Niklas Peltoniemi

Kestotestien kehittäminen toiminnallisen turvallisuuden laitteille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikan koulutusohjelma

Insinöörityö

12.5.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Niklas Peltoniemi Kestotestien kehittäminen toiminnallisen turvallisuuden laitteille 51 sivua + 11 liitettä 12.5.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Kappaletavara-automaatio
Ohjaaja(t)	Lehtori Antti Liljaniemi Design Manager, DI, Anne Kostiainen

Insinöörityön tavoitteena oli rakentaa ABB Oy:lle kestotestijärjestelmä FSO-turvamoduulin testausta varten. Moduulin toukokuussa 2014 julkaistava versio tuo uutena ominaisuutena mahdollisuuden käyttää PROFIsafe-turvaväylää, mikä erityisesti tarvitsee testausta. Kestotesti rakennettiin neljälle ACS880-taajuusmuuttajalle, turvalogiikkaohjelmointia hyväksikäyttäen.

Työn kannalta suurin haaste oli logiikkaohjelmointi, joka oli myös työn suurin osuus. Ohjelmoinnin opiskelu ja ongelmat toimivat hyvänä harjoituksena, vaikka aikataulu venyi ja haasteet näyttivät välillä ylitsepääsemättömiltä. Asennustyöt onnistuivat hyvin ammattikorkeakoulussa opittujen asioiden pohjalta, eikä tässä osuudessa ollut suurempia ongelmia. Kestotestin kehityksen aikana pidetyt projektipalaverit toimivat erittäin hyvänä harjoituksena projektityöskentelystä.

Teoriaosuudessa käydään läpi tärkeimpiä toiminnallisen turvallisuuden käsitteitä. Nykyyhteiskunnassamme turvallisuus nostetaan yhä useammin esille ja standardit sekä direktiivit määräävät jo laajasti miten teollisuusprosessit pitää suojata. Tuotekehityksen kannalta testaaminen on hyvin tärkeää tuotteen luotettavuuden ja turvallisuuden varmistamiseksi. Kestotestilaitteiston pääominaisuudet esitellään työn näkökulmasta tärkeimmiltä osin ja toiminnallista turvallisuutta silmällä pitäen.

Työn tuloksena rakennettiin testiympäristö, joka on valmis siirrettäväksi kestotestiympäristöön ympärivuorokautiseen käyttöön. Kestotestin kehitystyö tulee jatkumaan ja uusien versioiden valmistuessa myös kestotestiä pitää jatkuvasti päivittää. Kestotesti tulee tarpeeseen myös uusien turvamoduulien ja -enkoodereiden valmistuessa. Työn liitteenä on työn logiikkaohjelmisto kokonaisuudessa, projektipuu sekä viestikehykset kartoitettunne muuttujineen.

Avainsanat	logiikka, toiminnallinen turvallisuus, FSO, ohjelmointi

Author(s) Title Number of Pages Date	Niklas Peltoniemi Development of long term testing for functional safety devices 51 pages + 11 appendices 12 May 2014
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	Discrete Automation
Instructor(s)	Antti Liljaniemi, Principal Lecturer Anne Kostiainen, Design Manager, M.Sc.

The goal of this thesis was to build a long term test suite for the FSO safety module for ABB Oy Low Voltage Drives. The modules new version, the release of which is in May 2014, introduces a new option for the PROFIsafe technology, that especially needs testing. The long term test platform was built for four ACS880 drives using a safety PLC.

The biggest challenge was the PLC programming, which also was the major part of the thesis work. The problems and learning related to the programming were excellent practice, even though the schedule did not quite hold and the challenges seemed to be a bit too much at times. The installation work was fluent with the skills learned at the university. The project meetings regarding the long term test served as good practice for project work.

In the theoretical part the most important concepts of functional safety are introduced. In our society today safety is ever more important, standards and directives already widely legislate how to secure industrial processes. From the R&D perspective testing is crucial for ensuring the products reliability and safety. The main features of the devices used in the long term test are presented from the test environment point of view as well as functional safety characteristics in general.

