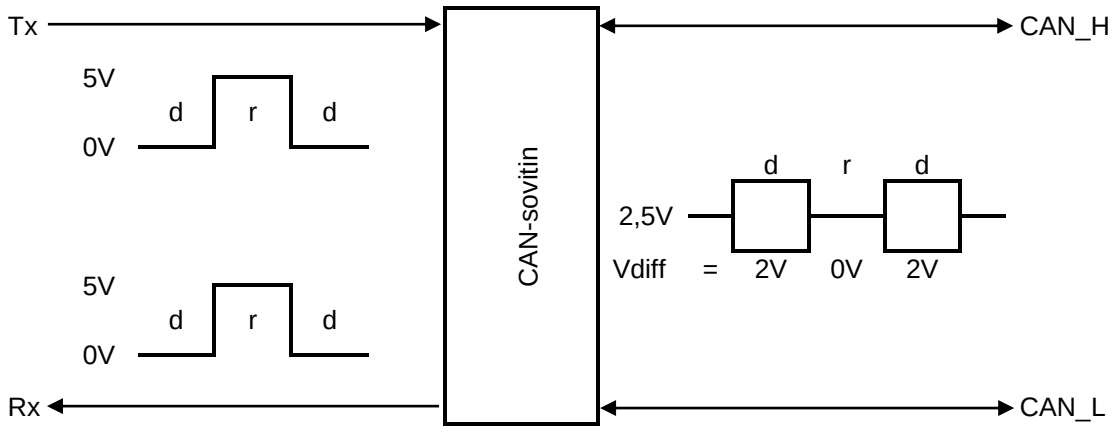
SAE J2411 single wire

SAE J2411 standardin mukaisen single wire -väylän yleisin käyttökohde on auton niin sanotun mukavuuselektroniikan ohjaus. Single wire -väylässä tiedonsiirto tapahtuu vain yhtä, CAN-high'ta vastaavaa väyläsignaalia pitkin. Siirtonopeus on rajoitettu 33,3 kb:iin/s ja solmujen määrä 32:een. Single wire -väylän topologian ei myöskään tarvitse olla lineaarinen. [4.]

Signaalit: Resessiivinen ja dominantti

Dominanttia tilaa vastaa looginen 0. CAN-sovitinpiirin TTL-tasoisissa lähetys- ja vastaanottosignaaleissa tämä tarkoittaa noin 0 volttia. Väylässä dominantti tila tarkoittaa 2 voltin eroa väyläjohtimien välillä. CAN-high ajetaan noin voltin ylöspäin ja CAN-high noin voltin alaspäin. [2, s. 207.]

Resessiivistä tilaa vastaa looginen 1. Sovitinpiirin signaaleissa tämä tarkoittaa 5 volttia (tai 3 V 3 voltin laitteissa). Väylässä resessiivinen taso tarkoittaa 0 voltin erotusta väyläjohtimien välillä. Molemmat johtimet ajetaan noin 2,5 volttiin. [2, s. 207.] Kuvassa 3 kirjain ”d” vastaa dominanttia ja ”r” resessiivistä tilaa.



Kuva 3. Resessiiviset ja dominantit signaalit ennen ja jälkeen CAN-sovittimen [2, s. 207].

CAN-väylässä dominantti signaali kirjoittaa resessiivisen tilan yli. CAN-väylä käyttäytyy kuin looginen AND-operaatio. Väylä on dominantissa tilassa, jos yksikin noodi kirjoittaa väylään dominantin bitin. Väylä on resessiivisessä tilassa vain jos kaikki noodit kirjoittavat väylään resessiivisen bitin. Tätä mekanismia käytetään törmäysten tunnistamiseen. Lähettäessään bitin väylään solmu samalla lukee väylän tilan, ja jos luettu arvo eroaa lähetetystä, on tapahtunut törmäys tai lähetysvirhe. Kehysten välissä, kun väylä

on vapaa, kaikki solmut lähettävät resessiivistä signaalia, eli väylä on resessiivisessä tilassa. [2, s. 207.]

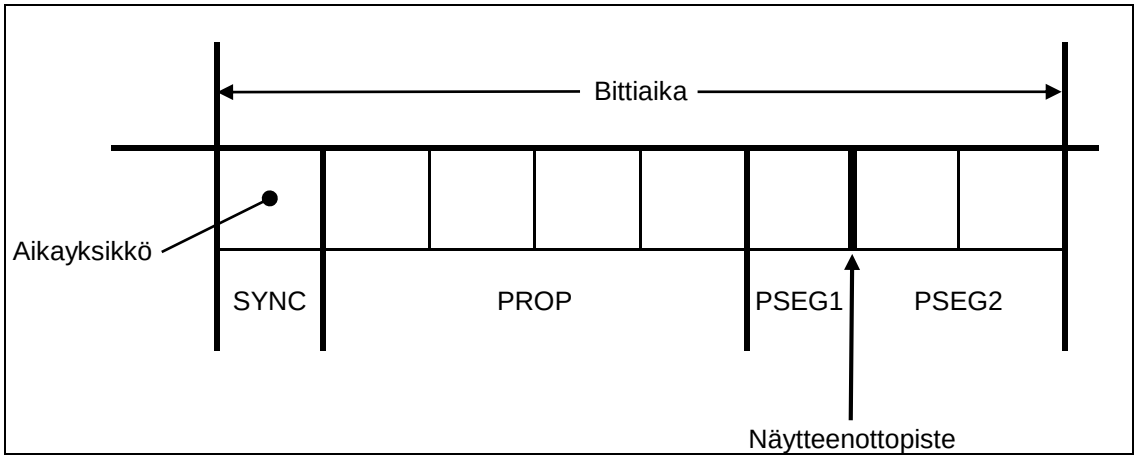
Bittikoodaus

CAN-väylässä käytetään Non Return to Zero -bittikoodausta (NRZ), mikä tarkoittaa sitä, että signaalitaso pysyy samana koko bittiajan [2, s. 210].

Bit-stuffing

Koska käytössä ei ole erillistä kellosignaalia, väylä sykronoidaan signaalin transienttien perusteella. Mikäli kehyksessä on paljon peräkkäisiä samanarvoisia bittejä, transientte- ja tulee liian harvoin. CAN-väylän kehyksissä käytetäänkin ns. bit-stuffing -tekniikkaa synkronoinnin varmistamiseen. Viiden peräkkäisen samaa arvoa olevan bitin jälkeen lisätään yksi vastakkainen bitti, jonka vastaanotin poistaa ennen kehyksen käsittelyä. [2, s. 210.]

Bittiajoitus



Kuva 4. Bittitason ajoitus CAN-väylässä

Yhden bitin lähetysaika on pituudeltaan $\frac{1}{väylän\ nopeus\ (bit/s)}$, se koostuu 8...25 aikayksiköstä (time quantum) ja on jaettu neljään osaan. Kuvassa 4 on esitetty yksi bitti ja sen koostumus eri segmenteistä.

Synkronointisegmentin *SYNC (synchronization segment)* pituus on yksi aikayksikkö ja sen aikana väylän solmut tahdistuvat. Kaikkien väylän bittien tulisi alkaa tämän aikaikkunan sisällä [5, s. 27].

Etenemissegmenttiä *PROP (propagation segment)* käytetään kompensoimaan väylän fyysisestä rakenteesta johtuvia viiveitä. Sen pituus on 1...8 aikayksikköä, jonka aikana signaalin tulisi ehtiä lähettävän solmun sovitinpiirin läpi, koko väylän päästä päähän, vastaanottavan solmun sovitinpiirin läpi ja takaisin [5, s. 27].

Ensimmäistä vaihevirheiden kompensointisegmenttiä *PSEG1 (phase buffer segment 1)* käytetään transienttien vaihevirheiden kompensoimiseen ja sitä voidaan pidentää jos solmun pitää uudelleentahdistua hitaammin lähettävän solmun lähetteeseen. Segmentin pituus on 1...8 aikayksikköä. [5, s. 28.]

Toista vaihevirheiden kompensointisegmenttiä *PSEG2 (phase buffer segment 2)* voidaan lyhentää, mikäli solmun pitää tahdistua nopeammin lähettävän solmun lähetteeseen. Segmentti on vähintään IPT:n ja korkeintaan PSEG1:n pituinen, eli 2...8 aikayksikköä. [5, s. 28.]

Informaation käsittelyaika *IPT (information processing time)* on osa PSEG2-segmenttiä, joka alkaa välittömästi näytteenottopisteen jälkeen. Sen pituus on enintään kaksi aikayksikköä ja tarkoitus on antaa CAN-ohjaimelle aikaa käsitellä näytteistetty bitti. [5, s. 28.]

Synkronointihypyn pituus *SJW (Synchronization Jump Width)* on aika, minkä verran solmu voi enintään pidentää PSEG1:tä tai lyhentää PSEG2:ta uudelleensynkronoinnin yhteydessä. Sen pituus on vähintään 1 aikayksikkö ja enintään PSEG1:n pituus, kuitenkin enintään 4 aikayksikköä, vaikka PSEG1 olisi suurempi kuin 4 aikayksikköä [5, s. 28].

Synkronointi

Kaikki väylän vastaanottavat solmut tulee tahdistaa lähetteeseen. Jokaisen bitin pitää siis alkaa vastaanottavan solmun synkronointisegmentin aikana. Koska solmujen sisäiset kellotaajuudet saattavat erota toisistaan hieman, näin ei kuitenkaan aina tapahdu. Siksi tarvitaan synkronointitekniikoita. CAN-väylälle on määritelty kaksi erilaista synkronointitapaa, kova synkronointi ja uudelleensynkronointi.

Kun väylän tila muuttuu väylän vapaan tilan aikana resessiivisestä dominantiksi, eli viestikehys alkaa, tapahtuu niin sanottu *kova synkronointi*. Tämä tarkoittaa sitä, että kaikki solmut kohdistavat alkavan bittiajan synkronointisegmentin tapahtuneen transientin kohdalle.

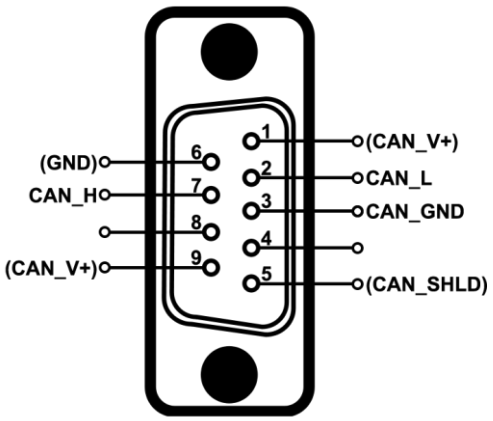
Kaikki muut kehyksen aikana tapahtuvat transientit resessiivisestä dominanttiin voivat aiheuttaa *uudelleensynkronoinnin*. Uudelleensynkronointi tapahtuu, mikäli vastaanottava solmu havaitsee transientin tapahtuneen synkronointisegmentin ulkopuolella.

Mikäli transientti tapahtuu synkronointisegmentin jälkeen, mutta ennen näytteenottopistettä, katsotaan transientin olevan myöhässä ja PSEG1:tä pidennetään niin monen aikayksikön verran kuin transientti oli myöhässä, kuitenkin enintään SJW:n verran. Käytännössä tämä tarkoittaa, että seuraavaa näytteenottopistettä viivästetään sen verran, että se on asetuksissa määritetyn aikayksikkömäärän päässä varsinaisesta transientista.

Jos transientti tapahtuu näytteenottopisteen jälkeen, ennen synkronointisegmenttiä, katsotaan bitin alkaneen etuajassa ja PSEG2:ta lyhennetään niin monen aikayksikön verran kuin transientti oli etuajassa, enintään SJW:n verran. Käytännössä tämä tarkoittaa, että seuraavan bitin synkronointisegmentti siirretään alkamaan välittömästi transientin jälkeen. [5, s. 29–30.]

Liittimet

CAN-standardissa ei määritellä käytettäviä liittimiä, mutta yleisin käytetty liitin on DB9. Laitteissa on urosliittimet ja kaapeleissa naarasliittimet. Kuvassa 5 on liitindiagrammi laitteeseen päin katsottuna. [2, s. 212–213.]



1. CAN_V+ – vapaaehtoinen jännitelinja
2. CAN_L – CAN-low väylälinja (dominantti 1,5 V)
3. CAN_GND – CAN maa
4. ei käytössä
5. CAN_SHLD – vapaaehtoinen liitos kaapelin suojavaipalle
6. GND – vapaaehtoinen maa
7. CAN_H – CAN-high väylälinja (dominantti 3,5 V)
8. ei käytössä
9. CAN_V+ – vapaaehtoinen jännitelinja.

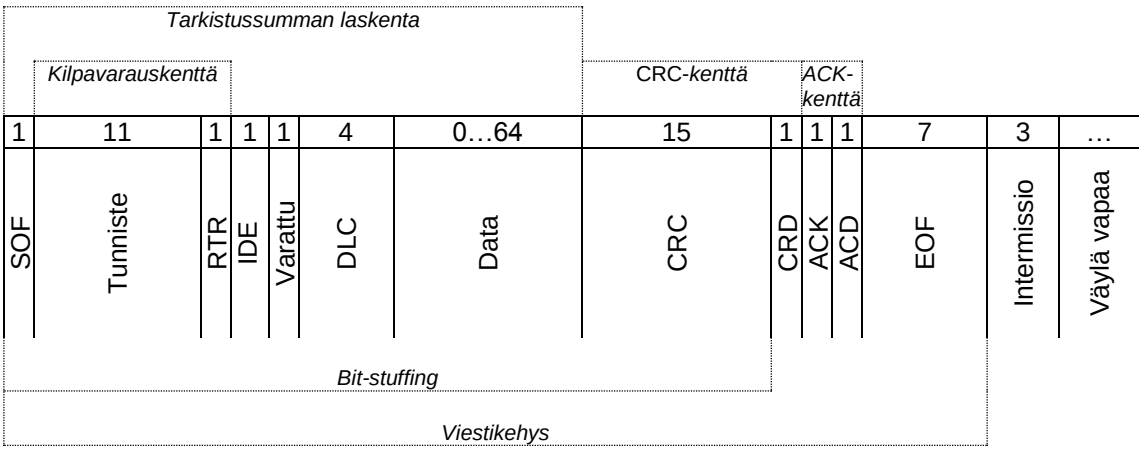
Kuva 5. DB9-liittimen signaalit

2.2 Kehykset

Viestikehys

Eniten käytetty kehys on viestikehys, joka on tarkoitettu datan siirtämiseen. Viestikehyksiä on määritelty kaksi erilaista. Niiden erona on tunnistekentän pituus. Laajennetussa kehyksessä on peruskehyksen 11 bitin tunnistekentän lisäksi 18-bittinen laajennusosa. 11 bittistä tunnistekenttää käytetään muun muassa CANopen-protokollassa ja laajennettua 29 bittistä tunnistekenttää esimerkiksi J1939-protokollassa. Kuvassa 6 on esitetty CAN-viestikehys ja sen jälkeinen vapaa tila. Alla on selitetty eri bittikentien tarkoitus.

Viestikehys alkaa *SOF-bitillä* (Start of Frame), eli kehyksen aloitusbitillä, joka on aina dominantti. Aloitusbitti muodostaa kehystenvälisen resessiivisen tilan jälkeen transientin, jonka avulla vastaanottavat solmut saadaan synkronoitua lähettävän solmun kanssa.



Kuva 6. CAN-viestikehys

Viestikehyksen *tunnistekenttää* käytetään viestikehysten yksilöimiseen ja törmäysten-käsittelyyn. Jokaisella viestillä pitää olla uniikki tunniste. Törmäystilanteessa pienimmällä tunnisteella varustettu viestikehys voittaa väylän kilpavarauksen. Törmäystenkäsittelystä kerrotaan lisää seuraavassa luvussa.

Etäpyyntöbitti (*RTR*) on viestikehyksessä dominantti ja RTR-kehyksessä resessiivinen. Näin RTR-kehyksen ja sitä vastaavan viestikehyksen törmätessä viestikehys voittaa väylän kilpavarauksen.

IDE-bitti määrää viestikehyksen tyypin. Jos se on dominantti, viestikehys sisältää 11-bittisen tunnisteen. Jos se on resessiivinen, tunniste on 29-bittinen.

Varattu-bitti on varattu tulevaisuuden käyttöä varten.

DLC-kenttä (Data Length Code) määrittää viestikehyksen sisältämien data-tavujen määrän. Vaikka DLC on 4-bittinen, vain arvot 0…8 ovat sallittuja.

Data-kenttä sisältää DLC-kentän ilmaiseman määrän tavuja.

CRC-kenttä (Cyclic Redundancy Checksum) sisältää 15-bittisen tarkistussumman laskettuna kaikista aloitusbitin jälkeisistä kentistä data-kentän loppuun asti. [5, s. 10–14.]

CRD-bitti (CRC Delimiter) on tarkistussumman erotusbitti, joka on aina resessiivinen. Sen tarkoitus on erottaa CRC-kenttä kuittausbitistä, jotta kuittaustoiminta ei häiritse CRC-kenttää mikäli vastaanottavien solmujen synkronointi on pielessä. CRD antaa

myös hitaille vastaanottaville solmuille aikaa reagoida data-kenttään ja CRC-tarkistussummaan.

ACK slot -bitti on kuittausbitti, jonka lähettävä solmu lähettää resessiivisenä. Jokaisen vastaanottavan solmun tulee kuitata viestikehys vastaanotetuksi ajamalla bitti dominanttiin tilaan. Jos väylä pysyy resessiivisessä tilassa kuittausbitin ajan, lähettää tämän huomaava solmu vikakehyksen heti kuittausbitin jälkeen.

ACD-bitti (ACK delimiter) on kuittausbitin jälkeinen erotusbitti ja se lähetetään aina resessiivisenä. Sen tehtävä on antaa tilaa ennen viestin loppuosaa sen varalta, että kuittausbittiä lähettävän solmun synkronointi on hieman pielessä ja kuittausbitin dominantti tila vuotaa seuraavaan bitti-aikaan. [6, s. 5.]

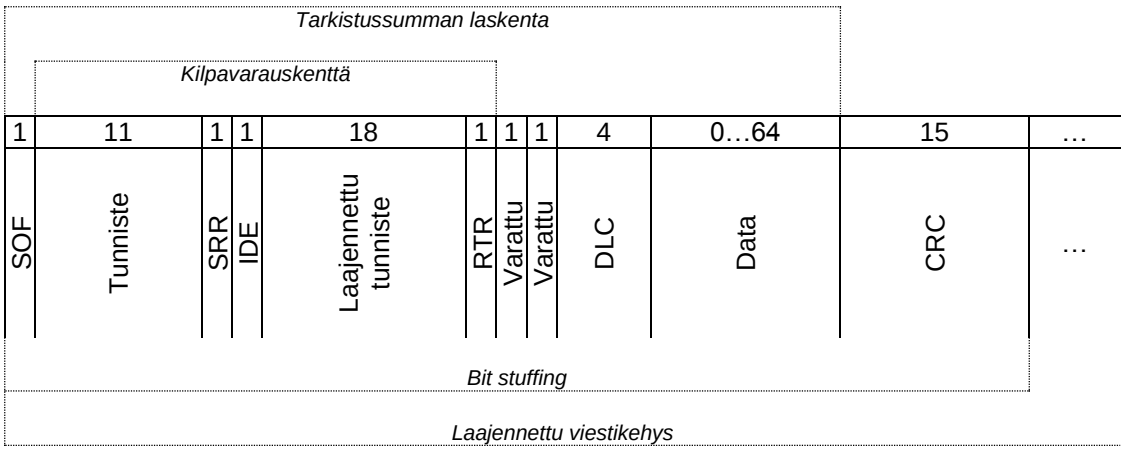
Viestikehys loppuu 7-bittiseen lopetuskenttään eli *EOF-kenttään* (End of Frame), jonka kaikki bitit ovat resessiivisiä [5, s. 14].

Viesti- ja RTR-kehykset erotetaan niitä edeltävistä kehyksistä bittikentällä jota sanotaan *kehystenväliseksi tilaksi* (interframe space). Kehystenvälinen tila sisältää kaksi osiota, nimeltään *intermissio* ja *väylä vapaa*. (intermission, bus idle)

Intermissio sisältää kolme resessiivistä bittiä. Intermission aikana mikään solmu ei saa aloittaa viesti- tai RTR-kehyksen lähettämistä. Ainoastaan viivekehyksen lähettäminen on sallittua.

Väylän vapaa tila voi olla minkä pituinen tahansa. Viestikehys joka odottaa lähetystä toisen kehyksen lähetyksen ajan voidaan lähettää välittömästi intermissio-kentän jälkeen. Vapaan tilan aikana väylältä luettu dominantti bitti tulkitaan aina viesti- tai RTR-kehyksen aloitusbitiksi. [5, s. 19.]

Laajennettu viestikehys eroaa tavallisesta viestikehyksestä siten, että tunnistebittejä on 11:n sijaan 29. Kuvassa 7 näkyy, kuinka laajennettu kehys rakentuu. Loppuosa kehyksestä, DLC-kentästä eteenpäin, on identtinen tavallisen viestikehyksen kanssa



Kuva 7. CAN:n laajennettu viestikehys.

SRR-bitti (Substitute remote request) korvaa standardikehyksen RTR-bitin laajennetussa kehyksessä. Se on aina resessiivinen. Tämän vuoksi standardikehys voittaa kilpavarauksessa laajennetun kehyksen, jolla on sama standarditunnistekenttä. [7, s. 13–14.]

RTR-kehys

Etäpyyntökehys eli RTR-kehys on viestikehyksen erikoistapaus, jonka avulla solmu voi pyytää toista solmua lähettämään viestikehyksen. RTR-kehyksen tunnistekenttä vastaa pyydettävän viestikehyksen tunnistetta, ja DLC-kenttä kuvaa vastaavan viestikehyksen sisältämien data-tavujen määrän. Itse RTR-kehys ei sisällä yhtään data-tavua. RTR-kehyksessä RTR-bitti on resessiivinen, kun se vastaavassa viestikehyksessä on dominantti. Viestikehyksen prioriteetti on siis suurempi ja se voittaa vastaavan RTR-kehyksen törmäystilanteessa. [5. s. 15.]

Virhekehys

Jokaisen solmun, joka havaitsee virheen, tulee välittömästi lähettää virhekehys. Virhekehys koostuu kahdesta osasta: virhelipusta (Error flag) ja erotuskentästä (Error delimiter). Virhelippuja on kahta eri tyyppiä, aktiivinen ja passiivinen.

Aktiivisessa tilassa olevat solmut lähettävät virheen kohdatessaan *aktiivisen virhelipun*, joka koostuu kuudesta dominantista bitistä. Tämä rikkoo bit-stuffing-säännön tai ylikirjoittaa kuittausbitit tai lopetuskentän ja aiheuttaa näin virheen kaikkiin muihinkin solmuihin, jotka jälleen lähettävät virhekehyksen. Näin virheen aiheuttama määrä peräkkäisiä dominantteja bittejä väylässä on kuudesta kahteentoista.

Passiivinen virhelipun lähettäminen tarkoittaa sitä, että passiivisessa tilassa oleva solmu odottaa, kunnes se saa luettua väylästä kuusi peräkkäistä resessiivistä bittiä.

Virhekehyksen erotuskenttä koostuu kahdeksasta resessiivisestä bitistä. Virhelipun lähettämisen jälkeen solmu siirtyy lähettämään resessiivistä signaalia ja tarkkailee väylää kunnes lukee sieltä resessiivisen bitin. Tämän jälkeen solmu lähettää vielä seitsemän resessiivistä bittiä lisää ja jatkaa sen jälkeen normaalia toimintaa. [5, s. 16–17.]

Viivekehys

Viivekehyksen lähettämiseen voi olla kaksi syytä:

1. Vastaanottava solmu tarvitsee aikaa käsitellä edellistä vastaanotettua viestikehystä tai tehdä muita toimenpiteitä ennen seuraavaa kehystä. Tästä syystä viivekehyksen saa lähettää

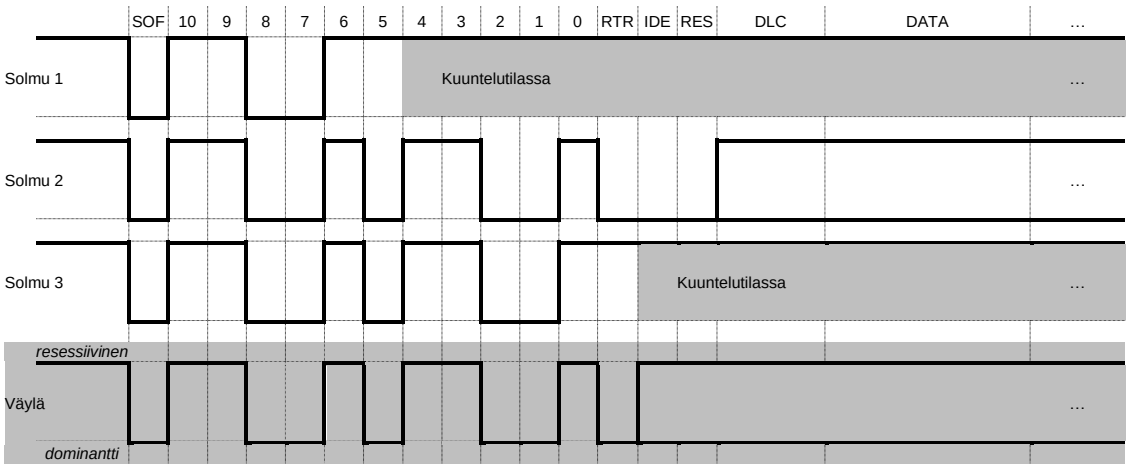
2. Solmu tunnistaa väylästä, että jokin muu solmu lähettää viivekehystä.

Viivekehys koostuu kahdesta osasta, samaan tapaan kuin virhekehys. Viivelippu sisältää kuusi dominanttia bittiä, minkä jälkeen solmu tarkkailee väylää kunnes se tunnistaa muunnoksen dominantista resessiiviseen ja lähettää tämän jälkeen vielä 7 resessiivistä bittiä.

Viivekehyksiä saa lähettää peräkkäin korkeintaan kaksi kappaletta. [5, s. 17.]

2.3 Törmäystenkäsittely

Törmäystenkäsittelyprosessi on CAN-väylän keskeisimpiä ominaisuuksia. Kun väylä on vapaa, mikä tahansa solmu voi alkaa lähettää viestiä. Jos kaksi tai useampi solmu alkaa lähettämään viestiä samaan aikaan, kilpatilanne ratkaistaan vertaamalla kehysten kilpavarauskenttiä toisiinsa bitti kerrallaan. Kuvassa 8 on esitettynä esimerkkitapaus kolmen solmun kilpatilanteesta.



Kuva 8. CAN-väylän törmäystenkäsittely.

Viestikehyksen perusmuodossa kilpavarauskenttään kuuluvat tunnistekenttä ja RTR-bitti, viestikehyksen laajennetussa muodossa taas molemmat tunnistekentät sekä bitit SRR, IDE ja RTR. Kilpavarauskentän ulkopuolella törmäykset eivät ole sallittuja ja aiheuttavat aina virhekehyksen lähettämisen. [5, s. 8–11; 7, s. 12–14.]

2.4 Virheidenhallinta

CAN-väylän virheidenhallinta jakaantuu kahteen osaan. Virheidenkäsittely pyrkii tunnistamaan virheet ja signaloimaan havaituista virheistä. Häiriöidenrajoitus rajoittaa virheitä aiheuttavien solmujen vaikutuksia virheettömästi toimivien solmujen väliseen tiedonsiirtoon.

Virheidenkäsittely

Virhetyyppejä on 5 erilaista. Virheen tunnistavan solmun tulee lähettää virhekehys. Virhekehys lähetetään välittömästi virhettä seuraavasta bitistä lähtien, paitsi CRC-virheen tapauksessa, jolloin virhekehys lähetetään ACD-bittin jälkeen.

Lähettävä solmu tarkkailee väylän tilaa. Jos väylästä luettava arvo eroaa sinne juuri kirjoitetusta arvosta, on kyse *bittivirheestä*. Poikkeuksena ovat kilpavarauskenttä ja ACK slot -bitti, joiden aikana bittivirhettä ei tapahdu, jos väylästä luettu arvo on dominantti.

Jos solmu havaitsee kuusi peräkkäistä samanarvoista bittiä kehyksen aikana, kyseessä on *bit-stuffing-virhe*.

Jos vastaanottavan solmun laskema CRC-summa eroaa vastaanotetun kehyksen CRC-kentän arvosta, kyseessä on *CRC-virhe*.

Jos kiinteästi määritellyn bitin arvo poikkeaa määritellystä, kyseessä on *muotovirhe* (form error).

Lähettävän solmun havaitessa resessiivisen tilan ACK slot -bitin aikana, yksikään vastaanottava solmu ei ole asettanut kuittausbittiä ja kyseessä on *kuittausvirhe* (acknowledgement error). [5, s. 23.]

Häiriöidenrajoitus

Virheidenhallinnan kannalta solmu voi olla kolmessa tilassa. *Aktiivinen* (error active) solmu voi ottaa vapaasti osaa väyläkommunikointiin ja lähettää aktiivisen virhekehyksen havaitessaan virheen.

Passiivinen (error passive) solmu voi ottaa osaa väyläkommunikointiin, mutta virheen havaitessaan sen pitää lähettää passiivinen virhekehys. Lisäksi passiivisen solmun tulee kehysten jälkeisten intermissio-kentän jälkeen odottaa vielä kahdeksan bitin verran eli lähettää kahdeksan resessiivistä bittiä, ennen seuraavan viestin lähetystä. Jos joku muu solmu aloittaa lähetyksen tänä aikana solmu voi ottaa viestin vastaan.

Jos solmu on *väylä irti -tilassa* (bus-off), se ei saa vaikuttaa väylään millään tavalla.

Virhelaskurit

Häiriöidenrajoitusta ja solmun tilan määritystä varten jokaisessa solmussa tulee olla kaksi virhelaskuria, lähetysvirhelaskuri ja vastaanottovirhelaskuri. Seuraavassa on listattu ISO 11898:n määrittelemät säännöt virhelaskureiden muutoksille.

1. Jos vastaanottava solmu havaitsee virheen, sen vastaanottovirhelaskurin arvo kasvatetaan yhdellä, paitsi jos kyseessä on bittivirhe aktiivisen virhekehyksen tai viivekehyksen aikana.

2. Jos vastaanottava solmu havaitsee dominantin heti virhelipun jälkeen, eli virhekehyksen erotuskentässä, vastaanottovirhelaskurin arvoa kasvatetaan kahdeksalla.

3. Jos lähettävä solmu lähettää virhekehyksen, sen lähetysvirhelaskurin arvoa kasvatetaan kahdeksalla.

4. Jos lähettävä solmu tunnistaa bittivirheen aktiivisen virhelipun aikana, sen lähetysvirhelaskurin arvoa kasvatetaan kahdeksalla.

5. Jos vastaanottava solmu tunnistaa bittivirheen aktiivisen virhelipun aikana, sen vastaanottovirhelaskurin arvoa kasvatetaan kahdeksalla.

6. Jokaisen solmun tulee sallia seitsemän peräkkäistä dominanttia bittiä aktiivisen tai passiivisen virhelipun tai viivelipun lähettämisen jälkeen. Tunnistaessaan 14. peräkkäisen dominantin bitin aktiivisen virhelipun ja viivelipun jälkeen tai 8. peräkkäisen dominantin bitin passiivisen virhelipun jälkeen, solmu kasvattaa virhelaskurin arvoa kahdeksalla jokaista seuraavaa kahdeksaa peräkkäistä dominanttia bittiä kohden.

7. Jokaisen virheettömästi lähetetyn viestikehyksen jälkeen lähettävä solmu vähentää lähetysvirhelaskurin arvoa yhdellä, kunnes sen arvo on 0.

8. Jokaisen virheettömästi vastaanotetun viestikehyksen jälkeen vastaanottava solmu vähentää vastaanottovirhelaskurin arvoa yhdellä, jos sen arvo on välillä 1...127. Jos laskurin arvo on yli 127, sen arvoksi asetetaan jokin arvo väliltä 119...127.

9. Solmu on passiivisessa tilassa, kun jompikumpi virhelaskureiden arvoista on yhtä suuri tai suurempi kuin 128.

10. Solmu on irti-väylästä-tilassa, kun lähetysvirhelaskurin arvo on yhtä suuri tai suurempi kuin 256.

11. Passiivinen solmu palaa aktiiviseksi kun molemmat virhelaskurit laskevat alle 128:n.

12. Irti-väylästä-solmu palaa aktiiviseksi, ja sen virhelaskurit asetetaan nollaan, kun se on lukenut väylästä 128 kertaa 11 perättäisen resessiivisen bitin jaksoa. [5, s.24–26.]