As a result of the thesis a long term test set-up, which is ready to be moved into the final testing location for continuous running, was created. The development of the set-up will be ongoing and with new versions, the test will see constant updates. The long term test will also be useful with the completion of new modules and encoders. Attached to the thesis you will find the PLC program, project tree and the message telegrams with mapped variables in its entirety.

Keywords	PLC, functional safety, FSO, programming

Sisällys

Liitteet

Liite 1. Demoräkit

Liite 2. Tavallisen ohjelman pääohjelma

Liite 3. Turvaohjelman pääohjelma

Liite 4. Ensimmäisen virhetaulukon kirjoituslohko

Liite 5. Toisen virhetaulukon kirjoituslohko

Liite 6. Kolmannen virhetaulukon kirjoituslohko

Liite 7. Neljännen virhetaulukon kirjoituslohko

Liite 8. Tarkistustaulukon kirjoituslohko

Liite 9. Globaalit muuttujat

Liite 10. Ensimmäisen taajuusmuuttajan muuttujien kartoitus PPO7-viestilohkoon

Liite 11. Ensimmäisen taajuusmuuttajan muuttujien kartoitus PS1-viestilohkoon

Lyhenteet

PFD — *Probability of failure on demand*. Yksittäisen vaarallisen tapahtuman todennäköisyys.

PFH — *Probability of failure per hour*. Arvio kuinka monta kertaa tunnissa vaarallinen tapahtuma esiintyy.

HFT — *Hardware fault tolerance*. Laitteen vikasietokyky.

IEC — *International Electrotechnical Commission* on standardeja säätävä järjestö, joka keskittyy pääasiassa sähköisten ja elektronisten laitteiden standardisointiin.

ISO — *International Organization for Standardization* on kansainvälinen standardeja säätävä elin, joka koostuu erinäisten kansallisten organisaatioiden jäsenistä.

ITU — *International Telecommunication Union*. Yhdistyneiden kansakuntien virasto, joka on vastuussa informaatio- ja telekommunikaatioon liittyvissä ongelmissa.

CEN — *European Committee for Standardization* on voittoa tavoittelematon organisaatio, joka pyrkii edistämään euroopan taloutta ja ihmisten hyvinvointi. CEN jäsenet laativat EN-standardeja.

E/E/PE — *Electrical/Electronic/Programmable Electronic*. Yleinen nimitys sähköisille, elektronisille ja ohjelmoitaville laitteille

SIL — *Safety Integrity Level*. Suomeksi turvallisuuden eheystaso (TET) on IEC 61508-standardin määrittämä, riskitason mukaan määritelty neljätasoinen turvallisuustaso turvafunktiolle.

PL — *Performance level*. ISO/EN 13849-standardin määrittelemä turvallisuustaso, joka määrittelee prosessin edellytykset suorittaa turvafunktio.

MTTF — *Mean time to failure*. Laitteen arvioitu vikaantumisaika.

IFA	*Institut für Arbeitsschutz*. Itsenäinen saksalainen turvalaitteiden tarkastajaelin.
TÜV	*Technischer Überwachungsverein*. Saksalainen organisaatio, joka ulkopuolisina konsultteina arvioivat tuotteiden turvallisuutta ja jakavat sertifikaatteja. TÜV on kansainvälisesti arvostettu ja tunnustettu.
HALT	*Highly accelerated lifetime testing*. Tuotteen luotettavuutta mittaava stressitesti, jossa altistetaan tuote kuluttaville olosuhteille.
STO	*Safe torque off*. ACS880-taajuusmuuttajaan integroitu turvatoiminto, joka katkaisee vääntömomentin moottorilta.
RAM	*Random-access memory*. RAM toimii tietokoneen työmuistina, toimii luku- ja kirjoitusmuistina tietokoneen ohjelmille ja sovelluksille.
RJ-45	Liitin, jota käytetään Ethernet-kaapeleiden kytkennässä.
RS232	Sarjaliikenneväylä. Yleisimmin käytetty, kahden laitteen välinen yksikanavaväylä.
RS485	Sarjaliikenneväylä. Monimutkaisempi, jopa 32 laitteen differentiaalinen väylä.
1oo2	Rendundanttiseksi rakennettu vertailu, jossa mittaukset ovat erilliset ja ehtona on, että yksi kahdesta riittää toteutumiseen.
PLC	*Programmable logic controller*. Logiikka, jota käytetään automaatioprosessien ohjauksessa.
Watchdog	Aikamääre, joka valvoo ohjelman suorittamiseen käytettyä aikaa. Ohjelman ylittäessä sallitun ajan, tapahtuu aikakatkaisu ja prosessi siirtyy turvatilaan.
IEEE	*Institute of Electrical and Electronics Engineers*. Teknologisten innovaatioiden edistämiseen omistautunut yhdistys.