ISO 11898 -standardin mukaan virhelaskurin arvo, joka ylittää 96:n, osoittaa väylän olevan hyvin vikaantunut. Siksi useissa CAN-ohjaimissa, kuten Atmelin AT91SAM7X-sarjan mikro-ohjaimen integroidussa CAN-ohjaimessa, on erillinen virhevaroitustila, kun virhelaskurin arvo on 96...127.

Jos väylän käynnistyksen yhteydessä vain yksi solmu on päällä ja se lähettää viestiä, tapahtuu kuittausvirhe, jonka jälkeen solmu tunnistaa virheen ja lähettää viestin uudes-

taan. Solmu menee pian passiiviseen tilaan, mutta se ei saa tässä tilanteessa mennä väylä-irti-tilaan. [5, s. 24–26.]

3 CANopen (CiA301)

3.1 Historia

CAN-väylän suurin ongelma alkuaikoina oli standardisoidun korkeamman tason protokollan puuttuminen. Jokainen CAN-väylää laitteissaan käyttävä yritys joutui kehittämään oman ratkaisunsa. Korkeamman tason protokollan ylläpitäminen ja parantaminen oli kuitenkin yllättävän työlästä. Tämän vuoksi CAN-väylätuotteita valmistavat ja niitä käyttävät yritykset perustivat vuonna 1992 yhteenliittymän nimeltä CAN in Automation (CiA), jonka tehtävänä oli standardisoida CAN-väylään liittyvät erilaiset ratkaisut.

CiA:n ensimmäinen tehtävä oli määrittää CAN-väylän sovelluskerros (CAL). CAL oli askel eteenpäin, mutta se ei kuitenkaan ratkaissut koko ongelmaa, sillä se määrittelee verkonhallintapalvelut ja viestiprotokollan, mutta ei viestien sisältöä. Jokaisen käyttäjän piti siis edelleen kehittää oma kommunikaatioprofiili.

Vuonna 1993 Boschin johtama yhteenliittymä alkoi kehittää CAL-pohjaista profiilia sulautettujen tuotantosolujen verkostoimiseen. Tästä profiilista kehittyi myöhemmin CANopen-määritelmä, joka luovutettiin CiA:lle ylläpidettäväksi ja edelleen laajennettavaksi. Ensimmäinen virallinen versio CANopen-kommunikaatioprofiilista julkaistiin vuonna 1995. Alkuperäistä CANopen määritystä on sittemmin päivitetty ja sitä on laajennettu eri käyttöön tarkoitetuilla laite- ja kommunikaatioprofiileilla. Nykyään CANopeniin perustuvia profiileja on kymmeniä erilaisia ja CiA:n jäsenyrityksiä noin 560. [1.]

3.2 Yleistä

CANopen on CAN väylälle suunniteltu korkeamman tason protokolla. Se sisältää kommunikaatioprofiilin ja sovelluskerroksen. Kommunikaatioprofiili määrittelee fyysisen kerroksen, kommunikaatio-objektien tunnisteet sekä kommunikaatio-objektit Emergency, Timestamp ja Sync, ja toteuttaa näin yhdessä CAN-väylästandardin kanssa OSI-mallin fyysisen kerroksen ja siirtokerroksen. Taulukossa 2 on esitetty, mitä osia CAN- ja CANopen-standardit OSI-mallissa toteuttavat.

Taulukko 2. CAN- ja CANopen standardien sijoittuminen OSI-mallin kerroksiin.

<table>
<tr><td>7. Sovelluskerros</td><td>CANopen sovelluskerros
CAN in Automation sovellus- ja laiteprofiilit</td></tr>
<tr><td>6. Esitystapakerros</td><td rowspan="4">CANopen sovelluskerros</td></tr>
<tr><td>5. Istuntokerros</td></tr>
<tr><td>4. Kuljetuskerros</td></tr>
<tr><td>3. Verkkokerros</td></tr>
<tr><td>2. Siirtokerros</td><td rowspan="2">CAN
CANopen kommunikaatioprofiili</td></tr>
<tr><td>1. Fyysinen kerros</td></tr>
</table>

CANopen sovelluskerros toteuttaa osittain OSI-mallin kerrokset verkkokerroksesta ylöspäin. Sovelluskerroksen määritelmä sisältää käytettävät tietotyypit, koodaussäännöt, objektikirjaston objektit, kommunikaatiopalvelut ja -protokollat sekä verkonhallintapalvelut ja -protokollat. [10, s.13]

CANopen-standardia voidaan soveltaa moneen eri käyttöön. Tätä varten CAN in Automation on määritellyt sen päälle laiteprofiileja, jotka määrittelevät tietynlaisten laitteiden toiminnan CANopen-verkossa, ja sovellusprofiileja, jotka määrittelevät erilaisista laitteista koostuvia sovelluksia ja voivat sisältää myös lisäyksiä CANopen-standardiin.

3.3 Objektikirjasto

Jokaisen CANopen-solmun keskeisin toiminto on objektikirjasto, joka erottaa toisistaan sovelluksen ja väyläliikennöinnin. Sen avulla voidaan muun muassa tunnistaa solmu ja hallita kaikkea solmun toimintaa väylältä käsin. Objektikirjaston merkinnät sisältävät kaikki solmun kommunikaatioon ja tapahtumiin liittyvän informaation, kuten konfigurointi-, tila-, mittaus- ja ohjaustiedot. Merkinnät voivat olla vakioita, muuttujia tai tietotyyppimääritelmiä. Taulukossa 3 on listattu objektikirjaston indeksialueet ja niiden käyttötarkoitukset.

Taulukko 3. CANopen objektikirjaston rakenne [8, s. 87].

Indeksi	**Objekti**
0x0000	Ei käytössä
0x0001…0x001F	Staattiset tietotyypit
0x0020…0x003F	Moniosaiset tietotyypit
0x0040…0x005F	Valmistajakohtaiset moniosaiset tietotyypit
0x0060…0x025F	Laiteprofiilikohtaiset tietotyypit
0x0260…0x03FF	Varattu
0x0400…0x0FFF	Varattu
0x1000…0x1FFF	Kommunikaatioprofiilin alue
0x2000…0x5FFF	Valmistajakohtainen alue
0x6000…0x67FF	Standardisoidun profiilin alue, 1. looginen laite
0x6800…0x6FFF	Standardisoidun profiilin alue, 2. looginen laite
0x7000…0x77FF	Standardisoidun profiilin alue, 3. looginen laite
0x7800…0x7FFF	Standardisoidun profiilin alue, 4. looginen laite
0x8000…0x87FF	Standardisoidun profiilin alue, 5. looginen laite
0x8800…0x8FFF	Standardisoidun profiilin alue, 6. looginen laite
0x9000…0x97FF	Standardisoidun profiilin alue, 7. looginen laite
0x9800…0x9FFF	Standardisoidun profiilin alue, 8. looginen laite
0xA000…0xAFFF	Verkon muuttujien alue
0xB000…0xBFFF	Systeemimuuttujien alue
0xC000…0xFFFF	Varattu

Objektikirjasto on hakutaulukko, jolla on 16-bittinen indeksi ja 8-bittinen alaindeksi. Tämä mahdollistaa 65536 objektia ja jokaiselle objektille 256 alaindeksiä. Objekteilla, jotka sisältävät vain yhden merkinnän, on vain yksi alaindeksi, 0. Useamman arvon sisältävien objektien alaindeksi 0 sisältää objektin suurimman alaindeksin ja varsinainen data sijaitsee seuraavien alaindeksien alla. [8, s. 88.]

Tietotyypit

Objektikirjaston alkupäässä, objekteissa 0x0001...0x025F, määritellään käytettävissä olevat tietotyypit. Staattiset tietotyypit on määritelty CANopen-standardissa, ja niillä jokaisella on paikka objektikirjaston indeksialueella 0x0001...0x001F. CANopen-standardin määrittelemät staattiset tietotyypit on listattu taulukossa 4.

Näiden objektien toteuttaminen solmun objektikirjastoon on vapaaehtoista, eikä objektien sisältöä ole määritelty profiilissa. Niillä on silti tarkoitus: määriteltäessä objektikirjaston merkintöjä ja moniosaisia tietotyyppejä merkitään tietotyypiksi indeksi, joka viittaa objektikirjaston kohtaan, jossa kyseinen tietotyyppi on määritelty. Objektien, joilla on staattinen tietotyyppi, data on alaindeksissä 0, eikä muita alaindeksejä ole.(lukuun ottamatta mahdollista alaindeksiä 0xFF, josta lisää objektikoodin yhteydessä.) [8, s. 90–92.]

Taulukko 4. CANopen staattiset tietotyypit [8, s. 90–91.]

Tietotyyppi	**Selitys**	**Indeksi**
BOOLEAN	1-bittinen totuusarvomuuttuja	0x0001
INTEGER8	8-bittinen kokonaisluku	0x0002
INTEGER16	16-bittinen kokonaisluku	0x0003
INTEGER24	24-bittinen kokonaisluku	0x0010
INTEGER32	32-bittinen kokonaisluku	0x0004
INTEGER40	40-bittinen kokonaisluku	0x0012
INTEGER48	48-bittinen kokonaisluku	0x0013
INTEGER56	56-bittinen kokonaisluku	0x0014
INTEGER64	64-bittinen kokonaisluku	0x0015
UNSIGNED8	8-bittinen etumerkitön kokonaisluku	0x0005
UNSIGNED16	16-bittinen etumerkitön kokonaisluku	0x0006
UNSIGNED24	24-bittinen etumerkitön kokonaisluku	0x0016
UNSIGNED32	32-bittinen etumerkitön kokonaisluku	0x0007
UNSIGNED40	40-bittinen etumerkitön kokonaisluku	0x0018
UNSIGNED48	48-bittinen etumerkitön kokonaisluku	0x0019
UNSIGNED56	56-bittinen etumerkitön kokonaisluku	0x001A
UNSIGNED64	64-bittinen etumerkitön kokonaisluku	0x001B
REAL32	32-bittinen liukuluku	0x0008
REAL64	64-bittinen liukuluku	0x0011
VISIBLE_STRING	ASCII-koodattu merkkijono	0x0009
OCTET_STRING	Taulukko 8-bittisiä etumerkittömiä kokonaislukuja	0x000A
UNICODE_STRING	Taulukko 16-bittisiä etumerkittömiä kokonaislukuja	0x000B
TIME_OF_DAY	48-bittinen tietue, joka sisältää UNSIGNED28-arvon "millisekunteja keskiyöstä" ja UNSIGNED16-arvon "päiviä sitten 1.1.1984". Arvojen välissä on 4 bitin määrittelemätön kenttä.	0x000C
TIME_DIFFERENCE	Aikojen erotus. Rakenne on sama kuin TIME_OF_DAY-tietotyypissä.	0x000D
DOMAIN	Vapaasti määrättävän pituinen sovelluskohtaisesti määritelty tietojakso	0x000F

Staattisten tietotyyppien lisäksi voidaan määritellä moniosaisia tietotyyppejä (Complex data types), jotka sisältävät yhtä tai useita staattisia tietotyyppejä yhdistettynä taulukoksi tai tietueeksi. CANopen-standardi sisältää kuusi ennalta määrättyä moniosaista tietotyyppiä, joiden lisäksi objektikirjastossa on alueet valmistajakohtaisille ja laiteprofiilikohtaisille moniosaisille tietotyypeille. Objektit, joilla on moniosainen tietotyyppi, koostuvat yhden indeksin alla olevista alaindekseistä. Ensimmäisen alaindeksin arvo sisältää viimeisen alaindeksin numeron ja loput alaindekseistä varsinaisen datan. Moniosaisten tietotyyppien määrittelyobjektit toimivat saman periaatteen mukaisesti. Niissä ensimmäinen alaindeksi sisältää viimeisen alaindeksin numeron ja loput alaindeksit määrittelevät kyseisen alaindeksin tietotyypin. Taulukossa 5 on esimerkki moniosaisen tietotyypin määrittelyobjektista.

Taulukko 5. Esimerkki moniosaisen tietotyypin määrittelyobjektista.

Indeksi	Alaindeksi	Arvo	Selitys
0x0020	0	0x06	6 alaindeksiä
	1	0x0007	UNSIGNED32
	2	0x0005	UNSIGNED8
	3	0x0006	UNSIGNED16
	4	0x0005	UNSIGNED8
	5	0x0006	UNSIGNED16
	6	0x0005	UNSIGNED8

Moniosaistenkin tietotyyppien toteuttaminen solmun objektikirjastoon on vapaaehtoista, mutta tietotyypit määritellään tässä muodossa myös EDS-tiedostossa ja yleensä myös muussa dokumentoinnissa. [8, s. 92.]

Objektikoodi

Kaikilla objekteilla on objektikoodi, joka kuvaa objektin tyypin. Taulukossa 6 on listattu CANopenin objektityypit ja niihin liitetyt objektikoodit.

Taulukko 6. CANopenin objektityypit ja niihin liitetyt objektikoodit.

Objektin tyyppi	Selitys	Objektikoodi
NULL	Objekti, jolla ei ole datakenttiä	0
DOMAIN	Suuri datamäärä, jonka koko on määrittelemätön	2
DEFTYPE	Tyyppimääritelmä	5
DEFSTRUCT	Moniosaisen tietotyypin määritelmä	6
VAR	Yksittäinen arvo	7
ARRAY	Objekti, jolla on useampi kuin yksi alaindeksi ja jonka kaikki alaindeksit, paitsi alaindeksi 0, ovat samaa tietotyyppiä (alaindeksiä 0xFF ei lasketa)	8
RECORD	Objekti, jolla on useampi kuin yksi alaindeksi ja jonka alaindeksien tietotyypit voivat vaihdella (alaindeksiä 0xFF ei lasketa)	9

Objektikoodi määritellään useimmiten EDS-tiedostossa tai muussa dokumentaatiossa, mutta objektin tietotyypin ja objektikoodin voi myös määritellä objektin alaindeksissä 0xFF. Merkinnän tietotyyppi on UNSIGNED32, ja sen rakenne näkyy kuvassa 9.

31 24	23 8	7 0
varattu (0)	Tietotyyppi (katso taulukko 4)	Objektikoodi (katso taulukko 6)
	UNSIGNED16	UNSIGNED8

Kuva 9. Alaindeksin 0xFF rakenne.

Jos solmun kaikki objektikirjaston merkinnät sisältävät alaindeksin 0xFF ja moniosaiset tietotyypit on määritelty objektikirjastossa, koko objektikirjaston rakenne on mahdollista lukea väylän kautta. [8, s. 92]

Käyttöoikeudet

Jokaiselle objektikirjaston merkinnälle on määritelty käyttöoikeudet. Määritelmä löytyy yleensä EDS-tiedostosta tai muusta dokumentaatiosta. Käyttöoikeuksia on kolmea erilaista ja ne on määritelty väylän suunnasta katsoen. Luku-, ja kirjoitusoikeudet (rw), vain kirjoitusoikeudet (wo), vain lukuoikeudet (ro) sekä vakio (const). Viimeinen käyttöoikeus, const, tarkoittaa vain lukuoikeuksia ja sitä että arvo pysyy vakiona. [8, s. 90.]

Kommunikaatioprofiilin alue

Objektikirjaston alue 0x1000...0x1FFF sisältää kaikille CANopen laitteille yhteiset kommunikaatioon liittyvät parametrit. Tämän alueen objektit kuvaavat muun muassa laitteen tyypin ja sen toimintoja sekä signaloivat virheistä ja laitteen tilasta. Niillä konfiguroidaan myös synkronointiobjektin, poikkeussanomien, aikaleimaobjektin sekä node guarding tai Heartbeat-palveluiden parametrit. Lisäksi valmistaja voi määritellä laitteen nimen, laitteistoversion ja ohjelmistoversion merkkijonoina sekä laitteen valmistajanumeron, laitenumeron, versionumeron ja sarjanumeron. Alueen objekteihin kirjoittamalla voidaan myös tallentaa objektikirjaston sisältämiä parametreja laitteen haihtumattomaan muistiin ja palauttaa laitteen oletusasetukset. On myös mahdollista tallentaa EDS-tiedosto laitteen muistiin, jolloin sen pystyy lukemaan objektista 0x1021. Alueelta löytyy myös objekteja, joiden avulla voidaan luoda komentopohjainen käyttöliittymä solmun ohjaamiseen tai vianetsimiseen väylän kautta. Alueen loppupuolen objekteissa määritellään SDO- ja PDO-parametrit. [2, s. 50; 8, s. 95–147.]

Kaikkia alueen objekteja ei ole pakko toteuttaa. Alla on listattu kaikille CANopen solmuille pakolliset kommunikaatioparametrit:

Objekti 0x1000 sisältää parametrin *laitetyypin*. Se on UNSIGNED32-arvo, joka koostuu kahdesta 16-bittisestä kentästä. Ensimmäinen kenttä määrittelee laite- tai sovellusprofiilin ja toinen kenttä antaa mahdollisia lisätietoja laitteen toiminnallisuudesta. Lisätietokenttä on laite- ja sovellusprofiilikohtainen, eikä sitä ole määritelty CANopen-standardissa. [8, s. 95–96.]

Objekti 0x1001 sisältää *virherekisterin.* Se on UNSIGNED8-tyyppinen arvo, joka ilmoittaa sattuneesta virheestä. Taulukko 7 esittää virherekisterin rakenteen.

Taulukko 7. Virherekisterin virhekoodit [8, s. 96].

Bitti	**Virheen syy**
0	Yleinen virhe
1	Virta
2	Jännite
3	Lämpötila
4	Kommunikaatiovirhe
5	Laiteprofiilikohtainen virhe
6	Varattu (aina 0)
7	Valmistajakohtainen virhe

Ainoastaan bitti 0, yleinen virhe, on pakko toteuttaa. Bitin tila 1 tarkoittaa virhetilaa. Yleinen virhe -bitti tulee asettaa kaikissa virhetilanteissa. [8, s. 96–97.]

Objektilla 0x100C säädetään Node Guarding -protokollan *valvonta-aika*. Solmujen on tuettava joko Node Guarding, tai Heartbeat -protokollaa. Valvonta-aika on UNSIGNED16-tyyppinen arvo, joka määrittelee millisekunteina, kuinka usein isäntäsolmu lähettää Node Guarding -pyynnön tai kuinka usein vastaanottajan pitää se saada. Arvo 0 poistaa Node Guarding -protokollan käytöstä. Objekti on pakollinen ainoastaan, jos solmu käyttää Node Guarding -protokollaa. [8, s. 102.]

Objekti 0x100D määrittelee Node Guarding -protokollan *elinaikakertoimen*. Se kertoo, kuinka monta kertaa valvonta-aika saa kulua ilman Node Guarding -viestien lähettämistä tai vastaanottamista ennen kuin solmu menee virhetilaan. Arvo 0 poistaa Node Guarding -protokollan käytöstä. Objekti on pakollinen ainoastaan, jos solmu käyttää Node Guarding -protokollaa. [8, s. 103.]

Objekti 0x1017 sisältää *Heartbeat-tuottajan aikavälin.* Se määrittelee, kuinka usein solmu lähettää Heartbeat-viestin. Objektin tietotyyppi on UNSIGNED16 ja arvo määrittelee Heartbeat-viestien lähetysvälin millisekunteina. Arvo 0 poistaa Heartbeat-viestien lähettämisen käytöstä. Tämä objekti ja Heartbeat-viestien lähettäminen on pakollista, jos solmu ei käytä Node Guarding -protokollaa. [8, s. 112.]

Objekti 0x1018 on *tunnisteobjekti.* Objekti voi sisältää neljä UNSIGNED32-tyyppistä alaindeksiä, mutta vain ensimmäinen niistä on pakollinen. Alaindeksi 1 sisältää CAN in

Automationin antaman valmistaja-id:n. Alaindeksit 2...4 sisältävät valmistajan määrittelemän tuotekoodin, versionumeron ja sarjanumeron. [8, s. 113–114.]

Valmistajakohtainen alue ja standardisoitujen profiilien alueet

Objektit indeksialueella 0x2000...0x5FFF on jätetty täysin määrittelemättä. Niihin voi toteuttaa laitteen valmistajan sovelluskohtaisia toimintoja, joita ei ole määritelty missään CiA-standardissa.

Indeksialue 0x6000...0x9FFF sisältää standardisoiduissa laite- tai sovellusprofiileissa määritellyt objektit. Nämä objektit voivat olla parametreja, niitä voidaan käyttää kuvaamaan laitteen toiminnallisuutta tai ohjaamaan laitteen toimintaa. CANopen-laite voi sisältää yhdestä kahdeksaan loogista laitetta. Jokaiselle loogiselle laitteelle on määritelty oma alue standardisoitujen profiilien alueesta. [8, s. 88.]

Verkon muuttujien ja systeemimuuttujien alueet

Objektit alueella 0xA000...0xAFFF on tarkoitettu ohjelmoitavan CANopen-laitteen sisääntulojen ja ulostulojen kuvaamiseen ja ohjaamiseen. Alueen objekteja ei ole määritelty CANopen-standardissa.

Indeksialue 0xB000...0xBFFF on varattu CANopen-väylien välisten siltojen muuttujille. Alueen objekteja ei ole määritelty CANopen-standardissa. [8, s. 88.]

3.4 SDO-protokolla

Service data object eli SDO-protokolla tarjoaa suoran pääsyn solmun objektikirjaston merkintöihin. Jokainen solmu sisältää objektikirjaston lisäksi SDO-palvelimen, joka käsittelee luku- ja kirjoituspyynnöt, jotka kohdistuvat sen objektikirjastoon. SDO-tiedonsiirron aloittaa ja sitä johtaa aina asiakas, palvelin vain vastaa asiakkaan pyyntöihin. [8, s. 39.]

CANopen-standardi määrittelee väylän isäntäsolmulle yhden kommunikaatiokanavan jokaista orjasolmua kohden. Isäntä toimii orjasolmujen asiakkaana. Asiakkaalta palvelimelle päin lähetettävissä kehyksissä käytetään kehyksen tunnistetta 0x600 + solmutunniste ja toiseen suuntaan tunnistetta 0x580 + solmutunniste. SDO-

kommunikaatiokanavia voidaan määritellä lisää sovellus-, laite- tai valmistajakohtaisessa profiilissa. Niiden määrittelemiseen on myös varattu objektikirjastosta alue 0x1200…0x127F. Näiden objektien toteuttaminen ei ole pakollista, mutta ne on yleensä määritelty ainakin EDS-tiedostossa tai muussa dokumentaatiossa. [2, s. 62]

SDO-viestien pituus on aina 8 tavua, vaikka siirrettävä tietomäärä ei tätä vaatisikaan [2, s. 63].

SDO-alustusviesti

SDO-tiedonsiirto alkaa aina alustusviestillä, jolla asiakas kertoo palvelimelle, että haluaa lukea tai kirjoittaa palvelimen objektikirjastoon. Jos siirrettävän datan määrä on neljä tavua tai vähemmän, se voidaan siirtää alustusviestissä. Neljää tavua suuremmat datamäärät siirretään seitsemän tavun segmenteissä tai 127 segmentin lohkoissa. Lohkosiirtoprotokollaa ei käsitellä tässä dokumentissa. Sen määritelmä löytyy CANopen-standardista. Kuva 10 esittää alustusviestin rakenteen. [8, s. 39.]

cs 7…5	x 4	n 3…2	e 1	s 0	index		sub	d	
0					1	2	3	4	7

Kuva 10. SDO-protokollan alustusviestin, vastausviestin ja keskeytysviestin rakenne

Alustusviesti on kahdeksan tavua pitkä, ja sen vähiten merkitsevä tavu määrittelee käytettävään tiedonsiirtoprotokollaan liittyvät parametrit.

Bitit 5...7 sisältävä kenttä on komentotarkenne *cs*. Se määrittelee käytettävän tiedonsiirtoprotokollan. Arvo 1 tarkoittaa, että kyseessä on siirto asiakkaalta palvelimelle, arvo 2 sitä, että kyseessä on lataus palvelimelta asiakkaalle. Jälkimmäisessä tapauksessa bittikentät s, e ja n eivät ole käytössä ja niiden arvo on 0.

Bitti *x* ei ole käytössä, ja se on aina 0.

Bittikenttä n määrittelee datan pituuden jos e on 1 ja s on 1, muuten se on 0. Arvo osoittaa d-kentässä olevien dataa sisältämättömien tavujen määrän. Tavut 8-n…7 eivät sisällä dataa.

Bitti *e* määrittelee tiedonsiirtotyypin. Jos se on 0, kyseessä on segmentoitu tiedonsiirto, muuten data siirretään kokonaisuudessaan aloitusviestissä.

Bitti *s* on 1 jos datan pituus on määritelty, muuten se on 0.

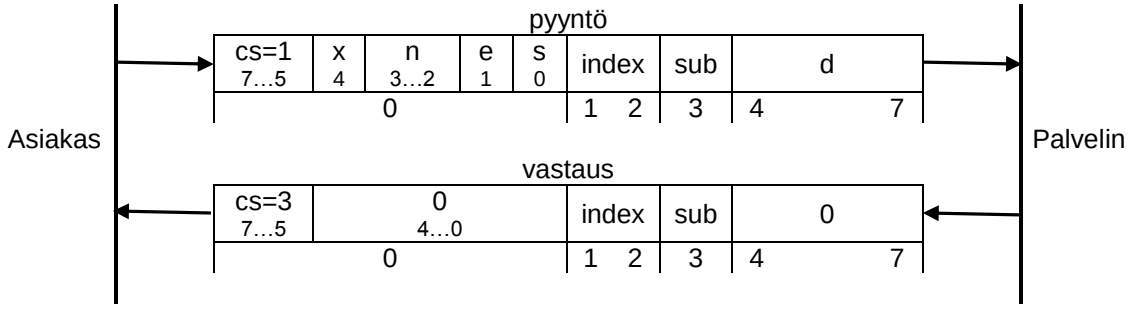
Alustusviestin tavut 1…2 osoittavat kirjoitettavan tai luettavan objektin indeksin ja tavu 3 objektin alaindeksin.

Alustusviestin *d-kenttä* sisältää siirrettävän datan jos e on 1. Muutoin kentän sisältö kertoo siirrettävien segmenttien määrän. Jos cs on 2, eli kyseessä on lataus palvelimelta asiakkaalle, d-kenttä ei ole käytössä ja sen arvo on 0. [8, s. 52–55.]

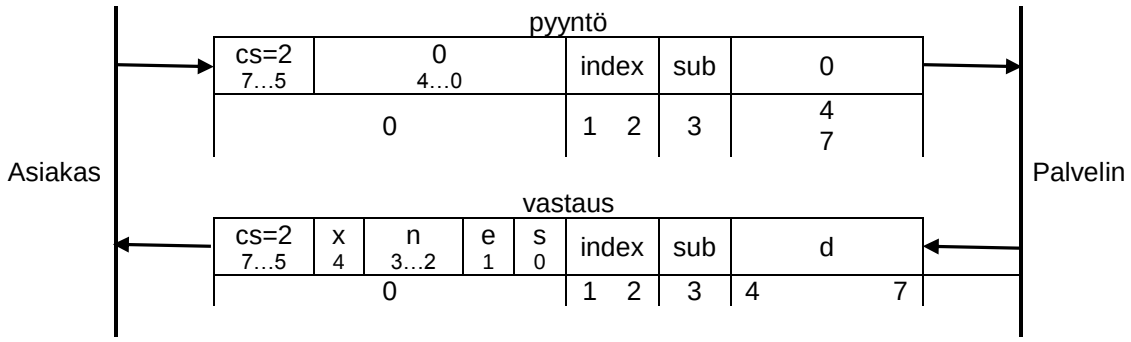
SDO-vastausviesti

Palvelin vastaa aina alustusviestiin. Vastausviestin ja alustusviestin rakenne on sama ja näkyy kuvassa 10. Jos kyseessä on siirto asiakkaalta palvelimelle, vastausviestin komentotarkenne on 3 ja bittikentät s, e, n, ja d eivät ole käytössä ja niiden arvo on 0. Jos kyseessä on lataus palvelimelta asiakkaalle, vastausviestin komentotarkenne on 2. Muut bittikentät ovat samat kuin alustusviestissä kappaleessa 0. [8, s. 39.]

Kuva 11 esittää vielä alustusviestin ja vastausviestin, kun halutaan siirtää dataa asiakkaan suunnasta palvelimen objektikirjastoon ja kuva 12 kun siirto tapahtuu toiseen suuntaan.



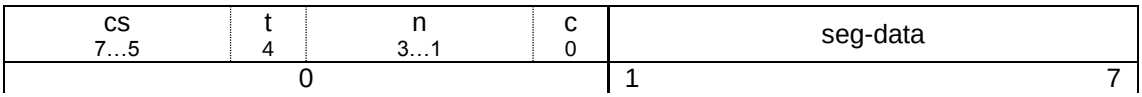
Kuva 11. SDO siirron alustus ja vastaus, siirto asiakkaalta palvelimelle [8, s. 52].



Kuva 12. SDO siirron alustus ja vastaus, siirto palvelimelta asiakkaalle [8, s. 55].

Segmenttien siirto

Kun asiakas on lähettänyt alustusviestin, jossa bitti e on nolla ja palvelin on vastannut viestiin, tai palvelin on vastannut lukupyyntöön viestillä, jossa e on nolla, voidaan aloittaa data-segmenttien siirtäminen [8, s. 39]. Kuva 13 esittää segmenttiviestin rakenteen.



Kuva 13. SDO-segmentin siirtoviestin ja vastausviestin rakenne.

Segmenttisiirron pyyntöviestin komentotarkenne *cs* on 0, kun kyseessä on siirto asiakkaalta palvelimelle ja 3, kun kyseessä lataus palvelimelta asiakkaalle. Jälkimmäisessä tapauksessa bittikentät *c*, *n* ja *seg-data* eivät ole käytössä ja niiden arvo on 0.

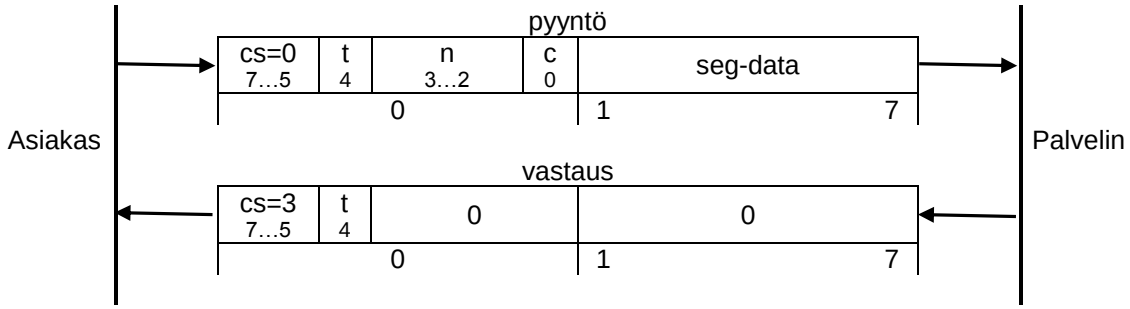
Bitti *c* ilmaisee onko tämän segmentin jälkeen tulossa lisää segmenttejä. Arvo on 0, jos lisää on tulossa, muuten 1.

Bitti *n* kertoo datan pituuden kentässä *seg-data*. Tavut 8-n…7 eivät sisällä dataa.

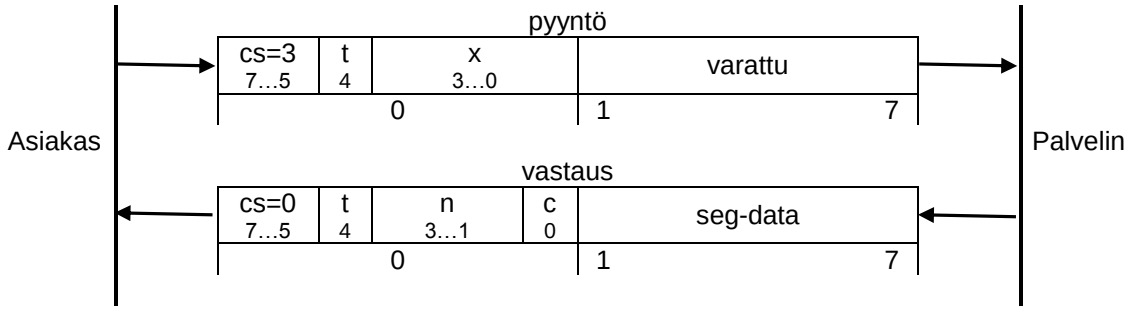
Bitti *t* on ensimmäisessä viestissä 0, sen jälkeen joka toisessa viestissä 1. Pyyntöviestin ja siihen vastaavan viestin t-bittien arvojen tulee olla samat. [8, s. 53–56.]