PID	*Proportional-integral-derivate*-säädin. Yksi säätötekniikan perussäätimistä, jonka kolme toimintoa ovat suhde, integrointi ja derivointi.
LED	*Light-emitting diode*, eli hohtodiodi on puolijohdekomponentti, joka säteilee valoa, kun sen läpi johdetaan sähkövirta.
CRC	*Cyclic redundancy check*. Viestin eheyttä valvova algoritmi, joka liitetään viestiin ryhmänä bittejä.
Cat 5e	Ethernet-johto, joka on suojaamaton, kierretty parikaapeli ja yltää 100 MHz nopeuteen.
COM	Yleisnimitys sarjaportille.
NC	*Normal closed*. Tarkoittaa relelähdössä sitä, että normaalitilassa sähkövirta kulkee COM-portin läpi NC-porttiin ja releen vetäessä virta katkeaa.
SD	*Secure Digital*. Muistikortti, jota käytetään yleisesti pienelektroniikassa, kuten matkapuhelimissa, digitaalisissa kameroissa ja GPS-navigointilaitteissa.

1 Alkusanat

Haluaisin kiittää kollegoitani tuesta ja ymmärryksestä. Työn edetessä olen saanut teiltä korvaamatonta apua ja neuvoja hetkinä, jolloin ongelmat vaikuttivat ylitsepääsemättömiltä. Tiedän, että olette olleet kiireisiä, mutta te olette aina löytäneet aikaa auttaa, ettekä ole kertaakaan hylänneet minua tyhmien kysymysteni äärelle. Olette kaikki raudanlujia ammattilaisia ja on ollut suuri kunnia työskennellä kanssanne. Olen oppinut teiltä paljon ja odotan innolla tulevia haasteita osana hienoa tiimiänne, kiitos teille kaikille.

2 Johdanto

Tämän insinöörityön tarkoituksena on luoda kestotestiympäristö FSO-11-turvalaitemoduulille. Kestotestejä käytetään yrityksessä jo esimerkiksi kenttäväylämoduuleitten tuotekehityksessä ja testaamisessa, mutta turvamoduulille ei vielä ole omaa kestotestiympäristöä. Kestotestillä pyritään saamaan luotettavaa tietoa moduulin pitkäaikaisesta ajosta tuotekehityksen näkökulmasta, uusien vikojen löytämiseksi ja tiedossa olevien todennukseen. Kestotestissä ajetaan ohjelmistoa, jolla testataan turvamoduulin luotettavuutta kuormittavissa tilanteissa. Mikäli moduulissa tapahtuu virhe, kirjataan se vikalokiin. Turva-automaatiojärjestelmien sekä ylimääräisten seisokkien välttämisen kannalta moduulin luotettavuus on olennaista. Moduulin ensimmäinen versio on julkaistu jo joulukuussa 2012, mutta uusimpaan versioon toteutettu PROFIsafe-kenttäväyläkommunikointi tarvitsee laajasti testausta.

Työn tavoitteena on saada kestotestiympäristö koottua kokonaisuudessaan sekä tiedonkeruun, ohjauksen että ohjelmoinnin osalta sekä ensimmäiset mittaustulokset luettua. Ympäristö rakennetaan ABB:n Pitäjänmäen taajuusmuuttajatehtaalle testauslaboratorioon. Tiedonkeruu toteutetaan etäkäyttölaitteella siten, että laitteiston tilan ja vikalokin voi lukea selaimella toimistolta. Työn haastavin osuus tulee olemaan kestotestin PROFIsafe-yhteyden ylläpito, koska turva-automaatiolaitteiden toiminta perustuu siihen, että niiden normaalitila on seis- eli turvatila. Pienikin häiriö viestissä tai yhteydessä lukitsee järjestelmän turvatilaan, joka pitää kuitata ennen uudelleenkäynnistystä.