SDO-palvelin vastaa myös segmenttisiirron pyyntöviesteihin aina. *Segmenttisiirron vastausviestin* komentotarkenne cs on 1, kun kyseessä on siirto asiakkaalta palvelimelle. Tässä tapauksessa bittikentät *c*, *n*, ja *seg-data* eivät ole käytössä ja niiden arvo on 0. Kun komentotarkenne on 0, on kyseessä lataus palvelimelta asiakkaalle. Tällöin bittikenttien sisältö on sama kuin edellisen kappaleen pyyntöviestissä. Bitin t arvo on molemmissa tapauksissa sama kuin pyyntöviestissä. [8, s. 53–56.]

Kuva 14 esittää yhden segmentin viestiliikennettä, kun tiedonsiirtosuunta on asiakkaalta palvelimelle päin ja kuva 15 kun suunta on vastakkainen.



Kuva 14. SDO-segmentin siirto asiakkaalta palvelimelle [8, s. 53].



Kuva 15. SDO-segmentin siirto palvelimelta asiakkaalle [8, s. 56].

SDO-siirron keskeytys

Jos SDO-tiedonsiirto epäonnistuu, lähettää palvelin SDO-siirron keskeytysviestin, jonka rakenne vastaa alustusviestiä ja näkyy kuvassa 10. Viestin komentotarkenne on 4 ja d-kenttä sisältää 4-tavuisen keskeytyskoodin, joka kertoo keskeytyksen tarkemman syyn. Viestin index- ja sub-kentät sisältävät osoitteen objektiin, jota yritettiin lukea tai kirjoittaa. Muut kentät eivät ole käytössä ja niiden arvo on 0. [8, s. 65.] Taulukossa 8 on listattu kaikki CANopen-standardin määrittelemät SDO-siirron keskeytyskoodit.

Taulukko 8. SDO-siirron keskeytyskoodit [8, s. 65–66].

Keskeytyskoodi	Selitys
0503 0000	t-bitti ei vaihtunut
0504 0000	SDO protokollan aikakatkaisu
0504 0001	Tuntematon komentotarkenne
0504 0002	Epäkelpo lohkon koko (lohkosiirto)
0504 0003	Epäkelpo järjestysnumero (lohkosiirto)
0504 0004	CRC-virhe (lohkosiirto)
0504 0005	Muisti loppu
0601 0000	Ei tuettu objektitoiminto
0601 0001	Yritys lukea objektia, johon vain kirjoitusoikeus
0601 0002	Yritys kirjoittaa objektiin, johon vain lukuoikeus
0602 0000	Objektia ei ole objektikirjastossa
0604 0041	Objektia ei voi kuvata PDO:hon
0604 0042	Kuvattavien objektien määrä ja pituus ylittävät PDO:n pituuden
0604 0043	Yleinen parametrin yhteensopivuusvirhe
0604 0047	Yleinen laitteen sisäinen yhteensopivuusvirhe
0606 0000	Laitevirhe
0607 0010	Tietotyyppi ei täsmää, parametrin pituus ei täsmää
0607 0012	Tietotyyppi ei täsmää, parametrin pituus liian suuri
0607 0013	Tietotyyppi ei täsmää, parametrin pituus liian pieni
0609 0011	Alaindeksiä ei ole olemassa
0609 0030	Epäkelpo parametrin arvo
0609 0031	Parametrin arvo liian suuri
0609 0032	Parametrin arvo liian pieni
0609 0036	Huippuarvo on pienempi kuin vähimmäisarvo
060A 0023	Resurssi ei ole käytettävissä: SDO-yhteys
0800 0000	Yleinen virhe
0800 0020	Tietoa ei pystytä siirtämään tai tallettamaan sovellukseen
0800 0021	Tietoa ei pystytä siirtämään tai tallettamaan sovellukseen paikallisesta ohjauksesta johtuen.
0800 0022	Tietoa ei pystytä siirtämään tai tallettamaan sovellukseen laitteen tilasta johtuen.
0800 0023	Objektikirjaston dynaaminen luominen epäonnistui tai objektikirjastoa ei ole.
0800 0024	Dataa ei ole käytettävissä.

3.5 PDO-protokolla

CANopen-standardi määrittelee toisenkin protokollan tiedonsiirtoon solmujen välille. Process data object eli PDO-protokolla on tarkoitettu nimensä mukaisesti prosessidatan siirtoon. Tämä tarkoittaa sitä, että toisin kuin SDO-protokollassa, asiakkaan ei tarvitse erikseen pyytää palvelimelta tietoja, vaan PDO-viestien tuottaja lähettää viestejä automaattisesti joko tietyin aikavälein tai tapahtumien laukaisemana. [8, s. 33.]

3.5.1 Transmit-PDO

PDO-viestien sisältö ja kommunikaatioparametrit määritellään viestejä tuottavan solmun objektikirjastossa. Kommunikaatioparametrit määritellään indeksialueella 0x1800…0x19FF ja kuvaus, eli viestien sisältö, indeksialueella 0x1A00…0x1BFF. Yhden PDO:n asetukset koostuvat yhdestä kommunikaatioparametriobjektista ja yhdestä kuvausobjektista. Ensimmäisen PDO:n asetukset löytyvät objekteista 0x1800 ja 0x1A00, toisen objekteista 0x1801 ja 0x1A01 ja niin edelleen.

TPDO-kommunikaatioparametrit

Taulukko 9. TPDO- ja RPDO-kommunikaatioparametriobjektin rakenne.

Alaindeksi	**Nimi**	**Tietotyyppi**
0	Alaindeksien määrä	UNSIGNED8
1	COB-ID	UNSIGNED32
2	Tiedonsiirtotyyppi	UNSIGNED8
3	Estoaika	UNSIGNED16
4	Ei käytössä	UNSIGNED8
5	Tapahtuma-aika	UNSIGNED16

Taulukossa 9 on esitetty TPDO-kommunikaatioparametriobjektin rakenne. Objektin alaindekseistä ainoastaan 0, 1 ja 2 ovat pakollisia.

Alaindeksi 1, *COB-ID* (Connection object identifier) määrittelee PDO-viestin CAN-tunnisteen. Lisäksi COB-ID:n bitti 31 määrittelee, onko PDO käytössä vai ei. Arvo 0 tarkoittaa, että PDO on käytössä.

Taulukko 10. PDO-viestin tiedonsiirtotyypit [8, s. 139].

Tiedonsiirtotyyppi	**Selitys**
0	PDO on synkroninen. Se minkä SYNC-objektin jälkeen PDO-viesti lähetetään, määritellään laiteprofiilissa.
1…240	PDO on synkroninen. PDO-viesti lähetetään *n* SYNC-objektin jälkeen, missä *n* = tiedonsiirtotyyppi. Esimerkiksi jos tiedonsiirtotyyppi on 5, lähetetään PDO-viesti joka viidennen SYNC-objektin jälkeen.
241…251	Ei käytössä
252	PDO:n data päivitetään SYNC-objektin jälkeen, mutta PDO-viestiä ei lähetetä. Viesti lähetetään vain jos sitä pyydetään RTR-kehyksellä.
253	PDO:n data päivitetään ja PDO-viesti lähetetään vain jos sitä pyydetään RTR-kehyksellä.
254	PDO on tapahtumaliipaistu.(valmistajakohtainen)
255	PDO on tapahtumaliipaistu.(laite- tai sovellusprofiilikohtainen)

Alaindeksi 2 määrittelee PDO-viestin *tiedonsiirtotyypin*. Tiedonsiirtotyypit on listattu taulukossa 10. Taulukossa mainittu SYNC-objekti määritellään luvussa 3.9.

Estoaika määrittelee vähimmäisajan tapahtumaliipaistujen PDO-viestien välillä. Estoaika ilmaistaan 100 ms monikertoina.

Alaindeksi 4 on jäänne aikaisemmista CANopen-standardin versioista, eikä sitä tule käyttää.

Tapahtuma-aika määrittelee ajan millisekunteina, minkä välein PDO-viesti lähetetään.

TPDO-kuvaus

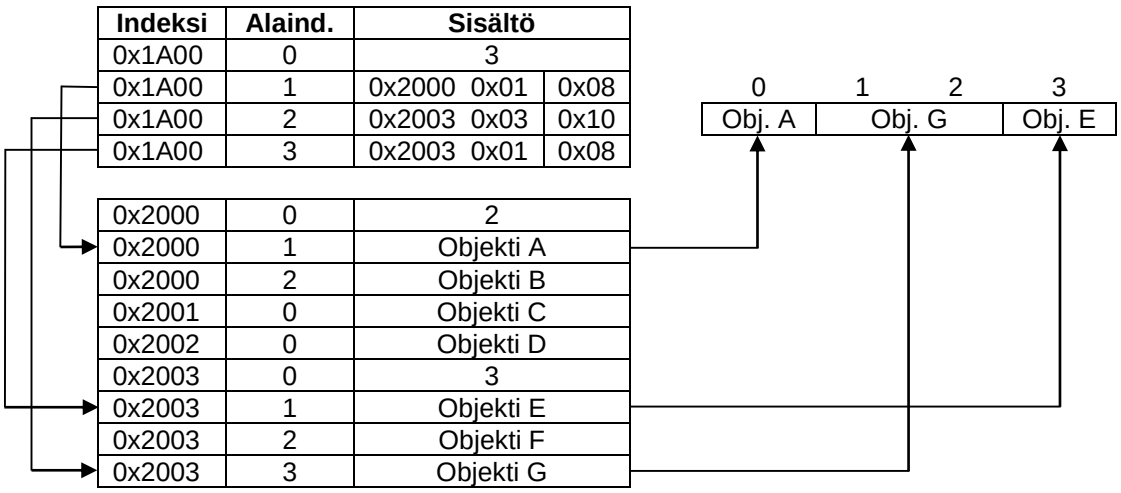
PDO-viestien sisältö koostuu PDO-tuottajan objektikirjaston merkinnöistä. Yksi PDO-viesti voi sisältää 8 tavua, eli 64 bittiä dataa. Tähän datakenttään voidaan kuvata enintään 64 objektikirjaston merkintää, jos jokainen niistä on yhden bitin pituinen. Yhden merkinnän kuvausparametri on 32-bittinen arvo, joka on jaettu kolmeen kenttään. Kuva 16 esittää kuvausparametrin rakenteen.

31 16	15 8	7 0
Indeksi	Alaindeksi	Pituus

Kuva 16. PDO-kuvausparametrin rakenne [8, s. 142].

Parametri sisältää kuvattavan merkinnän indeksin, alaindeksin ja pituuden bitteinä. Yhden PDO:n kuvausobjekti koostuu listasta tällaisia kuvausparametreja. kuvausobjektin ensimmäinen alaindeksi määrittelee kuvattavien merkintöjen määrän ja seuraavat alaindeksit 1…64 itse kuvausparametrit.

Kuvassa 17 on esimerkki kolmen valmistajakohtaisen objektikirjastomerkinnän kuvaamisesta PDO-viestiin. Kuvausobjektin alaindeksi 0 kertoo, että kuvattavia parametreja on kolme. Alaindeksi 0x01 määrittelee, että viestin ensimmäiseen tavuun kuvataan objektin 0x2000 alaindeksi 0x01. Alaindeksin 0x02 mukaan seuraaviin kahteen tavuun kuvataan objektin 0x2003 alaindeksi 0x03. Alaindeksin 0x03 mukaan neljänteen tavuun kuvataan objektin 0x2003 alaindeksi 0x01. Toisin kuin SDO-viesti, PDO-viesti voi olla lyhyempi kuin 8 tavua, jos kuvattavia merkintöjä on alle kahdeksan tavun verran. [8, s. 137–144.]



Kuva 17. Esimerkki kolmen objektikirjastomerkinnän kuvaamisesta PDO-viestiin [8, s. 143].

3.5.2 Receive-PDO

PDO-viestejä kuluttavan solmun objektikirjastossa indeksialueella 0x1400…0x15FF määritellään vastaanotettavien PDO-viestien kommunikaatioparametrit ja indeksialueella 0x1600…0x17FF kuvaus eli se, mihin objektihakemiston merkintöihin viestit puretaan [8, s. 131–137].

RPDO-kommunikaatioparametrit

Receive-PDO eli RPDO-kommunikaatioparametriobjektin rakenne on sama kuin TPDO:lla ja se näkyy taulukossa 9. Objektin alaindekseistä ainoastaan 0, 1 ja 2 ovat pakollisia.

Alaindeksi 1 sisältää *COB-ID-parametrin*, joka määrittelee vastaanotettavan viestin CAN-tunnisteen. Jotta linkitys PDO-tuottajan ja -kuluttajan välillä toimisi, tulee kuluttajan RPDO-parametri COB-ID vastata tuottajan TPDO-parametria COB-ID.

Alaindeksi 2 sisältää jälleen *tiedonsiirtotyypin*, joka määrittelee käsitelläänkö vastaanotettu data heti vai aloitetaanko käsittely vasta SYNC-objektin vastaanottamisen jälkeen. Jos data käsitellään heti, tiedonsiirtotyyppi on 0xFE tai 0xFF, jos taas odotetaan SYNC-objektia, tiedonsiirtotyyppi on 0x00…0xF0.

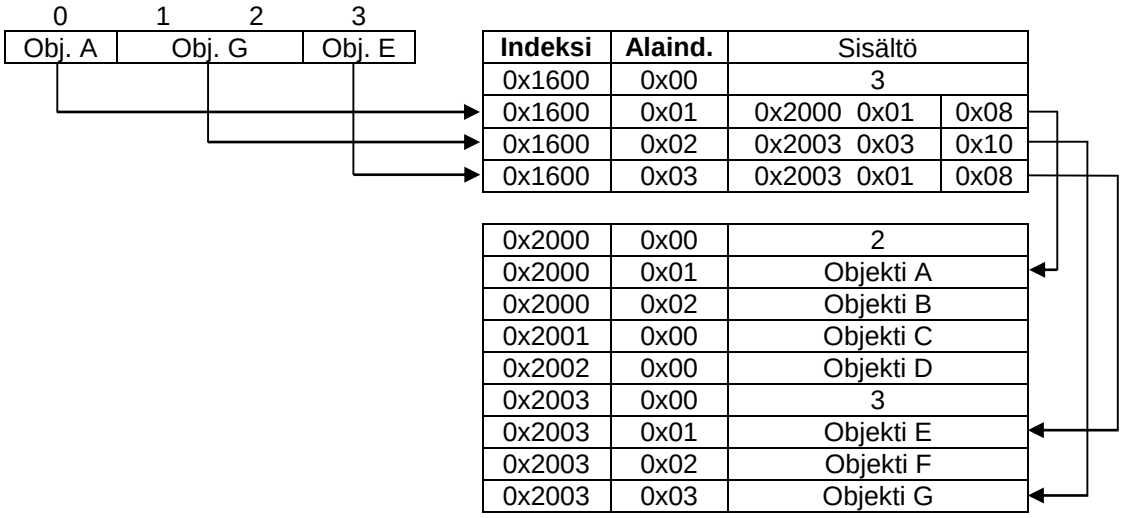
Estoaikaa alaindeksissä 3 ei RPDO:n konfiguroinnissa käytetä. Jos se on olemassa RPDO-kommunikaatioparametriobjektissa, arvon tulee olla 0.

Tapahtuma-aika-parametria voidaan käyttää siten, että solmu indikoi virheen, mikäli PDO-viestiä ei vastaanoteta säädetyn ajan kuluessa.

RPDO-kuvaus

PDO-viestin vastaanottamisen jälkeen viestin sisältämät merkinnät puretaan vastaanottavan solmun objektikirjastoon. RPDO:n kuvaussobjekti vastaa rakenteeltaan TPDO:n kuvaussobjektia, eli se on lista kuvausparametreja, joiden rakenne on esitetty kuvassa 16.

Kuvassa 18 on esimerkki PDO-viestin kuvaamisesta vastaanottavan solmun objektikirjastoon. Kuvausobjektin alaindeksi 0 kertoo, että kuvattavia objekteja on kolme. Alaindeksi 0x01 määrittelee, että PDO-viestin ensimmäinen tavu puretaan objektin 0x2000 alaindeksiin 0x01. Alaindeksin 0x02 mukaan seuraavat kaksi tavua puretaan objektin 0x2003 alaindeksiin 0x03. Alaindeksin 0x03 mukaan viestin viimeinen tavu puretaan objektin 0x2003 alaindeksiin 1.



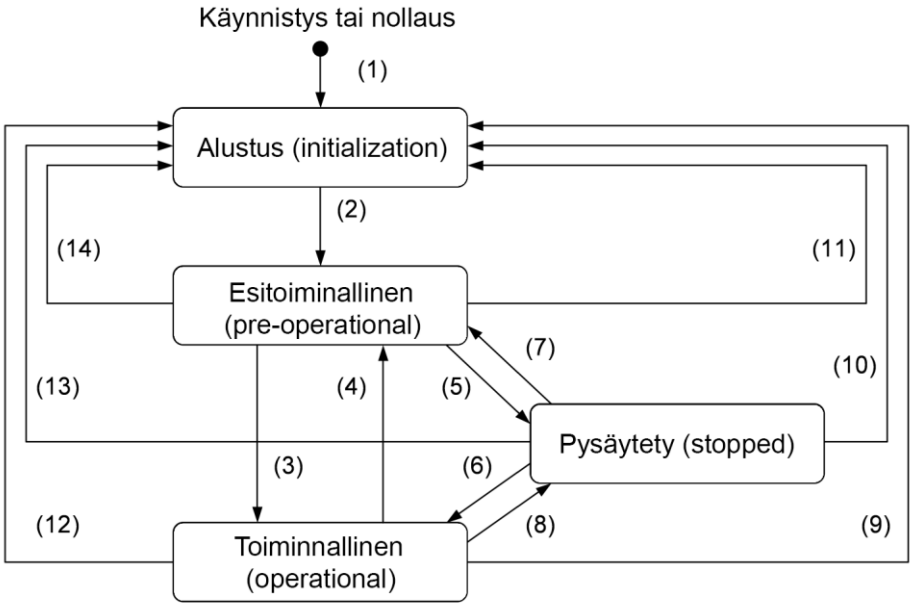
Kuva 18. Esimerkki PDO-viestin kuvaamisesta PDO-kuluttajan objektikirjastoon [8, s. 136].

Jos vastaanotetussa PDO-viestissä on enemmän dataa kuin RPDO-kuvausobjektissa on määritelty, käytetään viestistä ne tavut, jotka on kuvattu, ja jätetään loput huomioimatta. Vastaanottava solmu voi tässä tapauksessa lähettää EMCY-viestin, mutta se ei ole pakollista. Jos vastaanotetussa viestissä on liian vähän dataa, jätetään data käsittelemättä. EMCY-protokollaa tukevan solmun tulee tässä tapauksessa lähettää EMCY-viesti vikakoodilla 0x8210. [8, s. 131–137.]

3.6 Verkonhallinta

CANopen-standardi määrittelee verkonhallintaprotokollan NMT (Network management). Verkossa on yksi NMT-isäntä. Muut solmut toimivat NMT-orjina, joita isäntä ohjaa NMT-viesteillä. Isäntä ja orjat tunnistetaan verkossa uniikilla solmun tunnistenumerolla, joka voi olla väliltä 1…127.

Jokainen orjasolmu sisältää NMT-tilakoneen, joka koostuu tiloista alustus (initialization), esitoiminnallinen (pre-operational), toiminnallinen (operational) ja pysäytetty (stopped). Tilat eroavat toisistaan sillä, mihin toimintoihin liittyviä viestejä solmu saa lähettää ollessaan kyseisessä tilassa. Kuvassa 19 on kaavio solmun NMT-tilakoneesta.



(1)	Käynnistyksen yhteydessä alustustilaan siirrytään automaattisesti.
(2)	NMT alustustilan jälkeen siirrytään automaattisesti tilaan esitoiminnallinen.
(3)	NMT solmun käynistys -viesti tai paikallinen ohjaus
(4),(7)	NMT solmun siirto tilaan esitoiminallinen -viesti
(5),(8)	NMT solmun pysäytys -viesti
(6)	NMT solmun käynistys -viesti
(9),(10),(11)	NMT solmun täydellinen nollaus -viesti
(12),(13),(14)	NMT solmun kommunikaation nollaus -viesti

Kuva 19. CANopen-solmun NMT-tilakone [8, s. 83].

NMT-isäntä hallitsee orjien tilakoneiden tiloja NMT-solmujenhallintaviesteillä. Jotkut tietyt tilanvaihdokset voivat myös tapahtua ilman isännän määräystä. Solmunhallintaviestien tunniste on aina 0, ja ne sisältävät kaksi tavua dataa, joista ensimmäinen sisältää komentotarkenteen ja toinen ohjattavan orjasolmun solmutunnisteen. Jos viestin solmutunniste-tavu on 0, on viesti kohdistettu kaikille väylän solmuille. Solmunhallintaviestien komentotarkenteet on listattu taulukossa 11.

Taulukko 11. NMT-solmunohjausviestit.

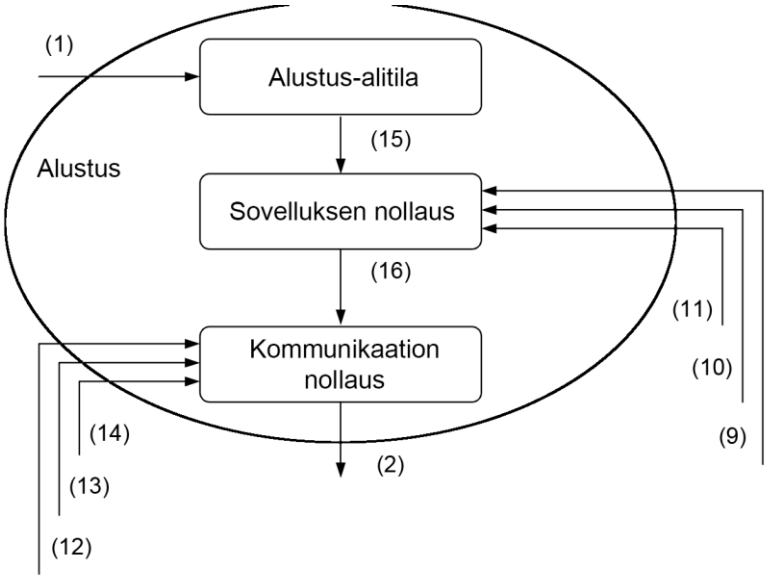
Komentotarkenne	**Toiminto**
0x01	Solmun käynnistys, eli siirto toiminnalliseen tilaan
0x02	Solmun pysäytys
0x80	Solmun siirto tilaan esitoiminnallinen
0x81	Solmun sovelluksen nollaus
0x82	Solmun kommunikaatioparametrien nollaus

Kun solmu käynnistyy tai se nollataan, se menee ensin *alustustilaan*, joka sisältää kolme alitilaa. *Alustus*-alitilassa alustetaan solmun laitteisto. *Sovelluksen nollaus* -alitilassa nollataan objektikirjaston valmistajakohtaisen ja laiteprofiilikohtaisen alueen parametrit. *Kommunikaation nollaus* -alitilassa nollataan objektikirjaston kommunikaatioprofiilialueen parametrit. Tämän jälkeen solmu yrittää lähettää boot-up-viestiä, jolla se kertoo muille solmuille alustuksen olevan valmis. Boot-up-viesti on Heartbeat-viestin erikoistapaus. Sen tunniste on 0x700 + lähettävän solmun solmutunniste, ja se sisältää yhden tavun dataa, jonka arvo on 0. Kun viesti on lähetetty onnistuneesti, solmu siirtyy esitoiminnalliseen tilaan. Alustus-tilan tilakaavio näkyy kuvassa 20.

Esitoiminnallisessa tilassa solmu saa ottaa osaa SDO-, EMCY-, SYNC-, TIME ja Heartbeat- tai Node Guarding -protokolliin liittyvään tietoliikenteeseen. Solmu voi siirtyä esitoiminnallisesta tilasta toiminnalliseen tilaan vastaanotettuaan NMT-isännältä solmun käynnistysviestin eli NMT-viestin, jonka komentotarkenne on 1.

Toiminnallisessa tilassa solmu saa edellisten lisäksi ottaa osaa myös PDO-tiedonsiirtoon.

NMT-isäntä voi asettaa esitoiminnallisessa tai toiminnallisessa tilassa olevan solmun *pysäytetty-tilaan* lähettämällä ohjausviestin komentotarkenteella 2. Pysäytetty solmu ei saa ottaa mitään osaa väylän tietoliikenteeseen lukuun ottamatta NMT-protokollaa.



(1)	Käynnistyksen yhteydessä alustus-alitilaan siirrytään automaattisesti.
(2)	NMT alustustilan jälkeen siirrytään automaatisesti tilaan esitoiminnallinen
(12), (13), (14)	NMT solmun kommunikaation nollaus -viesti
(9), (10), (11)	NMT solmun täydellinen nollaus -viesti
(15)	NMT alustus-alitila valmis – siirrytään automaatisesti alitilaan sovelluksen nollaus
(16)	NMT sovelluksen nollaus -alitila valmis – siirrytään automaattisesti alitilaan kommunikaation nollaus

Kuva 20. NMT-alustustilan alitilat [8, s. 84].

NMT-isäntä voi nollata esitoiminnallisessa, toiminnallisessa tai pysäytetyssä tilassa olevan solmun lähettämällä NTM-viestin, jossa on sopiva komentotarkenne. Komentotarkenteella 129 solmu nollataan täysin. Komentotarkenteella 130 nollataan ainoastaan kommunikaatioparametrit. [8, s. 83–85.]

3.7 Virheenhallintaprotokollat

CANopen verkossa solmut voivat tarkkailla toistensa tilaa virheenhallintaprotokollien avulla. Protokollia on kahta tyyppiä, joista toinen on standardin mukaan pakko toteuttaa.

Node guarding -protokollassa NMT-isäntä pyytää RTR-viestillä valvottavia orjasolmuja lähettämään viestin jos ne ovat elossa. Mikäli solmu ei ole vastannut määritellyn ajan

sisällä NMT-isäntä tietää, että solmussa on tapahtunut virhe. Sama protokolla toimii myös toiseen suuntaan, eli orjasolmut voivat valvoa NMT-isäntää. Tätä kutsutaan life guarding -protokollaksi. [8, s. 79.] Node guarding- ja life guarding -protokollia käytetään nykyään harvemmin, joten niitä ei käsitellä tässä tarkemmin [2, s. 86].

Heartbeat- eli sydämenlyöntiprotokollassa solmut lähettävät Heartbeat-viestejä automaattisesti määrätyin välein. Heartbeat-viestin lähetysaikaväli on määritelty objektikirjaston objektissa 0x1017 [8, s. 112]. Jos valvova solmu havaitsee, ettei valvottava solmu ole lähettänyt Heartbeat-viestiä määrätyn ajan kuluessa, se tietää, että solmussa on tapahtunut virhe. Heartbeat-viestin valvonta-aikaväli märitellään jokaiselle valvottavalle solmulle erikseen valvovan solmun objektikirjaston objektin 0x1016 alaindekseissä [8, s. 111–112].

Hearbeat-viestin tunniste on 0x700 + lähettävän solmun solmutunniste. Viesti sisältää yhden tavun dataa, joka ilmaisee solmun NMT-tilan. NMT-tila voi olla 0 eli alustustila, 4 eli pysäytetty, 5 eli toiminnallinen tai 127 eli esitoiminnallinen. [8, s. 80.]

3.8 EMCY-protokolla

Emergency- eli EMCY-protokollan avulla solmut voivat lähettää virheilmoituksia ja ilmoituksia virhetilojen poistumisesta. EMCY-viestin lähettämisen laukaisee aina laitteen sisäinen virhetapahtuma. Viesti lähetetään kerran per tapahtuma, eikä sitä toisteta, jos sama virhetilanne jatkuu. Virheen poistumisesta ilmoitetaan uudella EMCY-viestillä. [8, s. 69.]

EMCY-viestin tunniste määritellään objektikirjaston objektin 0x1014 biteissä 0…10. CANopen-standardin mukaan tulisi käyttää tunnistetta 0x80 + lähettävän solmun solmutunniste. Saman objektin bitti 31 määrittelee onko EMCY-protokolla käytössä kyseisessä solmussa. Objektissa 0x1015 voidaan määritellä EMCY-viestien lähetysten välinen vähimmäisaika mikrosekunteina. [8, s. 109–111.]

EMCY-viesti sisältää 8 tavua dataa, joista kaksi ensimmäistä tavua, tavut 0 ja 1, määrittelevät virhekoodin. Virhekoodin ylempi tavu määrittelee virheen tyypin ja alemmalla tavulla voidaan antaa tarkempaa tietoa virheestä. [8, s. 69.] Virhetyypit on listattu taulukossa 12.

EMCY-viestin tavu 2 saa arvon solmun virherekisteristä, objektikirjaston objektista 0x1001. Virherekisterin virhekoodit on listattu taulukossa 7. EMCY-viestin tavut 3…7 on varattu valmistajakohtaisille virhekoodeille. [8, s. 69–72].

Taulukko 12. EMCY-viestin virhetyypit [8, s. 69].

Virhekoodi	Selitys
0x00xx	Virhe on poistunut.
0x10xx	Yleinen virhe
0x20xx	Virta
0x21xx	Virta, laitteen sisääntulossa
0x22xx	Virta, laitteen sisällä
0x23xx	Virta, laitteen ulostulossa
0x30xx	Jännite
0x31xx	Verkkojännite
0x32xx	Jännite laitteen sisällä
0x33xx	Jännite laitteen ulostulossa
0x40xx	Lämpötila
0x41xx	Ulkolämpötila
0x42xx	Laitteen lämpötila
0x50xx	Laitteistovirhe
0x60xx	Ohjelmistovirhe
0x61xx	Sisäinen ohjelmisto
0x62xx	Käyttäjän ohjelmisto
0x63xx	Data
0x70xx	Lisämodulit
0x80xx	Monitorointi
0x81xx	Kommunikaatio
0x82xx	Protokollavirhe
0x90xx	Ulkoinen virhe
0xF0xx	Lisäfunktiot
0xFFxx	Laitekohtainen

3.9 SYNC-protokolla

Väylän solmujen tahdistamista PDO-viestien tahdistettua lähettämistä varten yksi solmu voi lähettää määrätyin aikavälein SYNC- eli synkronointiviestejä [8, s. 67]. Aikaväli määritellään mikrosekunteina objektikirjaston objektissa 0x1006. Arvo nolla poistaa SYNC-objektin lähettämisen käytöstä. [8, s. 100.]

SYNC-viestin tunniste määritellään objektikirjaston objektin 0x1005 biteissä 0…10. Tunnisteen oletusarvo on 0x80. Saman objektin bitti 30 määrittelee, onko SYNC-objektin lähetys käytössä kyseisessä solmussa. Arvo nolla poistaa SYNC-objektin lähetyksen käytöstä. Väylässä voi olla vain yksi SYNC-objektin lähettäjä kerrallaan. [8, s. 99.]

SYNC-objekti voi sisältää yhden tavun dataa, jota käytetään laskurina joka kasvaa yhdellä jokaisessa viestissä [8, s. 67].

3.10 TIME-protokolla

TIME-protokollalla voidaan luoda tuottaa aikatieto CANopen-verkkoon. TIME-viestin tunniste on 0x100 ja se sisältää 6 tavua dataa. Data sisältää aikaleiman, jonka tietotyyppi on TIME_OF_DAY. Se on 48-bittinen tietue, joka sisältää UNSIGNED28-arvon "millisekunteja keskiyöstä" ja UNSIGNED16-arvon "päiviä sitten 1.1.1984". Arvojen välissä on 4 bitin määrittelemätön kenttä. [8, s. 68–69.]

4 CiA447

Henkilöautoissa käytetyt CAN-väyläprotokollat ovat yleensä valmistajakohtaisia ja niihin on hyvin vaikea päästä käsiksi. Turvallisuussyistä autovalmistajat ovat vastahakoisia antamaan mitään tietoa auton väylistä. CiA447 on CANopen sovellustason protokollan päälle määritelty sovellusprofiili, jonka avulla autovalmistajat voivat suoda rajoitetun pääsyn auton sisäisiin väyliin. Kaikki tiedonsiirto tapahtuu yhdyskäytävän kautta, joten pääsy voidaan rajoittaa toimintoihin, jotka on määritelty turvallisiksi. [10, s. 4–11.]

CiA447 on CANopen-sovellusprofiili, joka määrittelee avoimen ohjausverkon erikoisajoneuvojen, kuten taksien, hälytysajoneuvojen ja inva-ajoneuvojen lisälaitteille.[8] Sovellusprofiilin kehittämistä johtavat Audi, BMW, Mercedes ja Volkswagen. Ne julkaisivat ensimmäiset versiot IVN-yhdyskäytävästä jo vuonna 2010, mutta toistaiseksi CiA447-sovellusprofiilia on käytetty lähinnä saksalaisissa ja brittiläisissä poliisiautoissa. [11.]

Uuden E-sarjan Mercedes Benzin taksivarusteluun kuuluu auton ohjekirjan mukaan CiA447-yhteensopiva IVN-yhdyskäytävä, ja Volkswagen on ilmoittanut varustavansa uudet Golf 6 autot CiA447-yhteensopivalla yhdyskäytävällä syksyllä 2013. Muut Volkswagen-mallit seuraavat yhtiön mukaan myöhemmin. [12; 13.]

4.1 Fyysinen kerros

Fyysisen kerroksen osalta CiA447 noudattaa standardia ISO 11898-2. Käytetty siirtonopeus on 125 kbit/s. Bittiajoituksen osalta noudatetaan CANopen-standardin suositusta kyseiselle nopeudelle:

- bittiaika 8 µs
- näytteenottopisteen sijainti 85...90 %
- suositeltu näytteenottopiste 87,5 %.

Liittimeksi suositellaan 18-pinnistä VDA-liitintä. Auton puoleinen liitin on naaras ja lisälaiteliitin uros. Sopivat liittimet löytyvät ainakin AMP Micro Quadlok -sarjasta. [14, s. 7.] Alla lista tarvittavista osista ja TE-connectivityn tuotenumerot. Liittimeen sopiva johdonpaksuus on 0,75 m2.(18AWG)

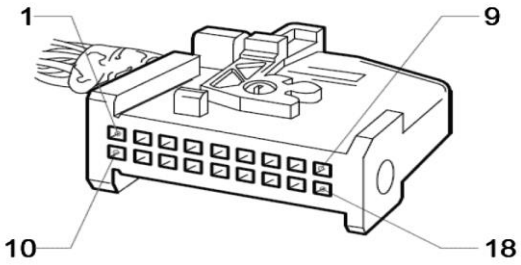
Naarasliitin:

- 1355348-1 liitinrunko
- 1-1355350-1 liitinkotelo
- 963729-1 vastake, irtonainen
- 963715-1 vastake, liuskassa

Urosliitin:

- 1-962692-2 liitinrunko
- 1-962693-1 liitinkotelo
- 962112-3 liitinkotelon kansi
- 963730-1 nasta, irtonainen
- 963716-1 nasta, liuskassa.[11]

Kuvassa 21 on auton puolella oleva 18-pinninen VDA-naarasliitin ja Taulukko 13 liittimen signaalien määrittelyt.



Kuva 21. 18-pinninen VDA-naarasliitin [14, s. 7].

Taulukko 13. 18-pinnisen VDA-liittimen signaalit [14, s. 8].

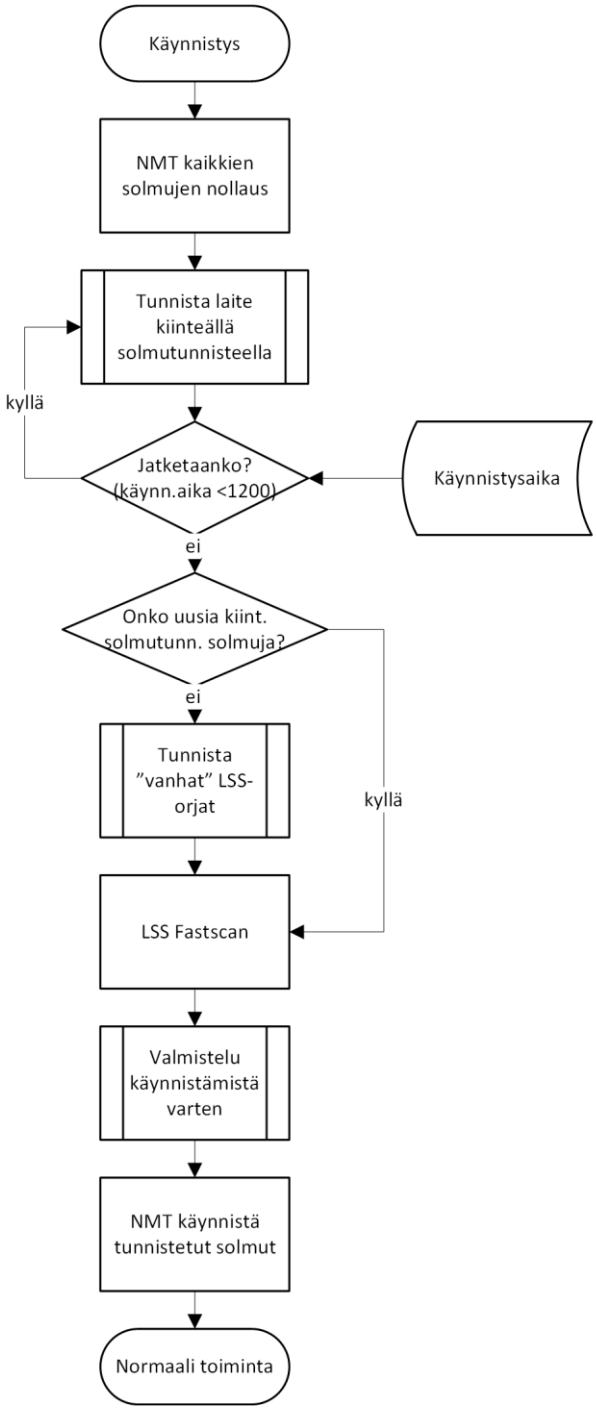
Nasta	Väri	Nimi	Suositeltu käyttö taksissa	Signaalin suunta suhteessa autoon
1	**ruskea**	**KL31**	**Maa (0V / 4A)**	**ulos**
2	ruskea/violetti	PTT	Puhepainike	sisään/ulos
3	vihreä/sininen	S1	Ei käytössä	sisään/ulos
4	violetti	S2	Varattu äänen mykistykselle	sisään/ulos
5	musta/valkoinen	S3	Matkustajan tunnistus	sisään/ulos
6	ruskea/musta	S4	Hätäkutsu	ulos
7	musta/vihreä	AUDIO_OUT+	Kaiutin +	sisään/ulos
8	**keltainen**	**CAN_L**	**CAN-low**	**väylä**
9	musta	AUDIO_IN-	Mikrofoni -	sisään/ulos
10	**musta/sininen**	**KL15**	**Virtalukko**	**ulos**
11	**punainen/keltainen**	**KL30**	**Syöttöjännite (11–16V / 4A)**	**ulos**
12	harmaa/sininen	KL58	Seisontavalojen tila	ulos
13	vihreä/keltainen	SPEED_PULS	Nopeuspulssi	ulos
14	sininen/keltainen	S5	Taksamittarin tila	sisään/ulos
15	sininen/valkoinen	S6	Kattokyltin tilapyyntö	sisään/ulos
16	vihreä	AUDIOGND	Kaiutin -	sisään/ulos
17	**keltainen/musta**	**CAN_H**	**CAN-high**	**väylä**
18	valkoinen	AUDIO_IN/MIC	Mikrofoni +	sisään/ulos

4.2 Solmutunniste (node-ID)

CiA447-profiilin mukaisessa väylässä solmujen enimmäismäärä on 16. Solmujen tunnisteet ovat välillä 1...16. IVN-yhdyskäytävän tunniste on aina 1. Muiden solmujen tunniste voidaan määrittää kolmella tavalla. Suositeltu tapa on käyttää CiA305-standardin mukaista LSS (Layer Setting Services) Fastscan -menetelmää, mutta tunniste voi myös olla kiinteästi määritelty tai ulkoisesti, esimerkiksi kytkimillä tai hyppyjohtimilla määritelty. [14, s. 9.]

4.3 Verkon käynnistys

Kun IVN-yhdyskäytävä käynnistyy, sen sisältämä NMT-isäntä tunnistaa ja käynnistää verkon muut solmut. Yhdyskäytävän käynnistysprosessi näkyy kuvassa 22.



Kuva 22. CiA447-yhteensopivan IVN-yhdyskäytävän käynnistysprosessi [14, s. 11].

Ensin NMT-isäntä nollaa kaikki solmut lähettämällä NMT-viestin, jonka komentotarkenne on 0x81 ja solmutunniste-kenttä on 0. [14, s. 11.]

Kiinteän solmutunnisteen solmujen tunnistaminen

Kun kaikki solmut on nollattu, isäntä tunnistaa solmut, joilla on kiinteä tai ulkoisesti määritelty solmutunniste, yrittämällä lukea objektit 0x1000, 0x6000 ja objektin 0x1018 alaindeksit 0x01…0x04. Tämä tehdään lähettämällä SDO-latauspyyntö jokaiselle mahdolliselle tunnisteelle välillä 2…16. Tätä toistetaan, kunnes solmut vastaavat, jokainen tunniste on skannattu vähintään kahdesti tai kun isännän käynnistyksestä on kulunut 1200 millisekuntia. Jos solmun käynnistys kestää yli 1200 millisekuntia, sen tulee tukea LSS Fastscan -menetelmää. [14, s. 12.]

Vanhojen LSS-orjien tunnistaminen

Kun NMT-isännän käynnistyksestä on kulunut 1200 millisekuntia, se siirtyy seuraavaan vaiheeseen. CiA447-profiilissa suositellaan, että NMT-isäntä tallentaa listan LSS-tunnistetuista orjasolmuista aina kun solmu liitetään väylään tai irrotetaan siitä. Jos solmuja, joilla on kiinteä tunniste, ei löytynyt, tai jos kiinteän tunnisteen solmut ovat samat kuin edellisellä käynnistyskerralla, isäntä käy läpi listan LSS-tunnistetuista orjasolmuista ja selvittää, ovatko samat solmut edelleen liitettynä väylään. [14, s. 12.]

CAN-ID = 0x7E5

cs	IDNumber				varattu (0x000)		
0	1	2	3	4	5	6	7

Kuva 23. LSS-orjan tunnistuskysely

Orjasolmut tunnistetaan siten, että isäntä lähettää kuvan 23 mukaisia kyselyviestejä ja orjasolmu vastaa jos sen tunnistenumerot vastaavat kyselyviestien IDNumber-kenttiä. Vastausviestin CAN-ID on 0x7E4, sen pituus on 8 tavua, ensimmäisen tavun arvo on 0x4F ja muiden tavujen 0. Jos numerot eivät täsmää, orjasolmu jättää vastaamatta. Nämä tunnistenumerot ovat samat, jotka on määritelty objektissa 0x1018 ja joita käytetään LSS Fastscan -menetelmässä.

Ensin isäntä lähettää kyselyviestin, jossa cs on 0x46 ja IDNumber on toimittajanumero. Toimittajanumero löytyy objektin 0x1018 alaindeksistä 1. Jos isäntä saa vastauksen

edelliseen viestiin, se kyselee seuraavaa numeroa. Viestin cs on 0x47 ja IDNumber on tuotekoodi. Tuotekoodi löytyy objektin 0x1018 alaindeksistä 2.

Seuraavaksi tarkistetaan solmun revisionumero. Tämä tehdään kahdella kyselyviestillä siten, että ensimmäisen viestin cs on 0x48 ja sen IDNumber määrittelee numeroalueen alarajan. Orjasolmu vastaa jos sen revisionumero on yhtäsuuri tai suurempi kuin IDNumber. Toisen viestin cs on 0x49 ja sen IDNumber määrittelee numeroalueen ylärajan. Orjasolmu vastaa jos sen revisionumero on enintään IDNumberin verran. Solmun revisionumero löytyy objektin 0x1018 alaindeksistä 3.

Sarjanumerokysely tapahtuu samaan tapaan. Ensimmäisessä viestissä cs on 0x4A ja IDNumber määrittelee sarjanumeroalueen alaraja. Toisessa viestissä cs on 0x4B ja IDNumber määrittelee sarjanumeroalueen ylärajan. Sarjanumero löytyy objektin 0x1018 alaindeksistä 4. [15, s. 42–43.]

LSS Fastscan

Kun mahdolliset vanhat LSS-orjat on tunnistettu, NMT-isäntä siirtyy tunnistamaan uusia solmuja LSS Fastscan -menetelmällä. LSS Fastcan -menetelmän tarkoitus on tunnistaa konfiguroimattomat solmut väylässä tunnisteen määräämistä ja muiden mahdollisten asetusten asettamista varten. Solmu tunnistetaan yksilöllisen 128-bittisen LSS-tunnisteen avulla. LSS-tunniste koostuu neljästä 32-bittisestä numerosta: toimittajanumerosta, tuotenumerosta, revisionumerosta ja sarjanumerosta. Nämä numerot löytyvät solmun objektikirjastosta indeksillä 0x1018, alaindekseillä 1, 2, 3 ja 4.

LSS-viestit on määritelty seuraavalla tavalla. LSS-isännän lähettämien kyselyviestien tunniste on 0x7E5. Orjien lähettämien vastausviestien tunniste on 0x7E4. Viestin data-kenttä sisältää vähintään yhden tavun. Data-kentän ensimmäinen tavu on nimeltään komentotarkenne (cs, command specifier), joka määrittää mihin LSS-protokollaan viesti liittyy.

Ennen varsinaisen LSS Fastscan -menetelmän aloittamista isäntäsolmu selvittää, onko väylässä orjasolmuja, jotka ovat tilassa NMT-alustus ja joilla ei ole aktiivista solmutunnistenumeroa. Tämä tapahtuu lähettämällä viesti, jonka tunniste on 0x7E5 ja datakentän komentotarkenne 0x4C. Sopivassa tilassa olevat solmut vastaavat tähän lähettämällä viestin, jonka tunniste on 0x7E4 ja komentotarkenne 0x50. [13, s.44.] CiA447:n mukaan orjasolmun pitää vastata kyselyyn 25 millisekunnin sisällä ja isäntäsolmu odot-

taa vastausta 1000 millisekuntia ennen kuin päättelee, ettei väylässä ole konfiguroimattomia solmuja [11, s.31].

Jos konfiguroimattomia solmuja löytyy, voi isäntäsolmu aloittaa Fastscan-prosessin. Kuva 24 esittää LSS-isännän lähettämän LSS Fastscan -viestin. Viestin tunniste on 0x7E5 ja siinä on kahdeksan data-tavua, jotka on jaettu viiteen kenttään.

CAN-ID = 0x7E5

cs=0x51	IDNumber				BitCheck	LSSSub	LSSNext
0	1	2	3	4	5	6	7

Kuva 24. LSS Fastscan -kyselyviesti isännältä orjalle

Ensimmäinen kenttä on nimeltään komentotarkenne (*cs, eli Command Specifier*) Numero 0x51 tarkoittaa, että kyseessä on LSS Fastscan -viesti.

BitCheck on tavun mittainen kenttä, joka määrittelee mitä bittejä IDNumber-kentästä käytetään vertailussa. BitCheck-arvo on välillä 0...31. Arvo 31 tarkoittaa, että vain bitti 31 tarkistetaan, 30 tarkoittaa, että bitit 30 ja 31 tarkistetaan ja niin edelleen. Arvo 0 tarkoittaa, että kaikki 32 bittiä tarkistetaan. LSS Fastscan -aloitusviestissä BitCheck-arvo on 0x80.

LSSSub-kenttä määrittelee, mihin LSS-tunnisteen osaan IDNumber-kenttää verrataan. Arvojen määrittely löytyy taulukosta 14.

LSSNext-kentällä LSS-isäntä kertoo mikä seuraavan viestin LSSSub-arvo on.

Taulukko 14. LSSSub- ja LSSNext-arvojen määrittely.

Arvo	**LSS-numero**
0	Toimittajanumero (Vendor ID)
1	Tuotekoodi
2	Revisionumero
3	Sarjanumero
4…255	Varattu

LSS Fastscan -prosessi

Löytääkseen konfiguroimattomat solmut LSS-isäntä lähettää LSS Fastscan-kyselyviestin, jonka BitCheck-kenttä on 0x80. Kaikki LSS Fastscania tukevat solmut, joita ei vielä ole konfiguroitu, lähettävät LSS Fastscan -vastausviestin, jonka tunniste

on 0x7E4 ja komentotarkenne 0x4F. Jos yksikään solmu ei vastaa kyselyyn 10 millisekunnin kuluessa isäntä tulkitsee, ettei väylässä ole konfiguroimattomia solmuja ja lopettaa prosessin. Kaavio LSS Fastscan -prosessista löytyy kuvasta 25.

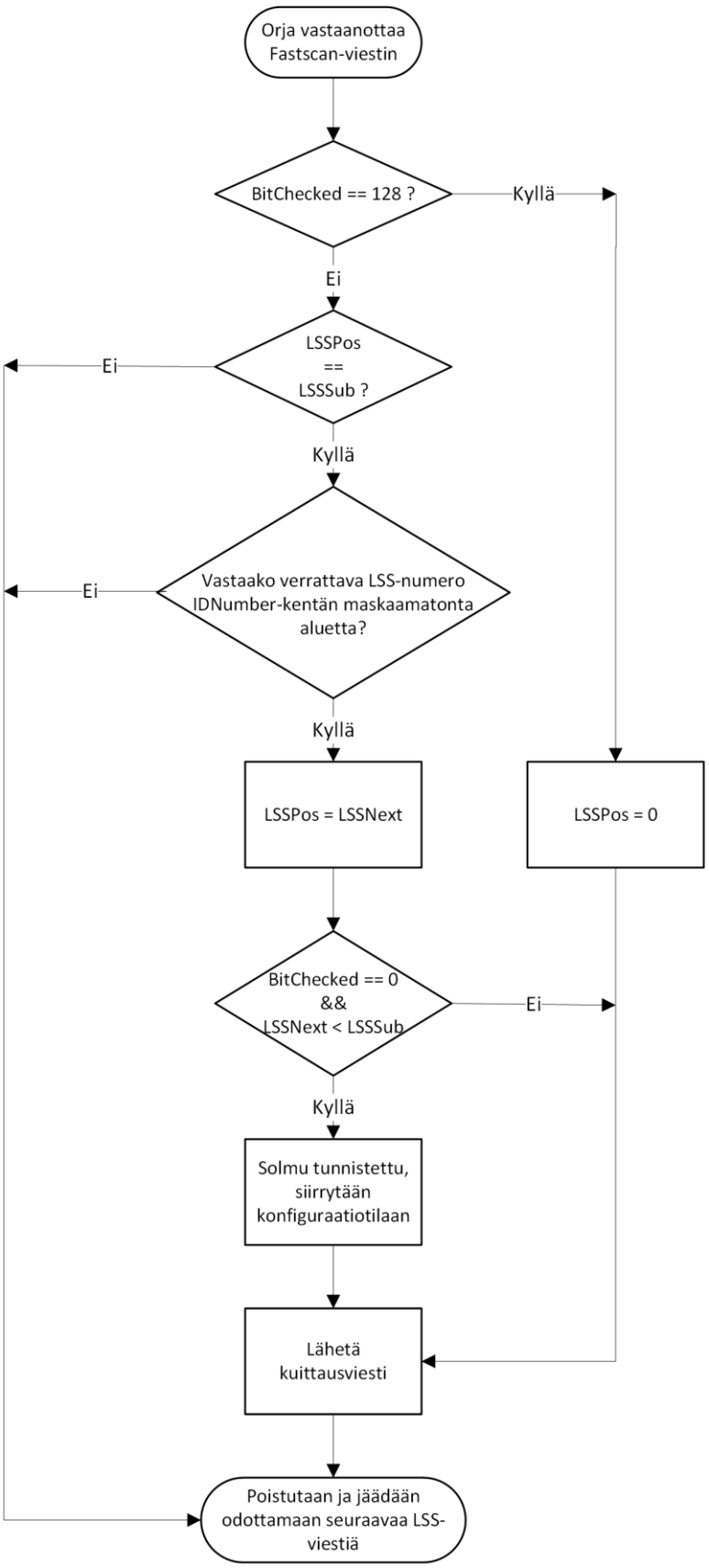
Tämän jälkeen isäntä aloittaa LSS-tunnisteen skannaamisen neljässä 32-bittisessä osassa. Alkuolettamus on, että tarkistettava 32-bittinen numero on 0x00000000 ja konfiguroitavan solmun LSS-numeroita verrataan sitä vasten. Ensimmäisessä kyselyviestissä IDNumber on siis 0x00000000, LSSSub on 0 ja BitCheck on 31. Konfiguroitava solmu vertaa toimittajanumeronsa bittiä 31 LSS-viestin IDNumber-kentän bittiin 31. Mikäli bitit täsmäävät, eli toimittajanumeron 31 bitti on nolla, solmu lähettää LSS Fastscan -vastausviestin. Jos bitit eivät täsmää, jättää solmu vastaamatta. Mikäli vastausviestiä ei kuulu 10 millisekunnin kuluessa, tulkitsee isäntä bitin ykköseksi.

Seuraavaksi LSS-isäntä lähettää viestin, jonka BitCheck-kenttä on 30. Jos konfiguroitava solmu kuittasi edellisen viestin bitin 31 täsmäävän, IDNumber kenttä on edelleen 0x00000000, muuten se on 0x10000000. Nyt konfiguroitava solmu vertaa toimittajanumeron bittejä 30 ja 31 IDNumber-kentän vastaaviin bitteihin ja lähettää vastausviestin jos ne täsmäävät.

Varsinaisen LSS Fastscan -menetelmän jälkeen LSS-isäntä antaa tunnistetulle solmulle solmutunnisteen lähettämällä LSS-viestin, jonka komentotarkenne on 0x11, ja sitä seuraava datatavu sisältää annettavan tunnisteen. Orja kuittaa tämän lähettämällä vastausviestin, jonka ainoa datatavu sisältää komentotarkenteen 0x11.

Tämän jälkeen isäntä lähettää viestin, jolla se pyytää orjasolmua tallentamaan asetukset. Viestissä on yksi datatavu ja se sisältää komentotarkenteen 0x17. Orja kuittaa viestin lähettämällä samansisältöisen viestin takaisin.

Isännän viimeinen viesti asettaa solmun LSS-tilaan odotus (waiting). Siinä on yksi datatavu, joka sisältää komentotarkenteen 0x04. Tätä viestiä orjan ei tarvitse kuitata ja se voi viestin vastaanottamisen jälkeen siirtyä NMT-tilaan esitoiminnallinen (pre-operational). [15, s. 25–31; s. 45.]



Kuva 25. LSS Fastscan -prosessi orjasolmun näkökulmasta [15, s. 27].

Taulukossa 15 on esimerkki LSS Fastscan -menetelmän ja sen jälkeen tapahtuvan LSS-konfiguraation viestiliikenteestä isännän ja yhden orjasolmun välillä.

Taulukko 15. LSS Fastscan -menetelmän ja sen jälkeisen konfiguroinnin viestiliikenne.

CAN-ID		Viesti (heksadesimaali)	IDNumber	Bit-Check	LSSSub	LSSNext
0x7E5	Kysely: Etsi konfiguroimattomat solmut	51 00 00 00 00 80 00 00		0x80		
0x7E4	Vastaus: Konfiguroimaton solmu kuittaa	4F 00 00 00 00 00 00 00				
0x7E5	Kysely: toimittajanumeron bitti 31	51 00 00 00 00 1F 00 00	0x00000000	31	0	0
0x7E4	Vastaus: bitti 31 on nolla	4F 00 00 00 00 00 00 00				
0x7E5	Kysely: toimittajanumeron bitit 30…31	51 00 00 00 00 1E 00 00	0x00000000	30	0	0
0x7E4	Vastaus: bitti 30 on nolla	4F 00 00 00 00 00 00 00				
0x7E5	Kysely: toimittajanumeron bitit 29…31	51 00 00 00 00 1D 00 00	0x00000000	29	0	0
0x7E4	Vastaus: bitti 29 on nolla	4F 00 00 00 00 00 00 00				
0x7E5	Kysely: toimittajanumeron bitit 28…31	51 00 00 00 00 1C 00 00	0x00000000	28	0	0
	Ei vastausta: bitti 28 on yksi					
0x7E5	Kysely: toimittajanumeron bitit 27…31	51 00 00 00 10 1B 00 00	0x10000000	27	0	0
0x7E4	Vastaus: bitti 27 on nolla	4F 00 00 00 00 00 00 00				
0x7E5	Kysely: toimittajanumeron bitit 26…31	51 00 00 00 10 1A 00 00	0x10000000	26	0	0
	Ei vastausta: bitti 26 on yksi					
0x7E5	Kysely: toimittajanumeron bitit 25…31	51 00 00 00 14 19 00 00	0x14000000	25	0	0
0x7E4	Vastaus: bitti 25 on nolla	4F 00 00 00 00 00 00 00				
	Jatkuu…					
0x7E5	Kysely: toimittajanumeron bitit 0…31	51 60 3F 55 14 00 00 00	0x14553F60	0	0	0
	Ei vastausta: bitti 0 on yksi					
0x7E5	Kysely: varmista koko toimittajanumero	51 61 3F 55 14 00 00 01	0x14553F61	0	0	1
0x7E4	Kuitataan toimittajanumero 0x14553F61	4F 00 00 00 00 00 00 00				
	Jatkuu…					
0x7E5	Kysely: varmista koko sarjanumero	51 44 33 22 11 00 03 04	0x11223344	0	3	4
0x7E4	Kuitataan sarjanumero 0x11223344	4F 00 00 00 00 00 00 00				
0x7E5	Anna solmulle solmutunniste 2, cs=11	11 02 00 00 00 00 00 00				
0x7E4	Kuitataan solmutunniste	11 00 00 00 00 00 00 00				
0x7E5	Pyyntö: tallenna asetukset, cs=17	17 00 00 00 00 00 00 00				
0x7E4	Vastaus: asetukset tallennettu	17 00 00 00 00 00 00 00				
0x7E5	Pyyntö: siirry tilaan LSS odotus (waiting)	04 00 00 00 00 00 00 00				
0x702	Solmu 2 käynnistyy	0				

Solmujen valmistelu käynnistystä varten

Kiinteän solmutunnisteen omaavien solmujen tunnistamisen ja LSS Fastscan -menetelmän jälkeen kaikkien orjasolmujen tulisi olla NMT-tilassa esitoiminnallinen. Tässä vaiheessa isäntä voi vielä valmistella ja konfiguroida solmuja lukemalla ja kirjoittamalla niiden objektikirjastoihin. CiA447 ei määrittele, mitä tässä vaiheessa tapahtuu, mutta isäntä voi esimerkiksi ottaa selvää, mitä virtuaalilaitteita solmut sisältävät ja konfiguroida PDO-objekteja.

Kun kaikki asetukset on tehty, isäntä käynnistää kaikki solmut lähettämällä niille NMT-solmunohjausviestin komentotarkenteella 0x01. Viestin voi lähettää jokaiselle yksitellen määrittelemällä viestin solmutunniste-kentän arvoksi ohjattavan solmun tunnisteen tai kaikille yhtä aikaa määrittelemällä solmutunniste-kentän arvoksi 0. Tämän jälkeen verkko on toimintakunnossa. [14, s. 12.]

4.4 Verkonhallinta

CiA447-yhteensopivien solmujen tulee tukea CANopen-standardin määrittelemiä verkonhallintaprotokollia, jotka on määritelty luvussa 3.6. Laite, joka sisältää IVN-yhdyskäytävän, toimii NMT-isäntänä. Muut solmut ovat IVN-orjia, ja niiden tulee tukea Heartbeat-protokollaa. Node- ja Life Guarding -protokollia ei tueta. [14, s. 13.]

4.5 Virheilmoitukset

CiA447-yhteensopivan solmun tulee tukea EMCY-protokollaa, kuten se on määritelty CANopen-standardissa ja luvussa 3.8 [14, s. 13]. CiA447 määrittelee lisäksi uusia virhekoodeja, jotka on listattu taulukossa 16.

Taulukko 16. CiA447-profiilin lisäämät virhekoodit [14, s. 14].

Virhekoodi	**Selitys**
0xF001	Yleinen varoitus
0xF002	Vakava varoitus
0xF003	Yleinen vika
0xF004	Vakava vika
0xF005	Osittainen verkon toiminta
0xF006	Verkon käynnistys
0xF007	Väylät lepotilassa
0xF008	Kommunikaatiovirhe ajoneuvon sisäisissä verkoissa

4.6 Objektikirjasto

Solmuille on CANopen-standardissa määritelty objektikirjaston merkinnät, jotka niiden vähintään tulee toteuttaa. CiA447 antaa näihin merkintöihin liittyviä lisämäärittelyjä ja määrittelee pari pakollista merkintää lisää.

Objektin 0x1000 sisältö määrittelee *laitetyypin*. Se on UNSIGNED32-arvo, jonka bitit 0…15 sisältävät laiteprofiilin numeron 447. Bitit 16…23 määrittelevät sovellusprofiilin version. Arvo 0 tarkoittaa versiota 1.0, arvo 1 versiota 2.0. Muut arvot eivät ole käytössä. Objektin bitit 24…31 eivät ole käytössä, ja niiden arvo on aina 0.

Objektikirjaston merkintä 0x1001 on määritelty pakolliseksi CANopen-standardissa ja se sisältää *virherekisterin*, jonka sisältö on määritelty kappaleessa 0. Virherekisterin lisäksi CiA447 määrittelee pakolliseksi objektin 0x1003, joka on *virhelista.* Virhelistan alaindeksi 0 kertoo kuinka monta virhettä listaan on merkitty, maksimissaan 254. Alaindeksit sisältävät UNSIGNED32-tyyppisen arvon, jonka ensimmäiset 2 tavua sisältävät virhekoodin, jotka vastaavat EMCY-objektin virhekoodeja. Virhetyypit on listattu taulukossa 12. Näiden lisäksi CiA447 määrittelee uusia virhekoodeja, jotka on listattu taulukossa 16. Virhelistan alaindeksien tavut 3…4 sisältävät valmistajakohtaisen lisätietokentän.

Objektikirjaston merkintä 0x1016 sisältää *Heartbeat-kuluttajan aikaväliarvon*. Tämä merkintä ei ole pakollinen, mutta jos se toteutetaan solmuun, sen suositeltu arvo on 700 millisekuntia.

Objektikirjaston merkintä 0x1017 sisältää *Heartbeat-tuottajan aikaväliarvon.* Merkintä on pakollinen ja sen suositusarvo on 200 millisekuntia. [14, s. 16.]

Objekti 0x6000 on pakollinen, ja se määrittelee laitteen tukemat *virtuaalilaitteet.* Virtuaalilaite on laitteessa oleva ohjelma, joka toteuttaa tietyn toiminnallisen kokonaisuuden. Yksi fyysinen laite voi sisältää useita virtuaalilaitteita. Objektilla 0x6000 on 2 tai 3 alaindeksiä. Ensimmäinen alaindeksi määrittelee seuraavien alaindeksien määrän ja seuraavat alaindeksit sisältävät virtuaalilaitemäärittelyt. Asetettu bitti tarkoittaa tuettua virtuaalilaitetta. [14, s. 17.] Objektin 0x6000 alaobjektin 1 rakenne on kuvattu taulukossa 17 ja alaobjektin 2 rakenne taulukossa 18.

Taulukko 17. Objektin 0x6000 alaindeksin 1 rakenne [14, s. 17].

Bitti	Selitys	Tyyppi
0	IVN-yhdyskäytävä, luokka 0	Yhdyskäytävä
1	IVN-yhdyskäytävä, luokka 1	Yhdyskäytävä
2	IVN-yhdyskäytävä, luokka 2	Yhdyskäytävä
3	IVN-yhdyskäytävä, luokka 3	Yhdyskäytävä
4	Palonsammutusjärjestelmä	Laite
5	Hätätuuletusjärjestelmä	Laite
6	Teholähde	Laite
7	Digitaalisisääntulot	Laite
8	Terminaali	Laite
9	GPS	Laite
10	Navigaatiojärjestelmä	Laite
11	Taksamittari	Laite
12	Tulostin	Laite
13	Reaaliaikakello	Laite
14	Kuljettajan tunnistus	Laite
15	Taksanäyttö	Laite
16	Taksin hälytysjärjestelmä	Laite
17	Radio	Laite
18	Audiokytkin	Laite
19	Kattopalkin valot	Laite
20	Kattopalkin äänet	Laite
21	Sinisen valon vilkuttaja	Laite
22	Kattopalkin kontrolleri	Kontrolleri
23	Radiokontrolleri	Kontrolleri
24	Inva-auton kontrolleri	Kontrolleri
25	Radion hands-free	Laite
26	Testeri	Testeri
27	Informaation signaloija	Laite
28	Video	Laite
29	Datatallennin	Laite
30	Näppäimistö	Geneerinen laite
31	Sisääntulojen/ulostulojen kytkin	Geneerinen laite

Taulukko 18. Objektin 0x6000 alaindeksin 1 rakenne [14, s. 18].

Bitti	Selitys	Tyyppi
0	Mootorinohjain	Laite
1	Nopeudensäädin	Laite
2…31	Varattu (aina 0)	-

Virtuaalilaitteiden tyypit on määritelty seuraavalla tavalla. *Yhdyskäytävä* on virtuaalilaite, joka on yhteydessä ajoneuvon sisäiseen verkkoon ja tarjoaa sen toimintoja lisälaiteverkkoon. *Laite* on virtuaalilaite, joka toteuttaa jonkin sovelluskohtaisen toiminnon sisään- ja ulostuloineen. *Geneerinen laite* tarjoaa sisään- ja ulostuloja yleiseen käyttöön, mutta ei määrittele niihin liittyviä sovellustoimintoja. *Kontrolleri* on virtuaalilaite, joka

lukee laitteen tiloja ja laukaisee näiden pohjalta toimintoja muissa virtuaalilaitteissa lähettämällä ennalta määrättyjä komentoja. *Testeri-virtuaalilaite* tarjoaa tiettyjä diagnostiikkatoimintoja järjestelmäintegraatiota ja huoltoa varten. [14, s. 6.]

4.7 SDO

CiA447-yhteensopivassa lisälaiteväylässä jokaisen solmun tulee toteuttaa SDO-palvelin jokaiselle väylässä olevalle laitteelle. Koska väylän solmujen määrä on rajoitettu kuuteentoista, tulee palvelimia olla 15. Lisäksi solmu voi toimia SDO-asiakkaana jokaiselle väylän solmulle. Näitä asiakkaiden ja palvelimien välisiä yhteyksiä kutsutaan kommunikaatiokanaviksi. CiA447-verkossa jokaisen SDO-kommunikaatiokanavan kehyksille on määritelty oma tunniste, joka perustuu asiakkaan ja palvelimen solmutunnistenumeroihin. [16, s. 4–5.]

Kehysten tunnisteet

Kehyksen tunniste SDO-pyynnölle asiakkaalta palvelimelle lasketaan kaavalla

$$(0x240 + (((i-1)\&0xC)<<6) + (((i-1)\&0x3)<<4)+(j-1))$$

ja SDO-vastaukselle palvelimelta asiakkaalle kaavalla

$$(0x1C0 + (((i-1)\&0xC)<<6) + (((i-1)\&0x3)<<4)+(j-1)),$$

missä i on asiakkaan solmutunniste ja j on palvelimen solmutunniste. [16, s. 15.]

Kehysten tunnisteet solmun kommunikaatiokanaville voidaan tallentaa myös objektikirjastoon alueille 0x1200…0x120E ja 0x1280…0x12FF. Ensin mainittu alue on tarkoitettu solmun SDO-palvelimen parametreille ja jälkimmäinen solmun SDO-asiakkaan parametreille. Näiden objektien alaindeksi 1 määrittelee tunnisteen asiakkaalta palvelimen suuntaan ja alaindeksi 2 palvelimelta asiakkaan suuntaan. Alaindeksi 3 määrittelee kyseisen kommunikaatiokanavan toisen osapuolen solmutunnisteen. [8, s. 126–130.] Täytyy muistaa, että jos solmun solmutunniste määritellään dynaamisesti LSS Fastscan -pavelulla, voi solmutunniste muuttua käynnistyskertojen välissä. Tällöin kommunikaatiokanavien parametreja ei voida tallentaa objektikirjastoon pysyvästi.

Toisin kuin CANopen-standardissa on määritelty, solmu saa CiA447-profiilin mukaan toimia SDO-asiakkaana ainoastaan NMT-tilassa toiminnallinen [16, s. 5].

5 Laitteisto

Semelin TM6000-taksamittarin keskusyksikkö TM206 sisältää Atmel AT91SAM7X512 -mikrokontrollerin, jossa on 32-bittinen ARM-prosessori. Piiri sisältää 512 kilotavua Flash-muistia ja 128 kilotavua RAM-muistia sekä 62 ohjelmoitavaa I/O-linjaa, jotka on multipleksoitu yleisen I/O:n ja oheislaitteiden välillä.

5.1 Sarjaportit

AT91SAM7X512 sisältää kolme sarjaliikenneporttia (USART). Näistä kaksi, USART1 ja USART2, tukevat RS232-standardin mukaisen asynkronisen tiedonsiirron lisäksi synkronista tiedonsiirtoa ja ovat myös RS485-yhteensopivia. Portit sisältävät lähetys- ja vastaanottolinjojen lisäksi myös kättelysignaalit RTS ja CTS ja kellolinjan SCK. USART1 sisältää näiden lisäksi niin sanotut modeeminohjauslinjat DTR, DSR, DCD ja RI. [17, s. 35.]

Kolmas sarjaliikenneportti, nimeltään Debug Unit, on tarkoitettu testaamiseen ja vianetsintään, mutta sitä voi käyttää myös yleisenä sarjaliikenneporttina. Debug Unit sisältää ainoastaan vastaanotto- ja lähetyslinjat ja tukee ainoastaan 8-bittistä datapakettia. [17, s. 28.]

Kaikki kolme sarjaporttia on TM260:ssa kytketty laitteen toiseen päätyyn, ulkoisiin modulaariliittimiin. USART0 menee liittimeen COM0, USART1 liittimeen COM1 ja Debug Unit liittimeen PRINTER.

Sarjaporttien konfigurointi tapahtuu kirjoittamalla konfigurointirekistereihin, mutta sen käsittely ei sisälly tämän työn laajuuteen. Tarkempia tietoja löytyy mikrokontrollerin datalehden luvusta 30.

5.2 CAN-sovitin

AT91SAM7X512 sisältää yhden CAN-ohjaimen, jonka lähetys- ja vastaanottolinjat on TM206-yksikössä liitetty NXP:n TJA1040 CAN-sovittimeen. Vastaanottolinja on mikrokontrollerin linja PA19, eli pinni 46 ja se on kytketty sovittimen pinniin 4. Lähetyslinja on linja PA20, eli pinni 47 ja se on kytketty sovittimen pinniin 1. TJA1040 sisältää CAN-väylälinjojen sekä ohjaimeen liitettyjen lähetys- ja vastaanottolinjojen lisäksi virransäästötilan kytkentälinjan, joka kytkee piirin virransäästötilaan jos linja vedetään käyttöjännitteeseen. TM260-yksikössä TM1040-sovittimen virransäästötila-linjaan, eli pinniin 8 on kytketty mikrokontrollerin linja PA21 eli pinni 49.

TM206-yksikössä TJA1040-sovittimen väylälinjat CAN-high eli pinni 7, CAN-low eli pinni 6 ja maa eli pinni 2 on kytketty suoraan ulkoiseen liittimeen. Samasta neljäpinnisestä liittimestä löytyy maan lisäksi ja erillinen CAN-maa, joka on erotettu maasta sadan ohmin vastuksella. TM206 kytkettiin simulaattoriin ja autoon juuri tämän erotetun maapinnin kautta.

5.3 CAN-ohjain

AT91SAM7X512-mikrokontrollerin CAN-ohjain on täysin CAN-standardin mukainen ja tukee sekä tavallisia, että laajennettuja viestikehyksiä ja RTR-, virhe- sekä viivekehyksiä. Ohjain tukee nopeuksia 1 Mbit:iin/s saakka. Ohjain sisältää kahdeksan viestipuskuria, joita kutsutaan postilaatikoiksi. Jokaisen postilaatikon voi määritellä joko lähetys- tai vastaanottotilaan ja asetusta voi muuttaa dynaamisesti. Vastaanottotilassa postilaatikolle voidaan määritellä suodatin, joka päästää läpi vain viestit, joiden tunniste on määrätyn alueen sisällä. [17, s. 503.]

Konfigurointi

Ennen CAN-ohjaimen konfigurointia tulee mikrokontrollerin pinnit 46 ja 47 asettaa CAN-ohjaimen käyttöön ja pinni 49 tavalliseksi TTL-ulostuloksi. Tämän jälkeen tulee CAN-ohjaimen kello aktivoida asettamalla PMC_PCER-rekisterin bitit 2 ja 15. Seuraavaksi mikrokontrollerin keskeytysohjain tulee konfiguroida CAN-keskeytyksiä varten. AIC_SMR15-rekisteriin kirjoitetaan keskeytyksen haluttu prioriteetti, AIC_SVR15-rekisteriin osoite keskeytyksenkäsittelyrutiiniin. Keskeytys nollataan asettamalla

AIC_ICCR-rekisterin bitti 15 ja aktivoidaan asettamalla AIC_IECR-rekisterin bitti 15. [17, s. 518–520.]

Itse CAN-ohjaimen konfigurointia varten on omistettu 11 32-bittistä rekisteriä. Ne käydään tässä läpi siinä järjestyksessä, kuin ne on ohjelmoitaessa järkevä määritellä.

Ajoitusparametrit

Ennen kuin CAN-ohjain aktivoidaan, tulee ohjaimelle asettaa väylän ajoitusparametrit. Parametrit määritellään rekisteriin CAN_BR, jonka rakenne näkyy taulukossa 19.

Taulukko 19. CAN-ohjaimen CAN_BR-ajoitusparametrirekisterin sisältö

Bitti	**Nimi**	**Selitys**
0…2	PSEG2	Toisen vaihevirheiden kompensointisegmentin pituus
3	-	
4…6	PSEG1	Ensimmäisen vaihevirheiden kompensointisegmentin pituus
7	-	
8…10	PROP	Etenemissegmentin pituus
11	-	
12…13	SJW	Synkronointihypyn pituus
14…15	-	
16…22	BRP	Siirtonopeuden esijakaja
23	-	
24	SMP	Näytteistystapa
25…31	-	

Vaihevirheiden kompensointiparametrit *PSEG1* ja *PSEG2*, etenemissegmentin pituus *PROP* ja synkronointihypyn pituus *SJW* on määritelty luvussa 2.1. On huomattava, että CAN_BR-rekisterissä parametrien arvoista on vähennetty 1. Täten esimerkiksi etenemissegmentin pituus on $aikayksikk\ddot{o} \times (PROP + 1)$.

CAN-ohjaimen aikayksikön pituus määritellään parametrilla *BRP*. Aikayksikön pituus on $\frac{\mathrm{BRP+1}}{\mathrm{MCK}}$, missä MCK on mikrokontrollerin pääkellon (Master Clock) taajuus. BRP-kentän arvon tulee olla välillä 0x01…0x7F.

SMP-parametri määrittelee CAN-ohjaimen näytteistystavan. Jos arvo on 0, vastaanotettava bitti näytteistetään kerran näytteenottopisteessä. Jos arvo on 1, bitti näytteistetään kolme kertaa siten, että keskimmäinen näyte otetaan näytteenottopisteessä. [17, s. 543.]

Toimintatila

Rekisteri CAN_MR määrittelee ohjaimen toimintatilan. Rekisterin rakenne on esitetty taulukossa 20. Sen avulla voidaan aktivoida CAN-ohjain ja muuttaa sen toimintatiloja ja asetuksia. Rekisterin arvo on käynnistettäessä 0.

Taulukko 20. CAN_MR-toimintatilarekisterin sisältö

Bitti	**Nimi**	**Selitys**
0	CANEN	CAN-ohjaimen aktivoiminen
1	LPM	Virransäästötila
2	ABM	Autobaud-/kuuntelutila
3	OVL	Viivekehyksen lähettäminen
4	TEOF	Aikaleiman ajankohta
5	TTM	Aikaliipaistu tila (Time Triggered)
6	TIMFRZ	Ajastimen pysäytys
7	DRPT	Uudelleenlähetyksen esto
8…23	-	
24…26	RXSYNC	Synkronoinnin vastaanottovaihe
27…31	-	

Bitti 0, *CANEN*, asettaa CAN-ohjaimen toimintatilaan, jos sen arvo on 1. Muutoin ohjain on pois päältä.

Bitti 1, *LPM*, asettaa CAN-ohjaimen virransäästötilaan, jos sen arvo on 1. Muutoin virransäästötila on pois päältä.

Bitti 2, *ABM*, asettaa päälle kuuntelutilan automaattista siirtonopeuden asettamista varten.

Bitti 3, *OVL*, määrittelee, onko viivekehyksen lähetys käytössä. Jos bitin arvo on 0, viivekehystä ei lähetetä. Jos arvo on 1, viivekehys lähetetään jokaisen vastaanotetun kehyksen jälkeen.

Bitti 4, *TEOF*, määrittelee, milloin CAN-ohjaimen aikaleima merkitään, eli CAN-ohjaimen laskurirekisterin CAN_TIM arvo kopioidaan aikaleimarekisteriin CAN_TIMESTP. Jos bitin arvo on 0, aikaleima otetaan kehyksen alussa, jos 1, aikaleima otetaan kehyksen lopussa.

Bitti 5, *TTM*, asettaa CAN-ohjaimen aikaliipaistuun tilaan (Time Triggered Mode), jos sen arvo on 1. Arvon ollessa 0 ohjain on normaalitilassa.

Jos bitti 6, *TMFRZ*, on 1, CAN-ohjaimen sisäinen laskuri pysähtyy, kun se saavuttaa arvon 0xFFFF. Arvolla 0 laskuri jatkaa pyörimistä ylivuodon jälkeen.

Bitti 7, *DRPT*, määrittelee, onko kehysten automaattinen uudelleenlähetys käytössä. Jos bitin arvo on 0, kehys lähetetään törmäystilanteen jälkeen automaattisesti uudestaan. Jos arvo on 1, kehyksen lähettäminen keskeytetään ja postilaatikon tilarekisterin CAN_MSRx liput MABT ja MRDT asetetaan.

Bittikenttä *RSYNC*, eli bitit 24…26, määrittelee synkronoinnin vastaanottovaiheen. Tämä parametri on tarkoitettu vianetsintään, eikä sitä käsitellä tarkemmin tässä. [17, s. 532.]

Aktivoiminen

CAN-ohjain aktivoidaan asettamalla CAN_MR-rekisterin bitti 0. Aktivoimisen yhteydessä ohjaimen sisäinen tilakone, virhelaskurit ja virheliput nollataan. Kun ohjain on aktivoitu, se pyrkii lukemaan 11 resessiivistä bittiä väylästä, ja onnistuessaan tässä tuottaa keskeytyksen ja asettaa WAKEUP-bitin CAN_SR-rekisterissä, josta lisää seuraavassa luvussa. Tätä prosessia kutsutaan AT91SAM7X512-mikrokontrollerin datalehdessä ja koodiesimerkeissä harhaanjohtavasti synkronoinniksi, mutta sillä ei ole mitään tekemistä CAN-väylän varsinaisen synkronoinnin kanssa. [17, s. 518.]

Keskeytykset

CAN-ohjain voi aiheuttaa keskeytyksen monen eri tapahtuman perusteella. Asetetut keskeytystapahtumat voi lukea rekisteristä keskeytysmaskirekisteristä CAN_IMR. Keskeytystapahtumat määritellään asettamalla halutut bitit rekistereihin CAN_IER ja CAN_IDR. CAN_IER on keskeytysten aktivoimisrekisteri, eli sinne asetettu bitti aktivoi keskeytyksen. CAN_IDR toimii toisin päin, eli sinne asetettu bitti deaktivoi keskeytyksen. Kun keskeytys tapahtuu, voidaan CAN-ohjaimen tilarekisterin CAN_SR perusteella päätellä, mikä keskeytyksen on aiheuttanut. Keskeytyksen syyn lisäksi CAN_SR antaa samanaikaisesti myös muuta tilatietoa CAN-ohjaimesta. Rekisterin asetetut bitit kertovat, mitä tapahtumia on käynyt tai mitkä tilat ovat päällä. CAN_IMR-, CAN_IER- ja CAN_IDR-rekistereillä on kaikilla sama rakenne. CAN_SR-rekisteri eroaa näistä ainoastaan kolmen viimeisen bitin verran, jotka kertovat lähettimen ja vastaanottimen tilatietoja. [17, s. 519–520.] Rekisterien rakenne on kuvattu taulukossa 21.

Taulukko 21. Rekisterien CAN_IER, CAN IDR, CAN_IMR ja CAN_SR rakenne.

Bitti	Nimi	Selitys
0	MB0	Postilaatikko 0
1	MB1	Postilaatikko 1
2	MB2	Postilaatikko 2
3	MB3	Postilaatikko 3
4	MB4	Postilaatikko 4
5	MB5	Postilaatikko 5
6	MB6	Postilaatikko 6
7	MB7	Postilaatikko 7
8…15	-	-
16	ERRA	Aktiivinen tila
17	WARN	Virhelaskurin varoitusraja ylitetty
18	ERRP	Passiivinen tila
19	BOFF	Väylä irti –tila
20	SLEEP	Virransäästötila
21	WAKEUP	Herääminen virransäästötilasta
22	TOVF	Laskurin ylivuoto
23	TSTP	Aikaleima
24	CERR	CRC-virhe
25	SERR	Bit-stuffing-virhe
26	AERR	Kuittausvirhe
27	FERR	Muotovirhe
28	BERR	Bittivirhe
29	(RBSY)	Vastaanotin varattu (vain CAN_SR)
30	(TBSY)	Lähetin varattu (vain CAN_SR)
31	(OVLSY)	Viivekehyksen lähettäminen kesken (vain CAN_SR)

Bitit 0…7 liittyvät postilaatikoiden lähetys- ja vastaanottokeskeytyksiin. Keskeytys tapahtuu, kun uusi viesti vastaanotetaan tai viestin lähetys on valmis.

Bitit 16…19 liittyvät virheidenhallintaan. Nämä bitit kertovat, missä virheidenhallintatilassa CAN-ohjain on. Keskeytys tapahtuu, kun ohjain siirtyy tilasta toiseen tai kun virhelaskurin varoitusraja ylittyy.

Bitit 20 ja 21 liittyvät virransäästöominaisuuksiin. Bitti 20, SLEEP, on tilatieto ja siihen liittyvä keskeytys tapahtuu, kun CAN-ohjain siirtyy virransäästötilaan. Bitti 21, WAKEUP, on tapahtumatieto ja siihen liittyvä keskeytys tapahtuu, kun CAN-ohjain herää virransäästötilasta. WAKEUP-keskeytys tapahtuu myös, kun CAN-ohjain aktivoidaan ensimmäisen kerran virran kytkemisen jälkeen.

TOVF-keskeytys tapahtuu kun CAN-ohjaimen laskuri vuotaa yli.

TSTP-keskeytys tapahtuu, kun CAN-ohjain merkitsee aikaleiman. Aikaleiman merkintäajankohta riippuu CAN_MR-rekisterin TEOF-bitin arvosta. (Taulukko 20)

Bitit 24…28 kattavat eri virheiden keskeytykset.

Bitit 29…30 ovat käytössä vain tilarekisterissä CAN_SR. RBSY-bitti on asetettu, jos CAN-ohjaimen vastaanotin on varattu, eli vastaanotto on kesken. TBSY on asetettu, jos lähetin on varattu, ja OVLSY, jos viivekehyksen lähetys on kesken. [17, 534–542.]

Ajastin, aikaleima ja virhelaskurit

CAN-ohjain sisältää 16-bittisen laskurin, joka käynnistyy samalla kun ohjain aktivoidaan. Laskurin on sidottu CAN-ohjaimen bittikelloon ja sen arvo löytyy rekisteristä CAN_TIM. Rekisteri on 32-bittinen, mutta vain 16 alinta bittiä sisältävät dataa. Laskuria käytetään sisäisesti kahteen eri ominaisuuteen. Kun CAN-ohjain on aikaliipaisutilassa, viestien lähetysten ajankohdat määritellään CAN-ohjaimen laskurin perusteella. Lisäksi ohjain kopioi laskurin arvon CAN_TIMESTP-aikaleimarekisteriin jokaisen vastaanotetun ja lähetetyn kehyksen yhteydessä. Tämä rekisteri on myös 32-bittinen, mutta sisältää vain 16 bittiä dataa. [17, s. 544–545.]

Rekisteri CAN_ECR sisältää CAN-ohjaimen *virhelaskurit*. Vastaanottovirhelaskuri on bittikenttä 0…7 ja lähetysvirhelaskuri bittikenttä 16…23. Rekisterin muut bitit eivät ole käytössä. Laskureita kasvatetaan ja vähennetään luvussa 2.4 lueteltujen sääntöjen mukaisesti. [17, s. 546.]

Ohjaus- ja keskeytysrekisterit

CAN-ohjaimen ohjausrekisteri CAN_TCR sisältää lähetyskäskybitit jokaiselle postilaatikolle biteissä 0…7. Ohjattavan postilaatikon numero vastaa bitin järjestysnumeroa rekisterissä. Bitit toimivat täsmälleen samoin kuin yksittäisen postilaatikon MTCR-bitti rekistereissä CAN_MCR0…8. Lisäksi CAN_TCR sisältää laskurin nollausbitin TIMRST, jonka asettamalla voidaan nollata CAN-ohjaimen sisäinen laskuri. [17, s. 547; s. 558–559.]

Lähetyksen keskeytysrekisteri CAN_ACR sisältää keskeytysbitit jokaiselle postilaatikolle biteissä 0…7. Asettamalla postilaatikon numeroa vastaava bitti voidaan viestin lähetys keskeyttää. Bitit toimivat samalla tavalla kuin MACR-bitti postilaatikoiden ohjausrekistereissä CAN_MCR0…7. [17, s. 548; s. 558–559.]

Postilaatikot

AT91SAM7X512 sisältää 8 CAN-viestipuskuria, joita kutsutaan postilaatikoiksi. Jokaista postilaatikkoa kohden on 8 rekisteriä, joilla ohjataan postilaatikon toimintaa, määritellään viestisuodattimet ja joista voidaan lukea postilaatikon tilatietoja sekä lukea tai kirjoittaa varsinainen siirrettävä data. [17, s. 503.]

Toimintamoodirekisteri CAN_MMRx (Message Mode Register)

Toimintamoodirekisterillä määritellään postilaatikon tyyppi, prioriteetti ja aikaliipaisutilassa viestin lähetysaika. Rekisterin rakenne näkyy taulukossa 22.

Taulukko 22. Toimintamoodirekisterin CAN_MMRx rakenne

Bitti	**Nimi**	**Selitys**
0…15	MTIMEMARK0…15	Aikaliipaistun viestin lähetysaika
16…19	PRIOR	Postilaatikon prioriteetti lähetystilassa
20…23	-	
24…26	MOT	Postilaatikon toimintatila
27…31	-	

MOT-bittikenttä määrittelee postilaatikon toimintatilan. Arvolla 0 postilaatikko on deaktivoitu. Arvo 1 asettaa postilaatikon vastaanottotilaan siten, että jos postilaatikon datarekisteri on täynnä kun uusi viesti saapuu, se hylätään. Arvolla 2 postilaatikko asetetaan tilaan ”vastaanotto ylikirjoituksella” siten, että uuden viestin saapuessa datarekisterissä oleva viesti hylätään. Arvo 3 asettaa postilaatikon lähetystilaan. Arvo 4 asettaa postilaatikon kuluttajatilaan ja arvo 5 tuottajatilaan. Kuluttaja- ja tuottajatiloja ei käsitellä tässä dokumentissa. Lisätietoa niistä löytyy AT91SAM7X512-mikrokontrollerin datalehden luvusta 36. [17, s. 549.]

Viestien tunnisteet ja suodattaminen

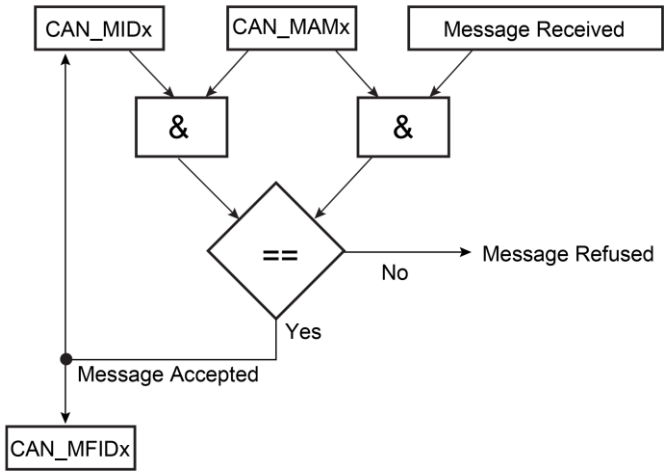
Postilaatikon tunnisterekisteri CAN_MIDx (Message ID) äärittelee lähetettävän viestin tunnisteen tai, yhdessä vastaanottomaskirekisterin CAN_MAMx (Message Acceptance Mask) kanssa, suodattimen vastaanotettaville viesteille. Molempien rekisterien rakenne on sama, ja se on esitetty taulukossa 23.

Taulukko 23. Rekisterien CAN_MIDx ja CAN_MAMx rakenne.

Bitti	**Nimi**	**Selitys**
0…17	MIDvB	Laajennettu tunniste
18…28	MIDvA	Standarditunniste

29	MIDE	Tunnistetyypin valinta
30…31	-	

Bittikenttä MIDvA määrittelee standardimittaisen tunnisteen. MIDvB määrittelee tunnisteen laajennetun osan. MIDE-bitti määrittelee, kumpi tunniste on käytössä. Arvolla 0 on käytössä vain standarditunniste, arvolla 1 käytössä on myös tunnisteen laajennettu osa.



Kuva 26. Viestien suodatusprosessi [17, s. 507].

Vastaanotettavien viestien suodatus tapahtuu siten, että saapuvan viestin tunnistetta ja CAN_MIDx-rekisterin tunnistekenttää verrataan molempia AND-operaatiolla maskirekisteriin CAN_MAMx. Jos molempien AND-operaatioiden tulos on sama, viesti läpäisee suodattimen. Kun viesti on läpäissyt vastaanottosuodattimen, kopioidaan sen tunniste rekisteriin CAN_MIDx. Vastaanottomaskin maskaamat bitit, eli ne vastaanotetun viestin tunnisteen bitit, jotka erottavat vastaanotetut viestit toisistaan, menevät peräkkäin tunnisteryhmärekisteriin CAN_MFIDx. (Message Family ID) Tämän rekisterin avulla voidaan helposti ryhmitellä vastaanotetut viestit esimerkiksi eri keskeytysrutiineja varten. Kuva 26 esittää kaavion viestien suodatusprosessista.

Suodatin määritellään siis kirjoittamalla CAN_MIDx-rekisteriin jokin tunniste halutun alueen sisältä ja määrittelemällä CAN_MAMx-rekisteriin, mitkä tunnisteen bitit ovat merkitseviä ja millä biteistä ei ole väliä. [17, s. 506–507.]

Postilaatikon tilatiedot

Postilaatikon tilatietoja varten on rekisteri CAN_MSRx (Message Status Register). Rekisterin rakenne on kuvattu taulukossa 24.

Taulukko 24. Postilaatikon CAN_MSRx-tilarekisterin rakenne.

Bitti	**Nimi**	**Selitys**
0…15	MTIMESTAMP0…15	Aikaleima
16…19	MDLC	Viestikehyksen datatavujen määrä
20	MRTR	Etäpyyntöbitti (RTR)
21	-	
22	MABT	Lähetys on keskeytetty
23	MRDY	Postilaatikko valmis
24	MMI	Viesti hylätty
25…31	-	

Tilarekisterin 16 alinta bittiä *MTIMESTAMP0…15* sisältävät aikaleiman, joka on otettu viimeisimmän lähetetyn tai vastaanotetun viestin yhteydessä.

Bittikenttä *MDLC* sisältää postilaatikon vastaanottopuskurissa olevan viestin datatavujen määrän.

Bitti *MRTR* sisältää vastaanottopuskurissa olevan viestin RTR-, eli etäpyyntöbitin.

MABT-bitti on asetettu, jos viimeisimmän viestin lähetys on keskeytetty.

MRDY-bitin arvo on yksi, kun postilaatikko on valmis ja sen viestipuskureista CAN_MDLx ja CAN_MDHx voi lukea tai niihin voi kirjoittaa. Jos arvo on 0, viestipuskurit on lukittu, eikä niihin voi kirjoittaa tai lukea.

MMI-bitti on asetettu, jos viimeisimmän viestin vastaanoton yhteydessä on hylätty viesti. Se, onko hylätty viimeisin vastaanotettu viesti vai vanha puskurissa ollut viesti, riippuu postilaatikon toimintatilasta. [17, s. 553.]

Postilaatikon ohjaus

Yksittäisen postilaatikon ohjaus tapahtuu rekisterillä CAN_MCRx. Sen rakenne on kuvattu taulukossa 25.

Taulukko 25. Postilaatikon CAN_MCRx-ohjausrekisterin rakenne

Bitti	Nimi	Selitys
0…15	-	
16…19	MDLC	Viestikehyksen datatavujen määrä
20	MRTR	Etäpyyntöbitti (RTR)
21	-	
22	MACR	Lähetyksen keskeytyskäsky
23	MTCR	Lähetyskäsky
24…31	-	

Bittikenttään *MDLC* määritellään lähetettävän viestin datatavujen määrä.

MRTR-bitti sisältää lähetettävän viestikehyksen RTR- eli etäpyyntöbitin arvon.

Asettamalla *MACR-bitti* voidaan viestin lähetys keskeyttää. Viestin lähettäminen tapahtuu asettamalla bitti *MTCR*. [17, s. 558–559.]

Postilaatikon datarekisterit

Lähetettävän tai vastaanotetun viestin dataa varten on kaksi rekisteriä. CAN_MDLx sisältää datan neljä alinta tavua ja CAN_MDHx neljä ylintä tavua. Tavut lähetetään ja vastaanotetaan vähiten merkitsevästä tavusta eniten merkitsevään, eniten merkitsevästä bitistä vähiten merkitsevään, eli seuraavassa järjestyksessä:

CAN_MDL[7:0], CAN_MDL[15:8], CAN_MDL[23:16], CAN_MDL[31:24],
CAN_MDH[7:0], CAN_MDH[15:8], CAN_MDH[23:16], CAN_MDH[31:24].
[17, s. 556–557.]

Kehysten vastaanottaminen

Ennen postilaatikon asettamista vastaanottotilaan on kyseisen postilaatikon keskeytys syytä deaktivoida asettamalla postilaatikon numeroa vastaava bitti rekisterissä CAN_IDR. Myös postilaatikon ohjausrekisteri CAN_MCRx pitää nollata ja itse postilaatikko deaktivoida viestisuodattimen asetuksen ajaksi. Deaktivointi tapahtuu kirjoittamalla 0 CAN_MMRx-rekisterin MOT-kenttään.

Tämän jälkeen määritellään viestisuodatin CAN_MIDx- ja CAN_MAMx-rekistereihin kuvan 26 avulla. Suodattimen määrittelemisen jälkeen asetetaan postilaatikko vastaanottotilaan kirjoittamalla arvo 1 postilaatikon toimintamoodirekisterin CAN_MMRx

bittikenttään MOT, tai tilaan ”vastaanotto ylikirjoituksella” kirjoittamalla bittikenttään arvo 2. Lopuksi voidaan halutessa aktivoida postilaatikon keskeytys asettamalla postilaatikon numeroa vastaava bitti rekisterissä CAN_IER.

Kun vastaanottotila on aktivoitu, CAN_MSR-tilarekisterin MRDY-bitti nollautuu automaattisesti. Kun uusi viesti saapuu, MRDY-bitti nousee ja tämä aiheuttaa keskeytyksen. Nyt viestin tunnisteen voi lukea rekisteristä CAN_MIDx, RTR- ja DLC-kentän rekisteristä CAN_MSRx ja datan rekistereistä CAN_MDLx ja CAN_MDHx. Nämä tiedot säilyvät rekistereissä, kunnes ne nollataan kirjoittamalla lähetyskäsky asettamalla CAN_MCRx-rekisterin MTCR-bitti. Tämä valmistaa postilaatikon seuraavan kehyksen vastaanottoa varten ja nollaa myös MRDY-signaalin.

Jos postilaatikko on vastaanottotilassa ja siihen saapuu uusi viesti, kun MRDY-bitti on ylhäällä, CAN-ohjain hylkää saapuvan viestin ja signaloi tästä asettamalla CAN_MSRx-tilarekisterin bitin MMI. Bitti nollaantuu, kun CAN_MSRx-rekisteriä seuraavaksi luetaan.

Jos postilaatikko on ”vastaanotto ylikirjoituksella” -tilassa ja siihen saapuu uusi viesti MRDY-bitin ollessa ylhäällä, saapuva viesti kirjoittaa vanhan viestin yli ja ohjain asettaa MMI-bitin merkkinä tästä. On mahdollista, että uusi viesti saapuu juuri, kun datarekisterien lukeminen on kesken. Tällöin saattaa käydä niin, että luetut CAN_MDLx- ja CAN_MDHx-rekisterien arvot kuuluvat eri viesteihin. Tämän vuoksi, mikäli postilaatikko on ylikirjoitustilassa, on syytä lukea CAN_MSRx-rekisterin MMI-bitin arvo ennen ja jälkeen datarekisterien lukemista. Jos bitin arvo muuttuu, on uusi viesti saapunut kesken lukemisen ja data on luettava uudestaan.

5.3.1 Kehysten lähettäminen

Ennen postilaatikon asettamista lähetystilaan on postilaatikon keskeytys deaktivoitava asettamalla postilaatikon numeroa vastaava bitti rekisterissä CAN_IDR. Lisäksi ohjausrekisteri CAN_MCRx tulee nollata ja postilaatikko deaktivoida kirjoittamalla 0 CAN_MMRx-rekisterin MOT-kenttään. Tämän jälkeen määritellään lähetettävän viestin tunniste rekisteriin CAN_MIDx ja asetetaan postilaatikko lähetystilaan kirjoittamalla 3 CAN_MMRx-rekisterin kenttään MOT. Samalla voidaan määritellä postilaatikon prioriteetti CAN_MMRx-rekisterin kenttään PRIOR. Pienempi arvo tarkoittaa tässä suurempaa prioriteettia. Lopuksi voidaan halutessa aktivoida postilaatikon keskeytys asettamalla postilaatikon numeroa vastaava bitti rekisterissä CAN_IER.

Kun lähetystila on aktivoitu, nousee CAN_MSRx-rekisterin MRDY-signaali automaattisesti. Kun MRDY-bitti on ylhäällä, voidaan lähetettävän viestin data kirjoittaa rekistereihin CAN_MDLx ja CAN_MDHx. Viesti lähetetään määrittelemällä ensin datatavujen määrä ohjausrekisterin CAN_MCRx kenttään MDLC ja asettamalla sitten CAN_MCRx-ohjausrekisterin bitti MTCR. Viestin voi myös lähettää asettamalla postilaatikon numeroa vastaava bitti CAN_TCR-ohjausrekisterissä. Tämän rekisterin avulla voidaan myös lähettää useampi kehys yhtä aikaa. Tällöin viestit lähetetään peräkkäin prioriteettijärjestyksessä. Postilaatikon prioriteetti on määritelty CAN_MMRx-rekisterin kentässä PRIOR. Jos kahdella lähettävällä postilaatikolla on sama prioriteetti, pienemmän järjestysnumeron omaava postilaatikko saa lähettää ensin.

RTR- eli etäpyyntökehys lähetetään asettamalla datatavujen määrän sijaan CAN_MCRx-rekisterin MRTR-bitti. Jos postilaatikko on lähetystilassa, pitää pyydettävä viesti ottaa vastaan toisella vastaanottotilaan määritellyllä postilaatikolla. On myös mahdollista hoitaa etäpyyntökehyksen lähettäminen ja pyydetyn viestin vastaanottaminen automaattisesti yhdellä postilaatikolla, asettamalla postilaatikko kulutustilaan. Kulutustilaa ei käsitellä tässä sen enempää.

Viestin lähetyksen voi keskeyttää asettamalla CAN_MCRx-rekisterin MACR-bitin. Useamman postilaatikon lähetyksen voi keskeyttä yhdellä kerralla asettamalla postilaatikoiden numeroita vastaavat bitit rekisterissä CAN_MACR.

Jos lähetetty viesti joutuu kilpavaraustilanteeseen ja häviää sen, CAN-ohjain lähettää viestin automaattisesti uudestaan niin monta kertaa, että viesti tulee kokonaan lähetetyksi. Tämän ominaisuuden voi ottaa pois käytöstä asettamalla DRPT-bitin CAN_MR-rekisterissä. Tällöin CAN-ohjain keskeyttää viestin lähettämisen, jos se ei onnistu ensimmäisellä kerralla. Samalla ohjain asettaa MABT-bitin rekisterissä CAN_MSRx.

6 CiA447-yhteensopiva CANopen-rajapinta AT91SAM7X-mikrokontrollerille

Insinöörityön käytännön osana suunniteltiin CiA447-yhteensopiva CANopen-rajapinta ja rajapinnan testausohjelma Semelin TM6000-taksamittarin TM206-keskusyksikölle. Rajapinta sisältää ISO-11898 -yhteensopivan CAN-ajurin, joka ei tue laajennettuja viestikehyksiä, CiA447-yhteensopivan CANopen-pinon, joka tukee NMT-, Heartbeat- ja SDO-protokollia sekä ne objektikirjaston merkinnät, jotka on määritelty pakollisiksi CANopen-standardissa ja CiA447-sovellusprofiilissa. Rajapinta sisältää vain ne toiminnot,

jotka ovat välttämättömiä asiakkaan tarvitsemien ominaisuuksien toteuttamiseksi. Esimerkiksi CiA447-profiilin määrittelemiä virransäästötoimintoja ja tukea PDO-viesteille ei tällä kertaa toteutettu.

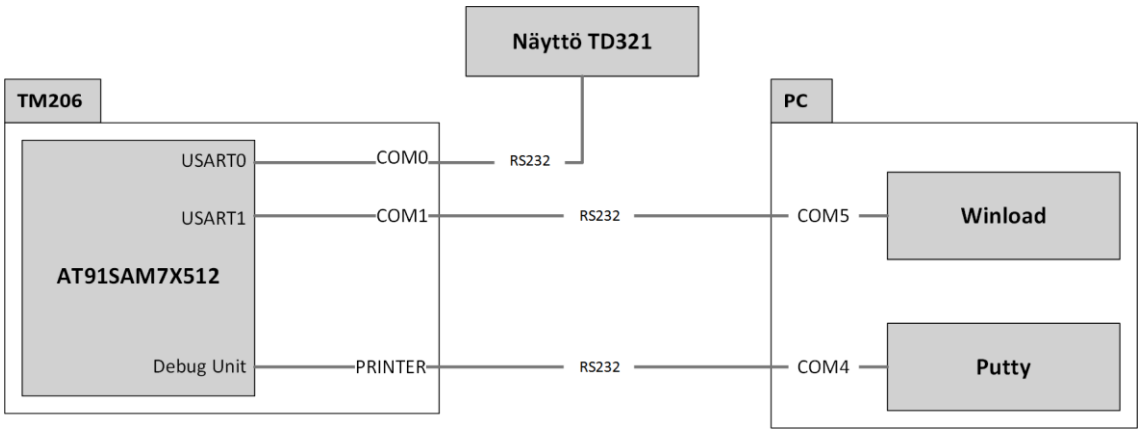
Testausohjelma sisältää terminaalipohjaisen käyttöliittymän sarjaportin kautta, ja sen avulla voidaan skannata väylässä olevan CiA447-yhteensopivan IVN-yhdyskäytävän objektikirjasto ja näin tunnistaa, mitä objektikirjaston merkintöjä yhdyskäytävä tukee. Lisäksi ohjelmalla voidaan lukea, tulkita ja muuttaa joidenkin yksittäisten objektikirjaston merkintöjen arvoja.

6.1 Työkalut ja vianetsintä

Ohjelmointi suoritettiin C-kielellä, ja ohjelmistokehitykseen käytettiin IAR Embedded Workbench for ARM -kehitysympäristön ilmaisversiota, jossa koodin koko on rajoitettu 32 kilotavuun. Kehitysympäristön versionumero on 6.60.1.5104.

Taksamittarin TM206-keskusyksikössä on Semelin oma käynnistyslataaja, ja mikrokontrollerin ohjelmointia varten Semellillä on oma Winloader-ohjelma. Uuden ohjelman lataaminen tapahtuu taksamittarin COM1-sarjaportin eli mikrokontrollerin USART1-portin kautta.

Vianenetsintää ja käyttöliittymää varten taksamittari kytkettiin mikrokontrollerin Debug Unit -sarjaportin eli taksamittarin PRINTER-portin kautta PC:n sarjaporttiin. Varsinaisia mikrokontrollerin vianetsintäominaisuuksia ei käytetty, vaan portti määriteltiin yleiseksi sarjaportiksi ja siihen otettiin yhteys Putty-terminaaliohjelmalla. PRINTER-portti toimii myös taksamittarin käynnistyslataajan diagnostiikkaporttina asetuksilla 19200 bps, 8 databittiä, ei pariteettia, 1 stop-bitti. Käynnistyslataajan diagnostiikkatilaan pääsee poistamalla varsinaisen ohjelman laitteen muistista ja käynnistämällä laitteen uudelleen. Diagnostiikkatilan avulla sarjaporttikytkentöjen toimivuus voitiin tarkistaa alussa, kun varsinaisen ohjelman ajoitusparametrien kanssa oli ongelmia. Kuvassa 27 on kaavio laitteiston kytkennöistä ohjelmistokehityksen aikana.



Kuva 27. Laitteiston kytkennät ohjelmistokehityksen aikana.

Joitakin kertoja työn aikana kävi niin, että Winloader jätti ohjelman kirjoittamatta. Tästä ei kuitenkaan tullut mitään virheilmoitusta ja asia jäi huomaamatta. Tämän vuoksi taksamittarin COM0-sarjaporttiin kytkettiin taksamittarin näyttöyksikkö, johon ohjelma kirjoittaa käännöspäivämäärän ja -ajan. Tällä tavoin voidaan saada näkyvä varmistus siitä, että uuden ohjelman lataus laitteeseen on onnistunut. Näyttöyksikön sarjaportin parametrit ovat 9600 bps, 8 databittiä, ei pariteettia, 1 stop-bitti.

6.2 Mikrokontrollerin asetukset

Mikrokontrollerin käynnistyskoodi, muistialueiden määrittelyt, rekisterien nimet ja muut alemman tason asetukset löytyivät suoraan IARin mukana tulleista Atmelin koodiesimerkeistä, at91sam7x-ek-kansiosta. Nämä tiedostot ovat AT91SAM7X512.h, chip.h, board.h, board_lowlevel.h, board_memories.h, board_memories.c ja board_cstartup.s. Tiedostoihin piti tehdä vain kaksi muutosta. Tiedostossa board.h muutettiin BOARD_MAINOSC-vakio vastaamaan taksamittarin oskillaattorin kellotaajuutta 11 059 200 Hz ja BOARD_MCK-vakio arvoon 48 000 000.

6.3 Pääohjelma ja yleiset funktiot

Pääohjelma sisältää eri laitteiden asetukset ja pääohjelmasilmukan lähdekooditiedostossa main.c.

Asetukset ja keskeytykset

Pääohjelman alussa määritellään tarvittavat muuttujat, eri lisälaitteiden asetukset ja keskeytykset, jonka jälkeen siirrytään varsinaiseen pääohjelmasilmukkaan.

Debug Unit ja vianetsintätasot

Ensin määritellään Debug Unit -sarjaportti käyttöön nopeudella 115 200. Aluksi käytössä oli sama nopeus kuin taksamittarin käynnistyslataajalla, 19 200 bps, mutta tämän todettiin aiheuttavan liian suuria viiveitä, joten nopeutta nostettiin. Portin määrittelemiseen käytetään IARin mukana tullutta Atmelin esimerkkikoodia trace, joka ylikuormittaa C-kirjaston funktiot fputc() ja fputs() siten, että printf()-funktion ulostulo ohjautuu Debug Unit -porttiin. Lisäksi trace määrittelee viisi eritasoista vianetsintätulostetta, jotka voidaan aktivoida ja deaktivoida koodissa määrittelemällä vianetsintätaso vakioon TRACE_LEVEL. Vianetsintätaso voi olla välillä 1…5 ja vianetsintätulosteet, joiden numero on yhtä suuri tai pienempi kuin vianetsintätaso, ovat aktiivisia. Vianetsintäfunktiot ja niiden numerot ovat TRACE_DEBUG(5), TRACE_INFO(4), TRACE_WARNING(3), TRACE_ERROR(2) ja TRACE_FATAL(1). Vianetsintäfunktiot toimivat samaan tapaan kuin printf()-funktio.

Näytön ohjaus

Seuraavaksi pääohjelma määrittelee USART0-sarjaportti taksamittarin näyttöyksikköä varten. Näyttöyksikköön kirjoitusta varten saatiin Semeliltä valmiit funktiot. Funktio sendText() kirjoittaa parametrina annetun puskurin text sisällön näytölle. Tämän lisäksi tehtiin pieni funktio sendTime(), joka valmistelee puskurin, joka sisältää päivämäärän ja kellonajan peräkkäin ja kirjoittaa sen sitten näytölle funktiolla sendText().

Timestamp-laskuri ja viiveet

ConfigurePit()-funktio on Atmelin esimerkeistä lainattu koodinpätkä, joka asettaa mikrokontrollerin tuottamaan keskeytyksen kerran millisekunnissa. Tämä keskeytys on viiveiden laskemista ja erilaisia laskureita varten. Tämän keskeytyksen käsittelyfunktio ISR_Pit() on myös Atmelilta ja se inkrementoi timestamp-nimistä globaalia laskurimuuttujaa. Wait() on viivefunktio, joka odottaa parametrina annetussa delay-muuttujassa määritetyn ajan millisekunteina, käyttäen apunaan timestamp-laskuria.

Lopuksi, ennen varsinaista pääohjelmasilmukkaa määritellään CAN-ohjaimen asetukset funktiolla CAN_Init() ja CANopen-pinon asetukset funktiolla CANOPEN_Init().

CAN_Init() ottaa parametriksi CAN-ohjaimen tiedonsiirtonopeuden kilotavuina sekunnissa ja palauttaa 0, jos ohjaimen käynnistämisessä on jokin ongelma. Tällöin pääohjelma antaa virheilmoituksen.

Pääohjelmasilmukka sisältää kaksi osaa. Ensimmäinen osa on CANopen-prosessi eli funktio CANOPEN_Process(), joka hoitaa kaiken CANOPEN-standardiin liittyvän tiedon vastaanoton ja käsittelyn.

Toinen osa on ohjelman käyttöliittymä. Se odottaa, että käyttäjä syöttää Debug Unit -porttiin merkkijonon ja tallentaa sen muuttujaan. Merkkijono voi päättyä numeroon, joka tallennetaan eri muuttujaan. Nämä muuttujat annetaan parametreina CiA447-testausfunktiolle CIA447_tester().

6.4 CAN-ajuri

CAN-ajuri suunniteltiin IARin mukana tulevan Atmelin CAN-esimerkin pohjalta. Alkuperäisestä koodista otettiin mallia lähinnä alustusfunktiossa ja aikaparametrien laskemisfunktiossa. Muut funktiot ovat pääosin omaa käsialaa. Ajuri on ISO 11898 -yhteensopiva, mutta se tukee ainoastaan viestikehyksen standarditunnistetta. CAN-ajurin ohjelmatiedosto on can.c ja header-tiedosto can.h.

Alkuarvojen asettaminen ja CAN-ohjaimen käynnistäminen

CAN-ohjaimen alkuarvot ja keskeytykset asetetaan ja ohjain käynnistetään funktiossa CAN_Init(), joka ottaa parametrina halutun tiedonsiirtonopeuden ja palauttaa 1 jos CAN-ohjain käynnistyy kunnolla ja 0, jos käynnistyksessä on ongelma.

CAN_Init() funktion alussa nollataan muuttuja mailbox_idle, joka sisältää postilaatikoiden valmiusliput. Jos muuttujan postilaatikon numeroa vastaava bitti on 0, lähetys tai vastaanotto on kesken. Jos se on 1, postilaatikko on valmis ottamaan vastaan käskyjä.

Seuraavaksi nollataan vastaanottopuskuri funktiolla CAN_ClearReceiveBuffer() ja määritellään CAN-ohjaimen tiedonsiirtopinnit ja pinni CAN-sovittimen virransäästötilan kytkemiseen, käynnistetään CAN-ohjaimen kello ja määritellään CAN-ohjaimen keskeytys ja ajoitusparametrit. CAN-kesketyksen prioriteetiksi määritellään 7 ja käsittelyrutiiniksi funktio CAN_Handler.

CAN-ohjaimen ajoitusparametrien asettamiseen käytetään CAN-ohjaimen asetuksissa CAN_BaudRateCalculate()-funktiota, joka löytyi valmiina IARin mukana tulleista Atmelin esimerkeistä. CAN_BaudRateCalculate() ottaa parametreina CAN-ohjaimen rekisterien alkuosoitteen ja halutun tiedonsiirtonopeuden. CAN-ohjaimen ajoitusparametrit pitäisi laskea väylän pituuden sekä lähettimien ja vastaanottimien viiveiden perusteella, mutta koska tiedonsiirto toimi CAN_BaudRateCalculate()-funktion laskemilla arvoilla sekä simulaattorin kanssa että Mercedes Benzin E-sarjan taksiautossa, ei ollut syytä alkaa laskea parametreja itse.

Seuraavaksi CAN_Init()-funktiossa nollataan kaikki postilaatikot eli deaktivoidaan kaikkien postilaatikoiden keskeytykset ja nollataan niiden ohjausrekisterit. Lopuksi CAN-ohjain käynnistetään funktiolla CAN_Synchronisation().

CAN_Synchronisation() aktivoi CAN-ohjaimen keskeytyksen, aktivoi CAN-ohjaimen ja jää sen jälkeen odottamaan WAKEUP-keskeytystä. AT91C_CAN_TIMEOUT-vakio määrittelee, kuinka monta kellojaksoa keskeytystä odotetaan. Jos keskeytystä ei määritellyn aikajakson sisällä tule, CAN_Synchronisation() palauttaa arvon 0. Muussa tapauksessa se palauttaa arvon 1. CAN_Init() palauttaa saman arvon kuin CAN_Synchronisation().

Vastaanottopuskuri

Vastaanottopuskuri CAN_ringbuffer[] on rengaspuskuri, jonka alkiot ovat CAN_buffer-tietueita. CAN_buffer on määritelty tiedostossa can.h, ja se sisältää vastaanotetun kehyksen tunnisteen, datan määrän ja datan. Rengaspuskurin koko määritellään vakiolla CAN_RINGBUFFER_SIZE, ja sen oletuskoko on 8.

Vastaanottopuskuriin kirjoitetaan CAN-ohjaimen keskeytysrutiinissa ja sieltä luetaan funktiolla CAN_ReadFromReceiveBuffer(), joka ottaa parametriksi osoitteen paikalliseen puskuriin. Funktio yksinkertaisesti kopioi ympyräpuskurin vanhimman alkion sisällön paikalliseen puskuriin ja inkrementoi lukuosoitinta. Funktio palauttaa arvon yksi, jos vastaanottopuskurista lukeminen onnistui, ja arvon nolla, jos puskuri oli tyhjä. Vastaanottopuskuri nollataan funktiolla CAN_ClearReceiveBuffer(), joka yksinkertaisesti nollaa molemmat osoittimet ja ylivuotolipun.

CAN_ringbuffer[]-taulukon lisäksi puskuri tarvitsee toimiakseen kaksi osoitinta ja ylivuotolipun. CAN_rb_w on kirjoitusosoitin, joka kertoo, mihin puskurin alkioon seuraavaksi

vastaanotetun kehyksen tiedot kirjoitetaan. CAN_rb_r on lukuosoitin ja kertoo seuraavan luettavan alkion numeron. Jos osoittimet ovat samanarvoiset ja ylivuotolippu on 0, puskuri on tyhjä. Ylivuotolippu CAN_rb_overflow asetetaan CAN-ohjaimen keskeytysrutiinissa, kun rengaspuskuriin kirjoittamisen ja kirjoitusosoittimen inkrementoinnin jälkeen luku- ja kirjoitusosoittimien arvot ovat samanarvoiset. Tämä tarkoittaa sitä, että mikäli rengaspuskurista ei lueta ennen kuin siihen seuraavalla kerralla kirjoitetaan, tulee tapahtumaan ylivuoto.

Postilaatikoiden käyttö

Postilaatikoiden konfiguroimiseen ja käyttöön on määritelty neljä funktiota. ResetMailboxRegisters() ottaa parametrina postilaatikon numeron, deaktivoi postilaatikon keskeytyksen ja nollaa rekisterit CAN_MCRx, CAN_MMRx, CAN_MAMx, CAN_MIDx, CAN_MDLx, CAN_MDHx ja CAN_MCRx.

ResetAllMailbox() on yksinkertainen funktio, joka nollaa kaikki postilaatikot kutsumalla ResetMailBoxRegisters()-funktiota kahdeksan kertaa. CAN_Receive() asettaa postilaatikon vastaanottotilaan, ja CAN_SendFrame() lähettää viestikehyksen.

Kehysten vastaanottaminen

CAN_Receive()-funktiolla postilaatikko määritellään vastaanottotilaan ja sille määritellään vastaanottomaski. Funktio saa parametreina tunnisteen id, tunnistemaskin mam, postilaatikon numeron mailbox_num ja overwrite-lipun. Ennen postilaatikon konfigurointia odotetaan, että postilaatikko vapautuu edellisestä toiminnosta, eli kunnes postilaatikon valmiuslippu muuttujassa mailbox_idle on yksi, jonka jälkeen postilaatikon keskeytys deaktivoidaan ja ohjausrekisterit nollataan. Sitten tunniste kopioidaan postilaatikon tunnisterekisteriin CAN_MIDx ja tunnistemaski vastaanottomaskirekisteriin CAN_MAMx. Jos overwrite-lippu on asetettu, postilaatikko asetetaan tilaan ”vastaanotto ylikirjoituksella,” muutoin postilaatikko asetetaan tavalliseen vastaanottotilaan. Lopuksi aktivoidaan postilaatikon keskeytys, ja postilaatikko on valmis vastaanottamaan viestejä.

Itse vastaanottaminen ja vastaanotettujen kehysten puskurointi tapahtuu CAN-ohjaimen keskeytysrutiinissa CAN_Handler(). Keskeytysrutiinin vastaanotto-osassa tarkistetaan ensin vastaanottopuskurin tila. Jos kirjoitusosoitin ja lukuosoitin ovat yhtä suuret, tapahtuu ylivuoto, josta annetaan virheilmoitus. Tämän jälken vastaanotetun

kehyksen tunniste, datatavujen määrä ja data kopioidaan rekistereistä CAN_ringbuffer[]-puskurin seuraavaan alkioon. Lopuksi asetetaan CAN_MCRx-rekisterin MTCR-bitti ja aktivoidaan postilaatikon keskeytys, jotta postilaatikko on valmiina seuraavan kehyksen vastaanottamista varten.

Kehysten lähettäminen

Kehysten lähettäminen tapahtuu funktiolla CAN_SendFrame(). Funktio saa parametreina tunnisteen id, datan määrän dlc, datakentät data_low ja data_high, postilaatikon numeron mailbox_num ja prioriteetin mailbox_priority. Ensimmäiseksi odotetaan postilaatikon vapautumista kuten vastaanottofunktiossakin. Kun postilaatikko on vapaa, sen keskeytys deaktivoidaan ja ohjausrekisterit nollataan. Tämän jälkeen kopioidaan tunniste rekisteriin CAN_MIDx ja asetetaan postilaatikko lähetystilaan. Sitten aktivoidaan postilaatikon keskeytys ja jäädään odottamaan postilaatikon valmistumista, eli sitä, että CAN_MSRx-rekisterin MRDY-lippu nousee, jonka jälkeen kopioidaan parametreina saadut data-muuttujat datarekistereihin CAN_MDLx ja CAN_MDHx, kopioidaan datan pituus -parametri ohjausrekisteriin CAN_MCRx ja asetetaan lähetysbitti MTCR samassa rekisterissä. Lopuksi nollataan postilaatikon valmiuslippu muuttujassa mailbox_idle merkiksi siitä, että postilaatikko on varattu.

CAN-ohjain tuottaa postilaatikkokeskeytyksen, kun kehyksen lähetys on valmis. Keskeytysrutiinin lähetysosassa ei tehdä muuta kuin merkitään postilaatikko vapaaksi asettamalla valmiuslippu muuttujassa mailbox_idle.

CAN-ohjaimen keskeytysrutiini

CAN-ohjaimen eri keskeytykset käsitellään kaikki samassa keskeytysrutiinissa. Keskeytyksen syy luetaan rekisteristä CAN_SR, jonka rakenne on taulukossa 21, sivulla 59. Lähetys- ja vastaanottokeskeytykset asettavat saman bittin CAN_SR-rekisterissä, mutta ne pystytään erottamaan toisistaan lukemalla postilaatikon tila rekisteristä CAN_MMRx.

CAN-ohjaimen keskeytysrutiinissa on viisi osaa eli alkuosa, CAN-ohjaimen käynnistys, vastaanotto, lähetys ja virheidenkäsittely. Alkuosa on kaikille keskeytyksille yhteinen, ja siinä luetaan keskeytyksen syy status-muuttujaan maskaamalla tilarekisteri CAN_SR keskeytysmaskirekisterillä CAN_IMR ja deaktivoidaan käsiteltävä keskeytys kirjoittamalla status-muuttuja rekisteriin CAN_IDR.

Jos CAN_SR-rekisterin WAKEUP-bitti on asetettu, CAN-ohjain on herännyt virransäästötilasta, tai se on käynnistetty ensimmäisen kerran. Tässä tapauksessa asetetaan muuttujalle test_can arvo AT91C_TEST_OK. CAN_Synchronise()-funktio odottaa ohjaimen käynnistymistä tarkkailemalla tätä muuttujaa.

Jos jokin postilaatikoiden keskeytyslipuista rekisterissä CAN_SR on asetettu, käydään for-silmukassa läpi kaikkien postilaatikoiden MRDY-bitti CAN_MSRx-rekisterissä. Näin kaikki keskeytykset tulevat käsitellyiksi, vaikka niitä olisikin tapahtunut useampi ennen kuin keskeytysrutiini on päässyt tähän kohtaan. Jos postilaatikon MRDY-bitti on asetettu, eli vastaanotto tai lähetys on valmis. Tämän jälkeen luetaan postilaatikon toimintatila CAN_MMRx-rekisterin MOT-kentästä, jonka perusteella päätellään, onko kyseessä vastaanotto- vai lähetyskeskeytys.

Jos jokin CAN_SR-rekisterin biteistä 16…19 tai 22…28 on asetettu, on tapahtunut virhe, CAN-ohjain on siirtynyt virheidenhallintatilasta toiseen tai virhelaskurin varoitusraja on ylitetty. Tällöin siirrytään virheidenkäsittelyrutiiniin kutsumalla funktiota CAN_ErrorHandling();

Virheidenkäsittely

CAN-ohjaimen virheet käsitellään keskeytyksen sisällä funktiossa CAN_ErrorHandling(). Funktio ottaa parametrina muuttujan status, joka on kopio CAN_SR-rekisteristä ja maskattu siten, että siinä näkyvät vain keskeytyksen aiheuttamat liput. Aluksi funktio tarkistaa CAN_SR-rekisteristä, missä virhetilassa CAN-ohjain on ja onko virhelaskurin varoitusraja ylitetty ja tulostaa tiedon näistä Debug Unit -porttiin. Lisäksi, jos ohjain on väylä irti -tilassa, se käynnistetään uudelleen. Virheet luetaan status-muuttujasta ja niistä tulostetaan virheilmoitus.

6.5 CANopen-solmu

CANopen-solmu tukee SDO-protokollaa, Heartbeat-protokollaa, NMT-orja-protokollaa ja sisältää CANopen-standardin ja CiA447-profiilin pakolliseksi määrittelemät objektikirjaston merkinnät.

SDO-viestien lähettämiseen on funktiot CANOPEN_ReadSDO() ja CANOPEN_WriteSDO(). CANopen-viestien lukeminen CAN-vastaanottopuskurista, niiden

käsitteleminen ja muut toistuvat toiminnot tapahtuvat funktiossa CANOPEN_Process(). Funktiota on syytä kutsua pääohjelmasta riittävän usein, jottei tapahdu CAN-vastaanottopuskurin ylivuotoa.

CANopen-solmun ohjelmakooditiedosto on canopen.c, ja header-tiedosto on canopen.h.

Objektikirjasto, muuttujat ja vakiot

Objektikirjasto sisältää merkinnät 0x1000, 0x1001, 0x1003, 0x1017, 0x1018 ja 0x6000. Nämä objektit ja niiden sisältö ja niihin liittyvät muuttujat on listattu taulukossa 26. Laitetyypin arvo 0x101BF kertoo, että käytössä oleva laiteprofiilin numero on 447 ja sen versio on 2.0. Virhelista on lista virherekisterin aikaisemmista arvoista. Tunnisteobjektin vakioiden arvot on arvottu. Taksamittari ei sisällä CiA447-profiilin mukaisia virtuaalilaitteita, joten Virtuaalilaitteet 1 -alaobjektin arvo on 0.

Taulukko 26. CANopen-solmun objektikirjaston merkinnät

Objekti	**Alaindeksi**	**Nimi**	**Muuttuja tai vakio**
0x1000	0	Laitetyyppi	0x101BF
0x1001	0	Virherekisteri	CANOPEN_error_register
0x1003	0	Virhelista	255 alaindeksiä
	x		CANOPEN_pre_def_error[x]
0x1017	0	Heartbeat-tuottajan aika	CANOPEN_Heartbeat_time
0x1018	0	Tunnisteobjekti	4 alaindeksiä
	1	*Toimittajanumero*	VENDOR_ID
	2	*Tuotekoodi*	PRODUCT_CODE
	3	*Revisionumero*	REVISION_NUM
	4	*Sarjanumero*	SERIAL_NUM
0x6000	0	Virtuaalilaitteet	1 alaindeksi
	1	*Virtuaalilaitteet 1*	0

Muita CANopen-solmun käyttämiä muuttujia ovat CANOPEN_Heartbeat_start, jota käytetään timestamp-muuttujan kanssa ajoittamaan Heartbeat-viestien lähetys ja CANOPEN_IVN_Heartbeat_start, jota käytetään IVN-yhdyskäytävän Heartbeat-viestien valvomiseen. Lisäksi on määritelty taulukot SDO_receive_request[], SDO_receive_response_id[], SDO_send_request_id[] ja SDO_send_response_id[], jotka sisältävät jokaisen 15 kommunikaatiokanavan CAN-kehyksien tunnisteet. Nämä tunnisteet riippuvat solmun CAN-id-tunnisteesta, ja ne lasketaan CANopen-solmun alustuksen yhteydessä.

LSS_param on LSS_parameters-tyyppinen tietue, joka on määritelty tiedostossa canopen.h. Se on tietynlainen puskuri, jota käytetään LSS Fastscan -menetelmän yhteydessä. LSS-viestien eri bittikentät puretaan siihen helpommin käytettävään muotoon funktiolla LSS_ReadParameters().

NMT-tilat ja niitä vastaavat numerot Heartbeat-viestissä on tallennettu vakioihin NMT_STATE_BOOTUP, NMT_STATE_STOPPED, NMT_STATE_OPERATIONAL, NMT_STATE_PRE_OP.

6.5.1 CANopen-solmun käynnistys

CANopen-solmu alustetaan ja käynnistetään funktiolla CANOPEN_Init(). Tämä funktio nollaa aluksi CANopen-virherekisterin ja -virhelistan, asettaa Heartbeat-tuottajan ajan CiA447-profiilin määrittelemään oletusarvoon 200, nollaa kaikki postilaatikot ja tyhjentää CAN-vastaanottopuskurin.

LSS Fastscan

Seuraavaksi solmulle pitää saada CAN-id-tunniste. Se hankitaan LSS Fastscan -menetelmällä kutsumalla funktiota LSS_Fastscan().

LSS_Fastscan()-funktiota varten luotiin tietuetyyppi LSS_parameters, joka sisältää muuttujina kaikki LSS-viestin bittikentät. (Katso Kuva 24, sivu 46) Tämän lisäksi funktio käyttää CAN-viestien vastaanottamiseen ja välittämiseen LSS_ReadParameters()-funktiolle paikallista CAN_buffer-tyyppistä tietuetta buffer. Funktio käyttää myös muutamaa apumuuttujaa ja lisäksi toimittajanumero-, tuotenumero-, revisionumero- ja sarjanumerovakiot kopioidaan taulukkoon LSS_ID[4], jotta niitä on helpompi käyttää silmukassa.

Aluksi postilaatikko 0 asetetaan lähetystilaan ja viestisuodatin määritellään siten, että vain viestit, joiden tunniste on 0x7E5, vastaanotetaan. Tämän jälkeen siirrytään silmukkaan, jossa varsinainen Fastscan-menetelmä tapahtuu. Silmukan alussa odotetaan, että CAN-ohjain vastaanottaa kehyksen, kopioidaan vastaanotettu kehys vastaanottopuskurista paikalliseen puskuriin buffer ja annetaan puskurin osoite parametrina funktiolle LSS_ReadParameters(), joka purkaa datan CAN-viestikehyksestä

LSS_parameters-tyyppiseen tietueeseen LSS_param. Tästä eteenpäin funktio toimii, kuten LSS Fastscan -menetelmästä on luvussa 4.3 kirjoitettu.

Kun solmu on tunnistettu LSS_ID:n perusteella, siirrytään konfigurointiosioon, jossa solmu saa solmutunnisteen ja se käynnistetään. Tämä osio on eriytetty omaan funktioonsa koodin selkeyttämiseksi, koska se ei tarkalleen ottaen kuulu LSS Fastscan -menetelmään, vaan tapahtuu sen jälkeen. Kutsuttavan funktion nimi on LSS_Config(), ja sen lopussa solmu on LSS-tilassa odotus (waiting) ja NMT-tilassa esitoiminnallinen (pre-operational).

Vastaanottavat postilaatikot

Kun solmu on käynnistetty, asetetaan yksi postilaatikko vastaanottamaan NMT-viestejä, joiden tunniste on 0. Toinen postilaatikko vastaanottaa SDO-pyyntöjä ja kolmas SDO-vastauksia kaikilta lisälaiteväylän solmuilta. Näiden kahden postilaatikon viestisuodatin lasketaan solmutunnisteen perusteella. Laskukaavat löytyvät kappaleesta 4.7. Lisäksi määritellään yksi postilaatikko vastaanottamaan IVN-yhdyskäytävän Heartbeat-viestejä tunnisteella 0x701. Yhdyskäytävän Heartbeat-viestien valvominen on vapaaehtoista, joten tämän postilaatikon voisi haluttaessa jättää määrittelemättä.

SDO-kommunikaatiokanavien kehysten tunnisteet

CANopen-solmun käynnistysfunktion lopuksi lasketaan kehysten tunnisteet lähetettäville ja vastaanotettaville SDO-viesteille ja tallennetaan ne taulukoihin myöhempää käyttöä varten funktiolla CANOPEN_SetSDOTables(). SDO_receive_request_id[16] sisältää vastaanotettavien pyyntöjen, SDO_receive_response_id[16] vastaanotettavien vastausten, SDO_send_request_id[16] lähetettävien pyyntöjen ja SDO_send_response_id[16] lähetettävien vastausten tunnisteet. Solmun, josta kehys vastaanotetaan tai johon se lähetetään, solmutunniste on taulukon alkion numero plus yksi.

6.5.2 CANOPEN_SDO_data

CANOPEN_SDO_data on tietuetyyppi, joka luotiin SDO-tiedonsiirtoa varten. Se sisältää muuttujat id, cs, s, e, n, index, subindex, data ja updated. Tämän tyyppisiä tietueita

on ohjelmistossa vain yksi, globaali tietue SDO_data. CANOPEN_SDO_data-tyyppi on määritelty ja SDO_data esitelty tiedostossa globals.h.

Tietueen muuttujat vastaavat kuvassa 10 määriteltyjä SDO-viestin bittejä ja bittikenttiä ja sitä käytetään lähetettäessä SDO-kirjoitus- tai lukupyyntöjä. Kohteen solmutunniste, objektin indeksi, alaindeksi ja mahdolliset muut asetukset kirjoitetaan tietueeseen ja updated-lippu nollataan. CANOPEN_Process()-funktio asettaa updated-lipun, kun se lukee CAN-viestipuskurista vastauksen SDO-pyyntöön. Samalla puretaan viestin sisältö muihin tietueen muuttujiin.

6.5.3 SDO-lukeminen

Lukeminen toisen solmun objektikirjastosta tapahtuu SDO-luku-palvelulla, funktiolla CANOPEN_ReadSDO(). Funktiolle annetaan parametreina kohdesolmun solmutunniste sekä luettavan objektin indeksi ja alaindeksi. Nämä muuttujat kopioidaan funktion alussa SDO_data-tietueeseen ja tietueen päivityslippu updated nollataan.

Seuraavaksi lähetetään varsinainen lukupyyntö funktiolla CAN_SendFrame(). Funktion kehyksen tunniste -parametri haetaan taulukosta SDO_send_request_id[]. Muilta osin seurataan SDO-alustusviestin rakennetta, joka näkyy kuvassa 10. Komentotarkenteeksi cs asetetaan 2.

Tämän jälkeen odotetaan 50 millisekuntia, jotta SDO-palvelimella on varmasti aikaa vastata. Sitten ajetaan CANOPEN_Process()-funktio, joka muun muassa kopioi mahdollisen SDO-vastauksen sisällön SDO_data-tietueeseen. Jos tietueen updated-lippu on tämän jälkeen 0, SDO-palvelimelta ei ole saatu vastausta ja funktio tulostaa virheilmoituksen ja palauttaa arvon NULL.

Updated-lipun ollessa CANOPEN_Process()-funktion jälkeen 1, voidaan vastauksen sisältö lukea tietueesta SDO_data. Jos komentotarkenne cs on 2, lukupyyntöön on saatu vastaus. Tällöin CANOPEN_ReadSDO()-funktio palauttaa vastaanotetun datan tietueesta SDO_data, paitsi jos vastaanotetun viestin bitti e on 0. Siinä tapauksessa pyydetty objekti on useaosainen ja sen lukeminen vaatii segmentoitua tiedonsiirtoa. Segmentoitua tiedonsiirtoa ei tueta, joten funktio tulostaa ilmoituksen tästä ja palauttaa SDO_data-tietueen data-muuttujassa olevan tiedon, joka tässä tapauksessa merkitsee objektiin tallennettujen segmenttien määrää.

Jos vastaanotetun viestin komentotarkenne cs on 4, kyseessä on SDO-siirron keskeytys ja keskeytyskoodin voi lukea SDO_data-tietueen data-muuttujasta. Tässä tapauksessa funktio palauttaa arvon NULL. Keskeytyskoodit on listattu taulukossa 8, sivulla 30.

Jos komentotarkenne on jokin muu kuin yllä mainitut 2 tai 4, funktio tulostaa virheilmoituksen ja palauttaa SDO_data-tietueen data-muuttujan arvon.

6.5.4 SDO-kirjoittaminen

Kirjoittaminen toisen solmun objektikirjastoon tapahtuu pitkälti samaan tapaan kuin sieltä lukeminen. Kirjoituspyyntö lähetetään funktiolla CANOPEN_WriteSDO(), joka ottaa parametreina kohdesolmun solmutunnisteen, luettavan objektin indeksin ja alaindeksin, datan pituuden ja itse kirjoitettavan datan. Alussa nämä muuttujat kopioidaan tietueeseen SDO_data ja tietueen updated-lippu nollataan.

Seuraavaksi lähetetään varsinainen kirjoituspyyntö funktiolla CAN_SendFrame(). Kehyksen tunniste haetaan taulukosta SDO_send_request_id[]. Muilta osin seurataan SDO-alustusviestin rakennetta, joka näkyy kuvassa 10. Komentotarkenteeksi cs asetetaan 1.

Tämän jälkeen odotetaan 50 millisekuntia ja ajetaan funktio CANOPEN_Process(), joka muun muassa kopioi mahdollisen vastaanotetun kuittausviestin tiedot tietueeseen SDO_data ja asettaa tietueen update-lipun. Jos vastausta ei saatu, eli update-lipun arvoksi jää 0, tulostetaan virheilmoitus ja palautetaan NULL.

Jos vastausviestin komentotarkenne cs on 3, on kirjoitus onnistunut. Funktio tulostaa ilmoituksen ja palauttaa 0. Jos komentotarkenne on 4, kyseessä on SDO-siirron keskeytysviesti. Keskeytyskoodi tulostetaan virheilmoituksen mukana ja toimitetaan funktion paluuarvona kutsuvalle funktiolle. Jos komentotarkenne ei ole kumpikaan näistä, tulostetaan virheilmoitus ja palautetaan SDO-tietueen data-muuttujan arvo.

6.5.5 CANopen-prosessi

Pääosa CANopen-solmun toiminnoista tapahtuu funktiossa CANOPEN_Process(). Se lukee viestit CAN-vastaanottopuskurista, käsittelee niiden sisällön ja lähettää Heartbe-

at-viestit määrätyin aikavälein. Funktiota on syytä kutsua pääohjelmasta riittävän usein, jottei tapahdu vastaanottopuskurin ylivuotoa ja jotta Heartbeat-viestit tulevat lähetettyä riittävän usein.

Heartbeat

Ensin lähetetään Heartbeat-viesti, jos on kulunut CANOPEN_heartbeat_time-muuttujan määrittelemä aika sitten edellisen Heartbeat-viestin lähettämisen. Tämän jälkeen tarkistetaan, kuinka kauan on viimeisimmästä IVN-yhdyskäytävän lähettämästä Heartbeat-viestistä. Jos aikaa on kulunut yli 1000 millisekuntia, tulostetaan virheilmoitus. Yhdyskäytävän Heartbeat-viestin tarkkaileminen on vapaaehtoista, joten tämä osio koodista voidaan haluttaessa poistaa.

CANopen-viestien vastaanottaminen

CANopen-viestit luetaan ja käsitellään silmukassa, joka suoritetaan niin monta kertaa kuin CAN-vastaanottopuskurissa on viestejä. Silmukan alussa viesti luetaan CAN-vastaanottopuskurista paikalliseen puskuriin buf. Sitten viesti ryhmitellään vastaanotetun kehyksen tunnisteen perusteella.

NMT

Jos vastaanotetun kehyksen tunniste on 0, kyseessä on NMT-viesti, jolla isäntä hallitsee asiakkaan NMT-tiloja. NMT-viestit käsitellään erillisessä funktiossa nimeltä CANOPEN_HandleNMT(), jolle välitetään NMT-viestin komentotarkenne ja kohdesolmun solmutunniste parametrissa data.

SDO-pyyntö

Jos kehyksen tunniste on jokin SDO_receive_request[]-taulukon alkioista, kyseessä on SDO-pyyntö solmulta, jonka solmutunniste on taulukon alkion numero plus yksi. SDO-pyynnön komentotarkenne, datan pituus, indeksi ja alaindeksi puretaan viestistä muuttujiin ja välitetään parametreina funktiolle CANOPEN_HandleOD(), joka käsittelee SDO-pyynnöt ja lähettää lukupyynnöille vastauksen ja kirjoituspyynnölle kuittauksen. Jos kirjoitus- tai lukupyynnön kohdeobjektia tai alaindeksiä ei ole kirjastossa, lähetetään SDO-keskeytysviesti. Keskeytysviesti lähetetään myös jos yritetään kirjoittaa ob-

jektiin, johon kirjoittaminen on kielletty tai jos kirjoituspyynnön datatyyppi ei täsmää. Keskeytysviestien keskeytyskoodit ja niiden selitykset löytyvät taulukosta 8, sivulta 30.

SDO-vastaus

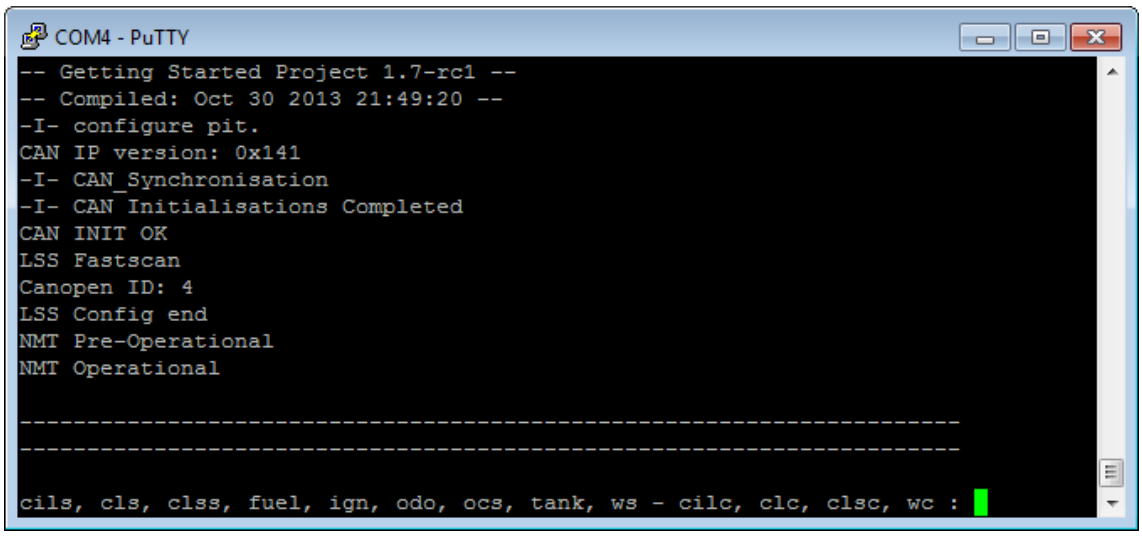
Jos kehyksen tunniste on jokin SDO_receive_response[]-taulukon alkioista, kyseessä on SDO-vastaus aiemmin lähetettyyn SDO-pyyntöön. Vastaus tulee solmulta, jonka solmutunniste on taulukon alkion numero plus yksi. CAN-kehyksen alemmassa data-kentässä olevat bittikentät puretaan SDO_data-tietueen muuttujiin. Nämä bittikentät on määritelty kuvassa 10, sivulla 26. Viestin indeksi- ja alaindeksi-kentät puretaan apu-muuttujiin index ja subindex.

Tämän jälkeen verrataan vastaanotetun viestin index- ja subindex-muuttujien arvoja SDO_data-tietueessa olevien muuttujien arvoihin. Jos nämä täsmäävät, tiedetään, että kyseessä on vastaus aiemmin lähetettyyn pyyntöön, ja data kopioidaan buf-puskurista SDO_data-tietueen data-kenttään. Jos datan pituus on määritelty vastausviestin kentässä n, data maskataan siten, että n-kentän määrittelemän datan pituuden yli menevät bitit nollataan. Kun data on luettu, asetetaan SDO_data-tietueen lippu updated merkiksi siitä, että tietueeseen on vastaanotettu uutta dataa.

6.6 CiA447-testausohjelma

Taksamittariin liitetyn auton CiA447-lisälaiteväylän toimintojen kartoittamiseksi ja kokeilemiseksi tehtiin testausohjelma, jonka ohjelmakoodi on tiedostossa cia447.c. Testausohjelman käyttöliittymä toimii Debug Unit -sarjaportin kautta, kuten luvussa 6.1 on määritelty.

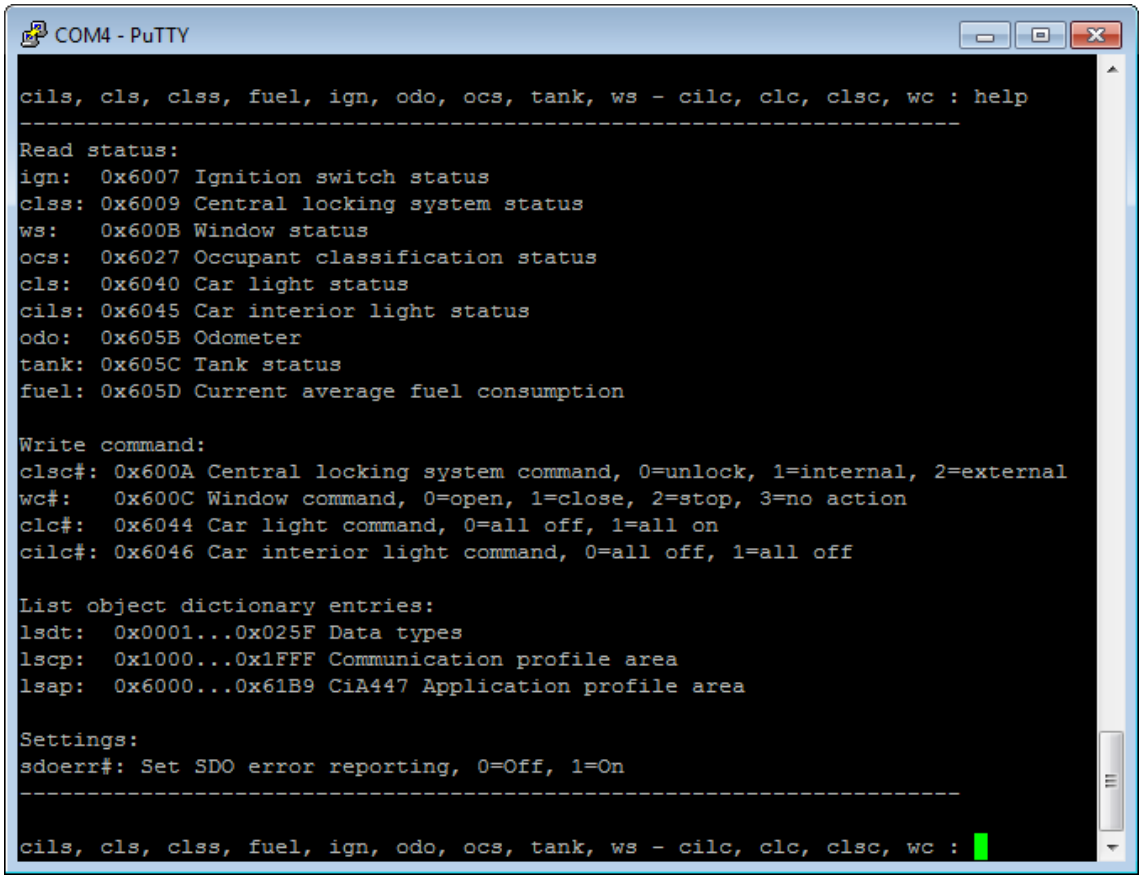
Taksamittarin käynnistyessä se käy läpi useita aiemmissa luvuissa selitettyjä alustustoimenpiteitä ja antaa niistä palautetta. Jos kaikki menee hyvin, viimeisenä tulostuu ”NMT Operational.” Tämä tarkoittaa, että CANopen-solmu on täydessä käyttövalmiudessa ja komentoja voidaan alkaa syöttää. Taksamittarin tulosteet onnistuneen käynnistymisen yhteydessä näkyvät kuvassa 28.



Kuva 28. Taksamittarin alustustoimenpiteet.

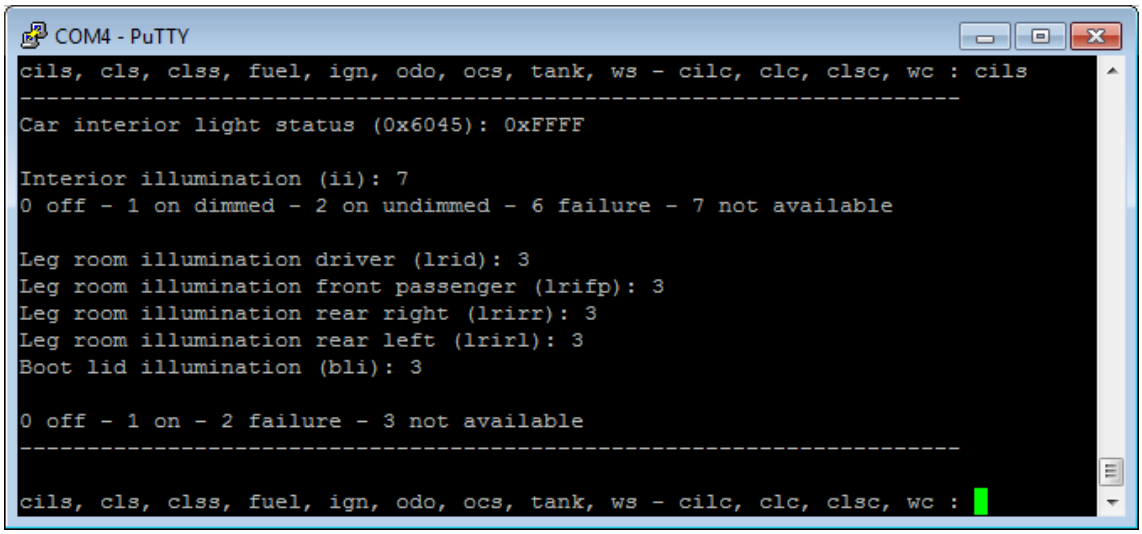
Luku- ja ohjauskomennot

Laitetta ohjataan käskyillä, joista tärkeimmät on listattu komentorivillä, jonka saa näkyviin syöttämällä rivinvaihdon. Käskyjen jälkeen tulee syöttää rivinvaihto. Käskyllä *help* voidaan tulostaa näkyviin apuruutu, joka näkyy kuvassa 29.



Kuva 29. Testausohjelman apuruutu.

Yksittäisiä toimintoja pystyy myös ohjaamaan. Niihin liittyvät komennot on listattu kyseisen objektin tilalistauksessa. Otetaan esimerkiksi auton sisävalot, joiden tilan listaus on kuvassa 30. Sisävalo-objekti on jaettu kuuteen osaan, ja jokaista osaa voidaan ohjata erikseen kirjoittamalla kyseisen toiminnon lyhenne ja haluttu numero peräkkäin komentoriville. Jos siis haluttaisiin sytyttää kuljettajan jalkatilan valo, käsky olisi *lrid1.* Sisävalojen lisäksi testausohjelmalla voidaan erillisohjata sähköikkunoita.



Kuva 30. Testausohjelmassa listattuna simulaattorin sisävalojen tila

Objektikirjaston skannaaminen

Yksittäisten objektien lukemisen ja kirjoittamisen lisäksi testausohjelmalla voidaan listata koko objektikirjaston sisältö. Tämä osio ohjelmasta lähettää SDO-lukupyyntöjä vuoron perään jokaiseen IVN-yhdyskäytävän objektikirjaston merkintään. Jos yhdyskäytävä vastaa jollain muulla kuin SDO-siirron keskeytysviestillä, tiedetään, että objekti on käytössä ja se on tilaobjekti. Jos yhdyskäytävä vastaa SDO-siirron keskeytysviestillä, objekti on ohjausobjekti tai sitä ei ole olemassa. Olemattomien objektien ja ohjausobjektien erottamiseksi ohjelma tulostaa yhdyskäytävän palauttaman keskeytyskoodin. Jos yhdyskäytävä antaa eri keskeytyskoodin olemattomille objekteille ja ohjausobjekteille, pystytään objektikirjaston sisältö kartoittamaan.

Objektit, joilla on useita alaindeksejä, erotetaan yksi-indeksisistä koettamalla lukea objektin alaindeksi 1. Jos se on olemassa, objektilla on enemmän kuin yksi alaindeksi ja dataa sisältävien alaindeksien määrä löytyy jo luetusta alaindeksistä 0. Jokaisesta

objektista yritetään lukea myös alaindeksi 0xFF, joka, jos on tuettu, sisältää objektin objektikoodin. Alaindeksin 0xFF rakenne näkyy Kuva 9, sivulla 22.

Objektikirjasto voidaan listata kolmessa osassa. Komento *lsdt* listaa tietotyypit eli objektit 0x0001…0x025F. Komento *lscp* listaa kommunikaatioprofiilin alueen eli objektit 0x1000…0x1FFF. Komento *lsap* listaa CiA447-sovellusprofiilin alueen eli objektit 0x6000…0x61B9. Ennen listauskomentojen käyttämistä kannattaa SDO-virheiden raportointi ottaa pois päältä komennolla *sdoerr0*.

7 Testaus

Testauksen tarkoituksena on selvittää taksamittarin kanssa samassa väylässä olevan IVN-yhdyskäytävän standardinmukaisuus ja kartoittaa sen tukemat objektitoiminnot. Testaustyökaluna käytettiin taksamittarin testausohjelman lisäksi ilmaisen CAN-väylän monitorointiohjelman PCAN-View versiota 3.2.3.221.

7.1 Testausprosessi

Solmun käynnistys

Testausprosessin ensimmäisessä osiossa tarkkaillaan PCAN-View-ohjelmalla taksamittarin ja testattavan laitteen välistä CAN-viestiliikennettä taksamittarin käynnistyksen yhteydessä ja tallennetaan viestit lokitiedostoon. Tätä viestiliikennettä verrataan CANopen-standardiin ja CiA447-sovellusprofiiliin ja todetaan IVN-yhdyskäytävän standardinmukaisuus käynnistykseen liittyvien protokollien osalta.

Yksittäisten objektien lukeminen

Testausprosessin toisessa osiossa luetaan yksittäisiä objektikirjaston merkintöjä, tulkitaan niiden sisältöä ja verrataan sitä CiA447-standardiin. Nämä objektit on valittu asiakkaan vaatimusten perusteella tai siksi, että niistä kuvitellaan mahdollisesti olevan jotain hyötyä asiakkaalle. Nämä objektit ja niiden kuvaukset on listattu taulukossa 27.

Taulukko 27. Yksittäin testattavat tilaobjektit.

Indeksi	Objektin kuvaus
0x6007	Virta-avaimen tila
0x6009	Keskuslukituksen tila
0x600B	Ikkunoiden tila
0x6027	Matkustajantunnistuksen tila
0x6040	Auton valojen tila
0x6045	Auton sisävalojen tila
0x605B	Matkamittarilukema
0x605C	Polttoainetankin tila
0x605D	Keskimääräinen polttoaineenkulutus

Ohjausobjekteihin kirjoittaminen

Testausprosessin kolmannessa osiossa testataan ohjausobjekteihin kirjoittamista, eli pyritään ohjaamaan simulaattorin tai auton toimintoja. Toimintoja yritetään ohjata kahdella tapaa. Ensin kokeillaan yhden objektin sisältämien toimintojen yhtäaikaista ohjaamista ja sen jälkeen yksittäisten toimintojen ohjaamista objektien sisällä. Taulukossa 28 on listattu testausohjelman tukemat ohjausobjektit. Näistä keskuslukitus-objekti sisältää ainoastaan yhden toiminnon ja ikkunoiden ohjaus sekä auton sisävalojen ohjaus tukevat yksittäisten toimintojen ohjaamista.

Taulukko 28. Testausohjelman tukemat ohjausobjektit.

Indeksi	Objektin kuvaus
0x600A	Keskuslukituksen ohjaus
0x600C	Ikkunoiden ohjaus
0x6044	Auton ulkovalojen ohjaus
0x6046	Auton sisävalojen ohjaus

Objektikirjaston skannaaminen

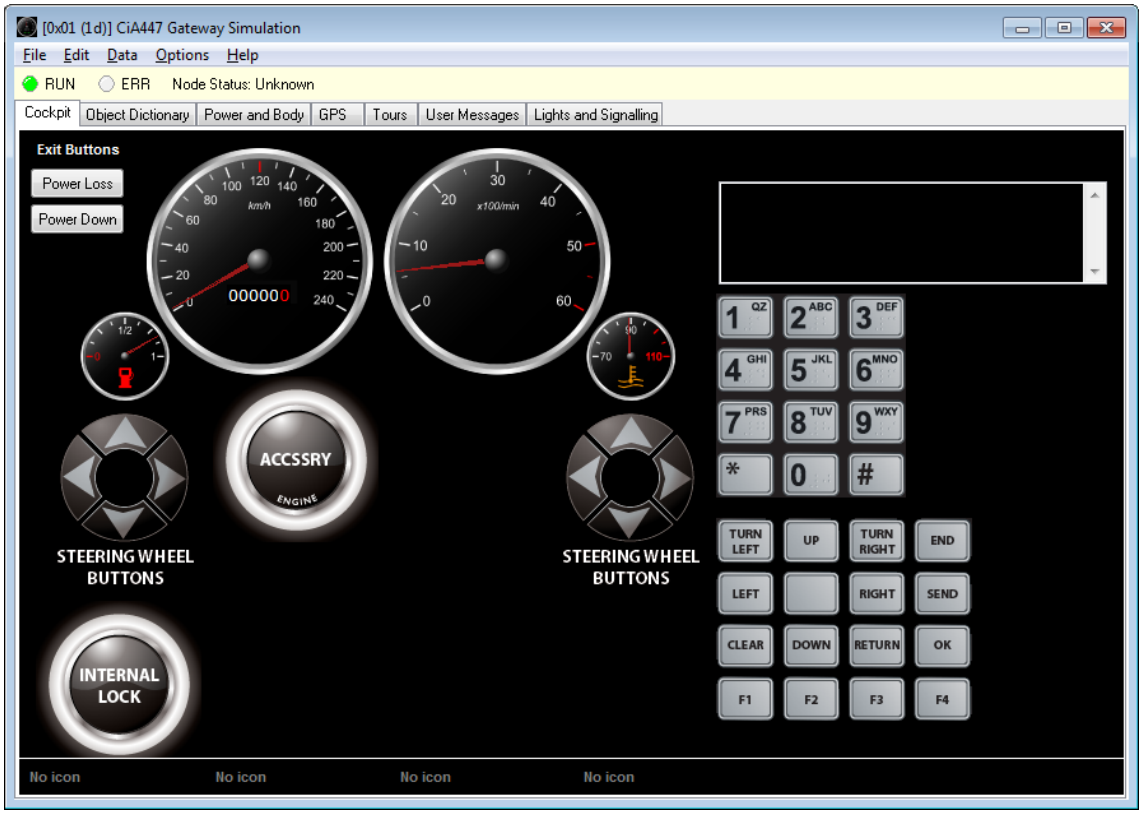
Testausprosessin viimeinen osio on objektikirjaston skannaaminen. Tällä pyritään kartoittamaan yhdyskäytävän tietotyyppi-, määrittely-, tila- ja ohjausobjektit. Ensin tietokoneen terminaaliohjelma asetetaan tekemään loki tiedostoon ja sen jälkeen laitetaan testausohjelma lukemaan yhdyskäytävän objektikirjaston merkinnät tietotyyppien alueelta 0x0001…0x025F, kommunikaatioprofiilin alueelta 0x1000…0x1FFF ja CiA447-sovellusprofiilin alueelta 0x6000…0x61B9.

Tärkeimmät tästä testistä saatavat tulokset ovat lista yhdyskäytävän tukemista tila- ja ohjausobjekteista CiA447-sovellusprofiilin alueella ja PDO-viestien kommunikaatio-

objekteista ja kuvausobjekteista kommunikaatioprofiilin alueella. Jälkimmäiset määrittelevät yhdyskäytävän lähettämät tapahtuma- tai aikaliipaistut PDO-viestit ja niiden sisältämän datan.

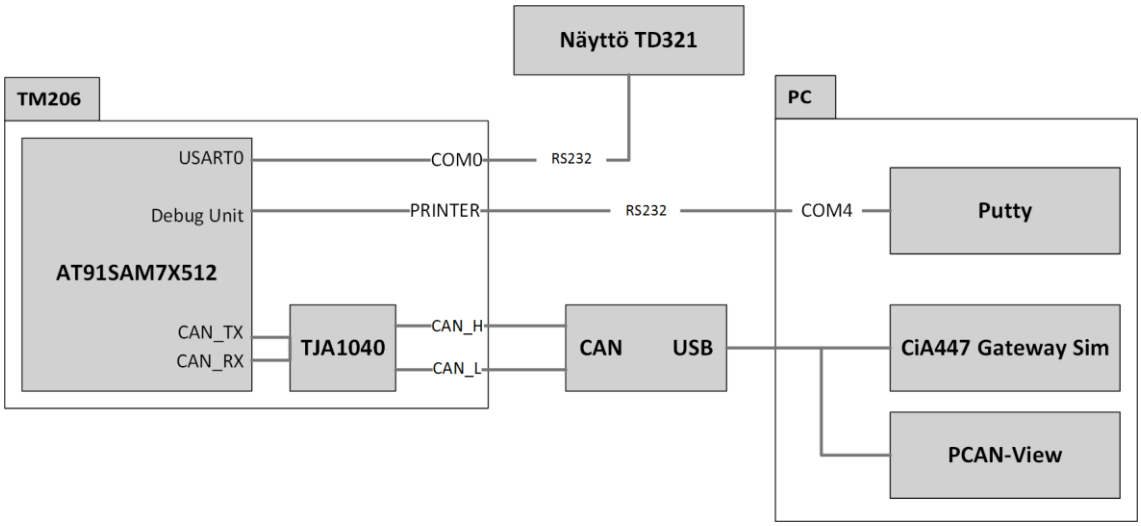
7.2 CiA447 Gateway Simulation

Ensimmäinen testauskohde oli Embedded Systems Academyn CiA447 Gateway Simulation -ohjelman versio 1.63. Se on Windows-ohjelma, joka simuloi auton IVN-yhdyskäytävää. Simulaatio sisältää kojelaudan mittareineen ja nappeineen sekä muita luettavia ja ohjattavia toimintoja. Ohjelma osaa myös simuloida ajamista, mikä on hyödyllinen ominaisuus testattaessa esimerkiksi moottorin tietoja, polttoaineenkulutuslukemia tai renkaan kierroslukuja ja muita vastaavia ominaisuuksia käyttäviä ohjelmia. Kuvassa 31 on CiA447 Gateway Simulation -ohjelman kojelauta. Kojelaudan yläpuolella olevia välilehtiä painamalla pääsee ohjelman muihin osioihin.



Kuva 31. CiA447 Gateway Simulation -ohjelman kojelauta.

CAN-väylänä toimi tässä tapauksessa PCAN-View ja Gateway Simulation -ohjelmien muodostama virtuaalinen CAN-väylä, johon taksamittari oli kytketty USB-liitäntäisellä CAN-sovittimella. Kuvassa 32 on kaavio kytkennöistä.



Kuva 32. Laitteiston kytkennät testattaessa CiA447 Gateway Simulator -ohjelmaa

Solmun käynnistystesti

Solmun käynnistystestiä tehtäessä huomattiin, että CiA447 Gateway Simulation ei toimi käynnistyviä solmuja etsiessään täysin standardin mukaisesti. Ennen LSS Fastscan -menetelmän aloittamista IVN-yhdyskäytävän tulisi etsiä konfiguroimattomia solmuja lähettämällä tasaisin aikavälein LSS-viesti, jonka tunniste on 0x7E4 ja komentotarkenne 0x4C. Gateway Simulation kuitenkin hyppää tämän askeleen yli ja etsii solmuja suoraan lähettämällä sekunnin välein LSS Fastscan aloitusviestiä. LSS Fastscan -menetelmästä eteenpäin solmun konfigurointi simulaatio-ohjelman yhdyskäytävän toimesta toimii standardin mukaisesti.

Kun Gateway Simulationin yhdyskäytävä on siirtänyt solmun NMT-toimintatilaan, se kirjoittaa solmun objektiin 0x1017 arvon 200, joka asettaa Heartbeat-viestin lähetysvälin 200 millisekuntiin. Tämän jälkeen yhdyskäytävä lukee solmun laitetyypin objektista 0x1000 ja solmun tukemat virtuaalilaitteet objektista 0x6000.

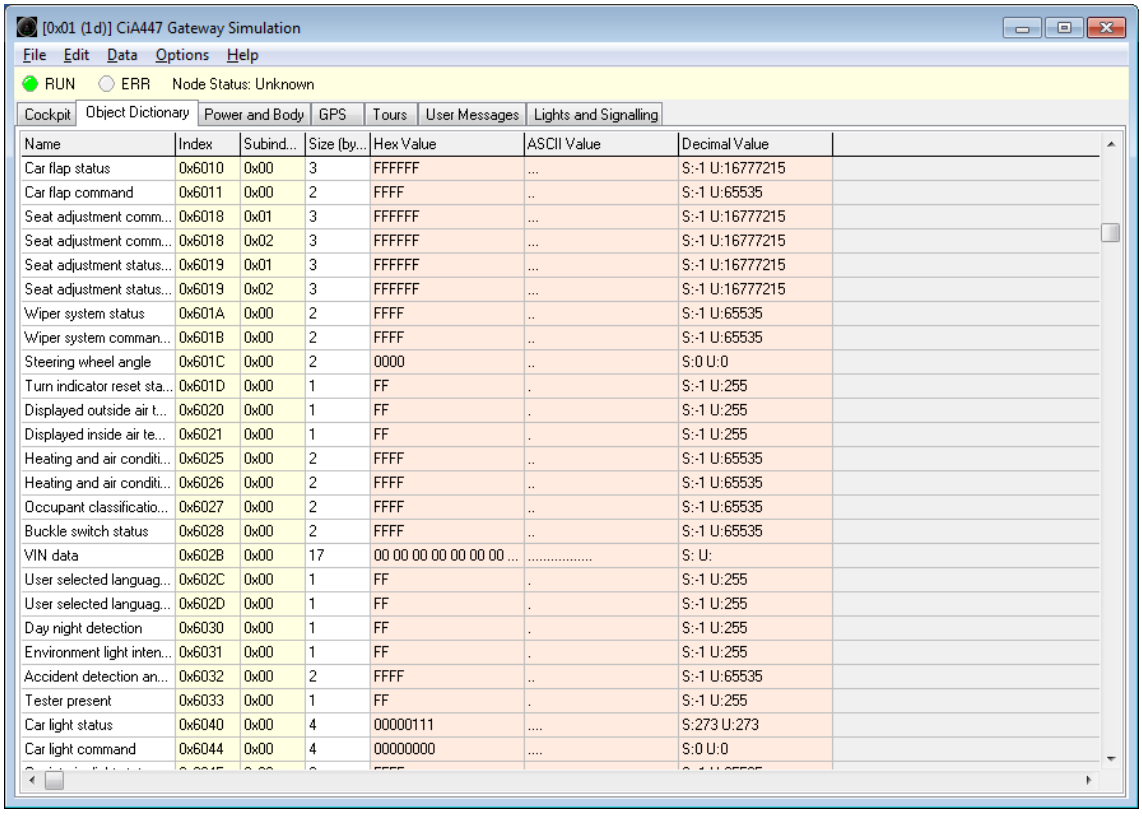
Yksittäisten objektien lukeminen

Yksittäisten objektien lukeminen Gateway Simulationin yhdyskäytävästä onnistui ongelmitta. Luetut arvot vastasivat ohjelman reaaliaikaisessa objektikirjastonäkymässä näkyviä arvoja.

Ohjausobjekteihin kirjoittaminen

Yritettäessä kirjoittaa Gateway Simulation -ohjelman ohjausobjekteihin huomattiin, että ohjelman yhdyskäytävä estää ohjausobjekteihin kirjoittamisen, jos kirjoittava solmu ei ole saanut siltä solmutunnistetta tai ei lähetä Heartbeat-viestejä yhdyskäytävän määrittelemän ajan välein.

Kirjoitettaessa ohjausobjekteihin havaittiin puute Gateway Simulation -ohjelmassa. Ohjausobjekteihin kirjoittaminen ei varsinaisesti ohjaa mitään, eli ohjausobjekteja ei ole kytketty varsinaisiin toimintoihin. Esimerkki tästä näkyy kuvan 33 alareunassa.



Kuva 33. CiA447 Gateway Simulation -ohjelman reaaliaikainen objektikirjastonäkymä.

Kuva on otettu CiA 447 Gateway Simulation -ohjelman objektikirjasto-välilehdestä, joka näyttää reaaliaikaisesti yhdyskäytävän objektikirjaston sisällön. Auton valojen ohjausobjektiin 0x6044 on kirjoitettu 0x00000000, eli kaikki valot on yritetty sammuttaa, mutta sisävalojen tilaobjektin 0x6045 arvo on edelleen 0x00000111. CiA447 Gateway Simulation -ohjelman valmistajan mukaan asioiden ohjaamista ei simuloida ohjelman tässä versiossa, mutta se on tulossa seuraavaan julkaisuversioon.

Objektikirjaston skannaaminen

CiA447 Gateway Simulation -ohjelman yhdyskäytävän testauksen viimeisessä osiossa yhdyskäytävän objektikirjasto skannattiin testausohjelmalla.

Objektin 0x1000 sisältö on 0x000101BF, mikä tarkoittaa, että simulaattorin yhdyskäytävä tukee CiA447-profiilin versiota 2.0. Objektikirjastossa ei määritellä tietotyyppejä eikä yhdyskäytävä tue objektien alaindeksiä 0xFF, joka määrittelee objektin objektikoodin ja tietotyypin.

Yhdyskäytävää ei ole asetettu vastaanottamaan PDO-viestejä, mutta lähetettäviä PDO-viestejä on määritelty 15. Viestit ovat aika- tai tapahtumaliipaistuja, ja ne sisältävät tietoa useista simulaattorin toiminnoista. Pikaisen kokeen perusteella nämä PDO-viestit vastaavat sisällöltään ja kommunikaatioparametreiltaan objektikirjaston määrittelyjä.

Objektikirjaston skannauksen lopuksi kartoitettiin CiA447-sovellusprofiilin alueen tila- ja ohjausobjektit. Esimrkkikoodissa 1 on ote testin tuottamasta lokitiedostosta CiA447-sovellusprofiilin alueelta.

```
0x6017 not supported or write only, 0x8000000
0x6017 sub256 not supported

0x6018 sub0: 0x2
0x6018 sub256 not supported

…

0x601B not supported or write only, 0x6010000
0x601B sub256 not supported
```

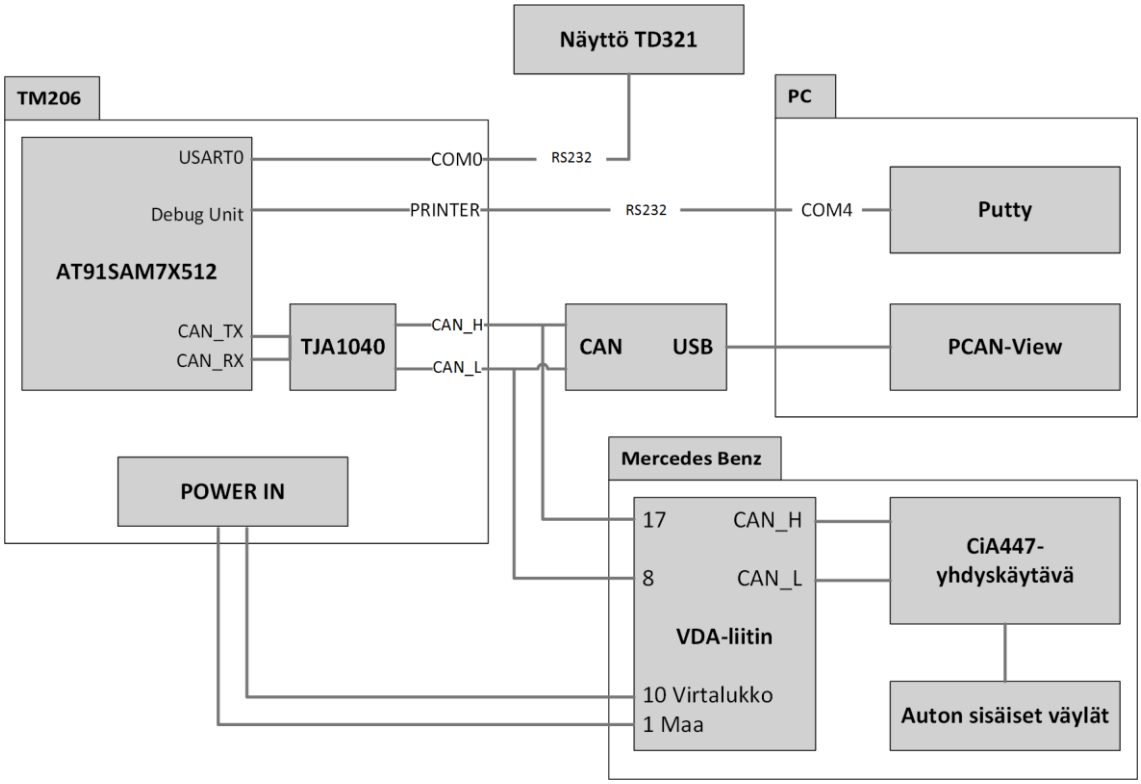
Esimerkkikoodi 1. Ote simulaattorin yhdyskäytävän objektikirjaston skannauslokista.

Esimerkistä nähdään, että lukuyritys objektiin 0x6018 palauttaa arvon 2. Se on siis tilaobjekti. Lukuyritys objektiin 0x6017 palauttaa keskeytyskoodin 8000 0000, joka tarkoit-

taa yleistä virhettä. Lukuyritys objektiin 0x601B taas palauttaa keskeytyskoodin 0601 0000, joka tarkoittaa, että objektitoiminto ei ole tuettu. Tästä voidaan päätellä, että objektia 0x6017 ei ole olemassa ja objekti 0x601B on ohjausobjekti, josta lukeminen ei ole sallittu. Näin Gateway Simulation -ohjelman yhdyskäytävän kaikki objektit voitiin kartoittaa.

7.3 Mercedes Benz E

Taksamittarin testausohjelmaa päästiin kokeilemaan myös uudehkossa taksivarustellussa E-sarjan Mercedes Benzissä. Taksamittari kytkettiin omatekoisella testauskaapelilla auton VDA-liittimeen, joka löytyy vaihdekepin oikealta etupuolelta suojamuovin alta. Laitteiston kytkennät näkyvät kuvassa 34.



Kuva 34. Laitteiston kytkennät testattaessa Mercedes Benzin IVN-yhdyskäytävää.

Taksamittarin lisäksi auton lisälaiteväylään kytkettiin tietokoneeseen liitetty USB-CAN-adapteri, jotta voitiin tarkkailla väylän liikennettä PCAN-View-ohjelmalla. Lisäksi tietokoneen sarjaportti kytkettiin taksamittarin PRINTER-porttiin käyttöliittymää varten.

Solmun käynnistystesti

Solmun käynnistäminen sujuu Mercedes Benzissä täysin standardin mukaan, eli sen yhdyskäytävä lähettää, Gateway Simulation -ohjelmasta poiketen, kyselyviestin ennen LSS Fastscan -menetelmän aloittamista. Kun solmu on konfiguroitu ja käynnistetty, yhdyskäytävä tekee vielä muutaman luku- ja kirjoitustoimenpiteen solmun objektikirjastoon. Näihin SDO-pyyntöihin tulee vastata, tai yhdyskäytävä keskeyttää kaikki solmun SDO-pyynnöt.

Ensin yhdyskäytävä lukee solmun laitetyypin objektista 0x1000, sitten solmun tukemat virtuaalilaitteet objektista 0x6000. Seuraavaksi yhdyskäytävä lukee toimittajanumeron, tuotenumeron, revisionumeron ja sarjanumeron objektin 0x1018 alaindekseistä. Lopuksi yhdyskäytävä lukee solmun Heartbeat-viestien aikavälin objektista 0x1017 ja kirjoittaa samaan objektiin arvon 200. Tämän jälkeen solmu voi ottaa osaa väylän viestiliikenteeseen.

Yksittäisten objektien lukeminen

Tässä testausvaiheessa luettiin Mercedes Benzin IVN-yhdyskäytävästä kaikki testausohjelmassa määritellyt tilaobjektit, purettiin saadut arvot CiA447-profiilin määrittelemiin eri toimintojen bittikenttiin ja verrattiin tilatietoja toimintojen todelliseen tilaan. Näin pystyttiin päättelemään, mitä toimintoja yhdyskäytävä tukee ja vastaavatko sen palauttamat arvot todellisuutta.

Virta-avaimen ollessa ”ignition on” -asennossa saatiin *virta-avaimen tilaobjektin* lukupyyntöön vastaukseksi arvo 0x15. Kuvassa 35 tämä arvo on purettu objektin CiA447-profiilin määrittelemiin kenttiin. Profiilin mukaan bittikentän arvo 0 tarkoittaa pois päältä ja arvo 1 päällä. Arvo 2 tarkoittaa virhettä ja arvo 3, että signaali ei ole käytössä. Kuvasta nähdään, että objektin arvo vastaa virta-avaimen todellista tilaa.

7 6	5 4	3 2	1 0
ignition start	ignition on	accessory	ignition lock
0	1	1	1

Kuva 35. Virta-avaimen tilaobjekti 0x6007 [18, s. 11].

Keskuslukituksen ollessa auki, *keskuslukituksen tilaobjektista* 0x6009 luettiin arvo 0xF0, joka on kuvassa 36 purettu CiA447-profiilin määrittelemiin kenttiin. Keskuslukituksen tilan arvo 0 tarkoittaa sovellusprofiilin mukaan, että keskuslukitus on auki. Objektin arvo siis vastaa keskuslukituksen todellista tilaa.

7 4	3 0
varattu, aina 0xF	keskuslukituksen tila
0xF	0

Kuva 36. Keskuslukituksen tilaobjekti 0x6009 ja ohjausobjekti 0x600A [18, s. 14–15].

Kaikkien ikkunoiden ollessa kiinni *sähköikkunoiden tilaobjektista* 0x600B luettiin arvo 0xFFFF00, joka on purettu sovellusprofiilin määrittelemiin bittikenttiin kuvassa 37. Profiilin mukaan bittikentän arvo 0 tarkoittaa, että ikkuna on auki ja arvo 1, että ikkuna on kiinni. Arvo 2 tarkoittaa, että ikkuna on avautumassa ja arvo 3, että ikkuna on sulkeutumassa. Arvot 4 ja 5 eivät ole käytössä. Arvo 6 tarkoittaa virhettä ja arvo 7, että signaali ei ole käytössä. Objektista luettu arvo ei siis ole sovellusprofiilin mukainen, eikä vastaa sähköikkunoiden tilaa.

23 18	17 15	14 12	11 9	8 6	5 3	2 0
varattu, aina 0x3F	w3rl	w3rr	wrl	wrr	wfp	wd
0x3F	7	7	7	4	0	0

Kuva 37. Sähköikkunoiden tilaobjekti 0x600B [18, s. 15].

Matkustajantunnistuksen tilaobjekti 0x6027 Mercedes Benzin IVN-yhdyskäytävässä ei tue kuin etumatkustajan tunnistusta. Tämän tilan voi lukea objektin biteistä 2…3. Kaikkien muiden objektin kenttien arvo on aina 3, mikä tarkoittaa, että nämä signaalit eivät ole käytössä.

Auton ulkovalojen tilaobjekti luettiin useaan otteeseen eri valojen ollessa päällä. Tämän perusteella koostettiin lista valoista, joiden tilan pystyy lukemaan objektista 0x6040. Nämä valot, ja niitä vastaavat bittikentät objektissa on listattu taulukossa 29.

Taulukko 29. Auton valojen tilaobjektissa 0x6040 tuetut toiminnot.

Bitit	**Toiminto**
2…3	Jarruvalo
6…7	Pitkät ajovalot
8…9	Lyhyet ajovalot
10…11	Etusumuvalot
12…13	Takasumuvalot
14…15	Vilkku oikealle
16…17	Vilkku vasemmalle
22…23	Hätävilkut
26…27	Seisontavalot
28…29	Pysäköimisvalot oikea
30…31	Pysäköimisvalot vasen

Sisävalojen tilaobjekti 0x6046 sisältää yleisen sisävalon tilan lisäksi erilliset kentät jalkatilojen ja takakontin valoille. Objektin rakenne näkyy kuvassa 38. Objektin sisältö

Mercedes Benzin yhdyskäytävässä on aina 0xFFF8. Tämä arvo on purettu kuvassa objektin bittikenttiin. Bitit 13…15 eivät ole käytössä ja niiden arvo tulisi aina olla 1. Muissa bittikentissä arvo 3 tarkoittaa, että kenttä ei ole käytössä. Ainoastaan yleisen sisävalon tila, eli kenttä ii, on siis käytössä. Sekään ei tosin muutu kun auton sisävaloja kytkee päälle ja pois.

15	14	13	12 11	10 9	8 7	6 5	4 3	2 0
r	r	r	bli	lrirl	lrirr	lrifp	lrid	ii
1	1	1	3	3	3	3	3	0

Kuva 38. Sisävalojen tila- ja ohjausobjektin rakenne ja tilaobjektista luetut arvot [18, s.44].

Matkamittarilukema luettiin objektista 0x605B ja se vastasi Mercedes Benzin aitoa matkamittarilukemaa. Objekti on UNSIGNED24-tyyppinen muuttuja, ja sen mittayksikkö on 100 metriä [18, s. 63].

Mercedes Benzin yhdyskäytävä ei tue *polttoainetankin tilaobjektia* 0x605C, vaan keskeyttää SDO-lukupyynnön koodilla 0602 0000, joka tarkoittaa, että objektia ei ole objektikirjastossa.

Objekti 0x605D, jonka pitäisi sisältää keskimääräisen *polttoaineenkulutuksen* millilitroina sekunnissa, ei toimi Mercedes Benzin yhdyskäytävässä. Objektin sisältö on tilanteesta riippumatta aina 0xFFFFF.

Ohjausobjekteihin kirjoittaminen

Ohjausobjekteihin kirjoittamista testattiin Mercedes Benzissä keskuslukituksen, sähköikkunoiden, ulkovalojen ja sisävalojen ohjausobjekteilla.

Keskuslukituksen ohjausobjekti on kahdeksan bittiä pitkä, ja sen neljä alinta bittiä muodostavat ohjausbittikentän ja ylempi neljän bitin kenttä on sovellusprofiilissa määritelty arvoon 0xF. Objektin rakenne näkyy kuvassa 36.

Yritettäessä kirjoittaa objektiin sovellusprofiilin mukaisia arvoja 0xF0, 0xF1 tai 0xF2, yhdyskäytävä vastasi aina SDO-siirron keskeytysviestillä. Keskeytyskoodi oli 0601 0000, joka tarkoittaa ei-tuettua objektitoimintoa.

Koska sähköikkunoiden tilaobjekti ei ole sovellusprofiilin mukainen, voidaan olettaa, ettei *sähköikkunoiden ohjausobjektikaan* ole. Ikkunoita kuitenkin yritettiin ohjata profiilin mukaisilla viesteillä, kuten 0xF000, jonka pitäisi sulkea kaikki ikkunat, tai 0xF555, jonka pitäisi avata kaikki ikkunat. Yhdyskäytävä palautti kuitenkin aina keskeytysviestin koodilla 0601 0000.

Ulkovalojen ohjausobjekti 0x6044 oli ainoa testattu Mercedes Benzin yhdyskäytävän ohjausobjekti, joka otti käskyjä vastaan. Valoja yritettiin ohjata viestillä 0x55555555, jonka pitäisi sytyttää kaikki valot ja viestillä 0x00000000, jonka pitäisi sammuttaa kaikki valot. Mutta vaikka objektiin kirjoittaminen ei päättynytkään SDO-siirron keskeytykseen, ei se varsinaisesti ohjannut mitään valoja.

Kuva 38 esittää *sisävalojen ohjausobjektin* rakenteen. Kenttä bli on takakontin valon ohjauskenttä ja lri-alkuiset kentät jalkatilojen valojen ohjauskenttiä. Kenttä ii ohjaa yleistä sisävaloa. Bitit 13, 14, 15 eivät sovellusprofiilin mukaan ole käytössä ja niiden arvojen tulisi aina olla 1.

Sisävaloja yritettiin ohjata viesteillä 0xEAAA ja 0xE000. Ensimmäisen pitäisi sytyttää kaikki valot ja jälkimmäisen sammuttaa ne. Yhdyskäytävä keskeytti kummankin viestin keskeytyskoodilla 0601 0000. Valoja yritettiin myös ohjata erikseen, lukemalla ensin sisävalojen tilaobjektin arvo, muuttamalla yhden bittikentän arvoa ja kirjoittamalla tulos ohjausobjektiin. Lopputulos oli sama keskeytysviesti kuin aiemmin.

Objektikirjaston skannaaminen

Mercedes Benzin IVN-yhdyskäytävän testauksen viimeisessä osiossa yhdyskäytävän objektikirjasto skannattiin testausohjelmalla.

Objektin 0x1000 arvo on 0x1BF, mikä tarkoittaa, että Mercedes Benzin IVN-yhdyskäytävä tukee CiA447-profiilin versiota 1.0. Tämä saattaa selittää, miksi yhdyskäytävä ei vaikuta toimivan profiilin mukaisesti, sillä testausohjelmisto ja tämä dokumentti on toteutettu version 2.0 pohjalta.

Yhdyskäytävä ei tue objektien alaindeksiä 0xFF, joka määrittelee objektin tietotyypin ja objektikoodin.

PDO-viestit

Yhdyskäytävää ei ole asetettu vastaanottamaan PDO-viestejä, mutta lähetettäviä PDO-viestejä on määritelty 14 kappaletta. Tästä joukosta löydettiin kaksi viestiä joiden sisältö saattaa olla asiakkaalle hyödyllinen.

Objekti 0x1810 määrittelee 100 millisekunnin välein lähetettävän PDO-viestin, jonka tunniste on 0x181. Viestin sisältö määritellään objektissa 0x1A10. Viestin bitit 0…15 sisältävät moottorin kierrosluvun objektista 0x6050, bitit 16…31 renkaiden kierrosluvun objektista 0x6053 ja bitit 32…47 renkaan pulssilaskurin arvon objektista 0x6055. Tällä hetkellä taksamittari laskee kuljetun matkan ja auton nopeuden taksamittarille erikseen johdotetun pulssianturin avulla. Käyttämällä hyväksi tämän PDO-viestin sisältöä päästään uuden taksin varustelussa vähemmällä asennustyöllä ja johdotuksella.

Objekti 0x1845 määrittelee tapahtumaliipaistun PDO-viestin, jonka estoaika on 10 millisekuntia. Viesti siis lähetetään, kun viestiin kuvattujen objektien arvot muuttuvat jos edellisestä viestistä on kulunut vähintään 10 millisekuntia. Viestiin kuvatut objektit määritellään objektissa 0x1A45. Viesti sisältää GPS-koordinaatit siten, että bitit 0…31 sisältävät leveyskoordinaatin ja bitit 32…63 pituuskoordinaatin.

Objektikirjaston skannauksen lopuksi kartoitettiin CiA447-sovellusprofiilin alueen tila- ja ohjausobjektit. Esimerkkikoodissa 2 on ote testin tuottamasta lokitiedostosta CiA447-sovellusprofiilin alueelta.

```
0x6017 not supported or write only, 0x6020000
0x6017 sub256 not supported

0x6018 sub0: 0x2
0x6018 sub256 not supported

…

0x601B not supported or write only, 0x6010000
0x601B sub256 not supported
```

Esimerkkikoodi 2. Ote Mercedes Benzin yhdyskäytävän objektikirjaston skannauslokista.

Esimerkistä nähdään, että lukuyritys objektiin 0x6018 palauttaa arvon 2. Se on siis tila-objekti. Lukuyritys objektiin 0x6017 palauttaa keskeytyskoodin 0602 0000, joka tarkoittaa, ettei objektia ole objektikirjastossa. Lukuyritys objektiin 0x601B taas palauttaa keskeytyskoodin 0601 0000, joka tarkoittaa, että objektitoiminto ei ole tuettu. Tästä voi-

daan päätellä, että objektia 0x6017 ei ole olemassa ja objekti 0x601B on ohjausobjekti, josta lukeminen ei ole sallittu. Näin Mercedes Benzin yhdyskäytävän kaikki objektit voitiin kartoittaa

Asiakkaalle mahdollisesti hyödyllisiä tilaobjekteja ovat renkaan kierrosluku indeksissä 0x6053, renkaan pulssilaskuri indeksissä 0x6055, renkaan kierroslaskurin ylivuoto indeksissä 0x6056, pulsseja per renkaan kierros indeksissä 0x6057, matkamittarilukema indeksissä 0x605B ja GPS-järjestelmän sijainti- nopeus-, korkeus-, suunta- ja tilaobjektit indekseissä 0x60B0…60B6.

8 Yhteenveto

Insinöörityössä suunniteltiin Semelin taksamittarille CAN-ajuri, CANopen-pino sekä CiA447-testiohjelma. CAN-ajuria ja CANopen-pinoa voidaan tulevaisuudessa käyttää CAN- ja CANopen-ominaisuuksien lisäämiseksi taksamittarin tuotantoversioon.

CiA447-testiohjelmalla testattiin sekä Embedded Systems Academyn CiA447 Gateway Simulation -ohjelman että E-sarjan Mercedes Benzin IVN-yhdyskäytävän standardinmukaisuus.

Simulaatio-ohjelman testauksessa huomattiin, että ohjelman versio 1.63 on vielä vähän keskeneräinen, eikä se tue toimintojen ohjaamista ohjausobjekteilla. Lisäksi ohjelman yhdyskäytävä jättää solmujen konfiguroinnissa yhden askeleen väliin, eikä lähetä sovellusprofiilin mukaista kyselyviestiä ennen LSS Fastscan.-menetelmän aloittamista, vaan etsii konfiguroimattomia solmuja suoraan LSS Fastscan -aloitusviestillä. Tämä ei varsinaisesti estä toimintaa, mutta on otettava huomioon solmua toteutettaessa.

Mercedes Benzin yhdyskäytävä ei myöskään ole täysin CiA447-sovellusprofiilin mukainen. Yhdyskäytävä toimi käynnistystestissä täysin CANopen-standardin ja CiA447-profiilin mukaisesti, mutta objektikirjaston objektien rakenne ei vastaa sovellusprofiilin määritelmiä. Esimerkiksi sisävalojen tilaobjektin sisältö ei vastaa sovellusprofiilin määritelmää. Toimintojen ohjaaminen ohjausobjekteihin kirjoittamalla ei myöskään toiminut.

Näihin ongelmiin saattaa olla syynä, että testatun auton yhdyskäytävä tukee CiA447-sovellusprofiilin versiota 1.0, kun testausohjelma suunniteltiin version 2.0 pohjalta. Ainoa testattu ohjausobjekti, johon kirjoittaminen ei palauttanut SDO-siirron keskeytys-

viestiä, oli auton ulkovalojen ohjausobjekti, mutta siihenkään kirjoittaminen ei varsinaisesti ohjannut mitään valoja. Tämä objekti on testatuista objekteista ainoa, joka ei sisällä varattuja bittejä, vaan objektin koko sisältö on käytössä. Muissa objekteissa on kolmesta kuuteen varattua bittiä, joiden arvon tulisi sovellusprofiilin mukaan olla aina 1. Saattaa olla, että sovellusprofiilin aiemmassa versiossa näiden bittien arvot on määritelty nollaksi ja yhdyskäytävä lähettää keskeytysviestin kun niihin yritetään kirjoittaa ykkösiä. Saattaa myös olla, että objektit ovat sovellusprofiilin versiossa 1.0 eri järjestyksessä tai eri tietotyyppiä.

Asiakkaan vaatimuksista pystyttiin toteuttamaan matkamittarilukeman lukeminen. Lisäksi löydettiin joitain muita mahdollisesti hyödyllisiä tilaobjekteja ja kaksi mahdollisesti hyödyllistä yhdyskäytävän lähettämää PDO-viestiä. Sisävalojen ohjaaminen ei onnistunut. Mercedes Benzin ohjekirjan mukaan autoon on saatavissa peilitaksamittari, joka sytyttää sisävalot automaattisesti siirtyessään maksutilaan. Oikea sisävalojen ohjausviesti voidaan todennäköisesti löytää tarkkailemalla tällaisen taksamittarin lähettämiä viestejä PCAN-View-ohjelmalla tai muulla CAN-monitorointi-ohjelmalla.

Lähteet

1 CAN History, Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=systemdesign-can-history>. Luettu 13.11.2013.

2 Pfeiffer, Olaf; Ayre, Andrew; Keydel, Christian. 2008. Embedded Networking with CAN and CANopen. Greenfield, MA, USA: Copperhill Technologies Corporation.

3 Saha, Heikki. 2005. CAN-väylä. FLUID Finland 4, s. 6–12.

4 CAN physical layer, Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=systemdesign-can-physicallayer>. Luettu 13.11.2013.

5 CAN Specification 2.0 part A.

6 Watterson, Conal. CAN Implementation Guide. Analog Devices application note AN-1123.

7 CAN Specification 2.0 part B.

8 CiA301 CANopen application layer and communication profile version 4.2.0. 2011. Nuremberg, Saksa: CAN in Automation.

9 CiA306 Electronic data sheet specification version 1.3.0. 2005. Nuremberg, Saksa: CAN in Automation

10 Pfeiffer, Olaf. 2012. Features of CiA 447 Application profile for special-purpose car add-on devices. International Conference on Communications 2012, s. 4-11–4-15.

11 German police and CiA447. 2010. Verkkodokumentti. CAN in Automation. <http://www.can-cia.org/index.php?id=731&tx_ttnews[pS]=1310830271&tx_ttnews[pointer]=5&tx_ttnews[tt_news]=886&tx_ttnews[backPid]=726&cHash=3a3d4a164a>. Luettu 13.11.2013.

12 E-sarja Lisäkäyttöohje ja asennusohje takseille ja tekniikkapaketilla varustetuille autoille. 2012. Stuttgart, Saksa: Daimler AG.

13 Spying on fuel consumption. 2013. Verkkodokumentti. CAN in Automation. <http://can-newsletter.org/engineering/applications/nr_spying-fuel-consumption_fuelspy_130507/>. Luettu 13.11.2013.

14 CiA447 Application profile for special-purpose car add-on devices Part 1: General definitions version 2.0.0. 2012. Nuremberg, Saksa: CAN in Automation.

15 CiA305 Layer setting services (LSS) and protocols version 3.0.0. 2013. Nuremberg, Saksa: CAN in Automation.

16 CiA447 Application profile for special-purpose car add-on devices Part 4: Predefined CAN-IDs and communication objects version 2.0.0. 2012. Nuremberg, Saksa: CAN in Automation.

17 AT91SAM ARM-based Flash MCU 6120J-ATARM-05-Mar-12. 2012. Datalehti. Atmel Corporation.

18 CiA447 Application profile for special-purpose car add-on devices Part 3: Detailed process data specification version 2.0.0. 2012. Nuremberg, Saksa: CAN in Automation.