Teoriaosuudessa käytetyt tutkimusmenetelmät aineiston keräämisessä käytetään kvalitatiivista tutkimusta ja haastattelun kautta sovellettua, kokemuksellista tutkimusta. Kerätyn aineiston analysoinnissa käytetään induktiivista sisällönanalyysimenetelmää. Työn käytännön kokoonpano-osuudessa sovelletaan kokeellista tutkimusmenetelmää tuotekehityksessä.

3 ABB Oy

ABB on johtava sähkövoima- ja automaatioteknologiayhtymä, joka työllistää 150 000 henkilöä noin sadassa eri maassa. Pääkonttori sijaitsee Zürichissä ja se on listautunut Zürichin, Tukholman ja New Yorkin pörsseihin. ABB:n liiketoiminta koostuu viidestä divisioonasta, jotka ovat organisoituneet omille liiketoiminta-alueilleen. Yhtiön tarina nykyisessä muodossaan alkoi 1988 mutta sen historia ulottuu jopa 120 vuotta taaksepäin. (ABB Oy 1 2014.) ABB-yhtymällä on myös sinivalkoista historiaa. Se muodostui, kun ruotsalainen Asea ja sveitsiläinen Brown Boveri yhdistyivät vuonna 1988. Suomalainen Strömberg oli vain kaksi vuotta aikaisemmin siirtynyt Asealle. Strömberg toimii yhä ytimenä konsernin suomalaiselle tytäryhtiölle, ABB Oy:lle. (ABB Oy 2 2014.)

ABB:n menestystarina on pitkälti sen suuren tuotekehityskoneiston ansiota. Sillä on seitsemän tuotekehityskeskusta ympäri maailmaa ja se on jatkanut tuotekehitykseen panostamista, suhdanteista riippumatta. Tuloksena tästä on syntynyt pitkä linja innovaatioita, joista monet luovat nyky-yhteiskuntamme perustan. Korkean tasajännitteen siirto pitkillä välimatkoilla ja markkinoita mullistava laivojen propulsiojärjestelmä ovat hyviä esimerkkejä tuotteista, jotka ovat ABB:n kehittämiä ja kaupallistamia. Tänä päivänä ABB on suurin sähkömoottoreiden ja taajuusmuuttajien toimittaja teollisuuteen sekä suurin generaattoreiden toimittaja tuulivoimateollisuudelle. Lisäksi yhtiö on maailman johtava sähköverkkojen toimittaja. (ABB Oy 1 2014.)

Yhtiö rakentuu viidestä divisioonasta, jotka ala-osastoidensa kanssa muodostavat seuraavat kokonaisuudet:

- Process Automation; ohjausjärjestelmä-, mittalaite- ja turboahdinosasto
- Low Voltage Products; katkaisijat, liittimet ja mittarit
- Discrete Automation and Motion; taajuusmuuttujat, sähköelektroniikka, ohjelmoitavat logiikat, sähkömoottorit ja generaattorit sekä nivelvarsirobotit
- Power Systems; sähkönsiirto ja voimantuotanto
- Power Products; muuntajat sekä suur- ja keskijännitetuotteet. (ABB Oy 3 2014.)

Suomessa ABB työllistää noin 5500 henkilöä 30 paikkakunnalla. Tehtaat sijaitsevat Helsingissä, Vaasassa ja Porvoossa. Yhtiö on Suomen teollisuuden suurin kunnossapi-

täjä sekä yksi suurimmista työnantajista ja tuotekehitykseen käytetään noin 184 miljoonaa ja liikevaihto on noin 2,3 miljardia euroa. (ABB Oy 4 2014.)

4 Toiminnallinen turvallisuus

Turvallisuuden määritelmä on vapaus mahdollisista ihmiseen kohdistuvista fyysisten vammojen sekä terveyshaittojen vaaroista ja ympäristölle tapahtuvista haitoista (IEC61508-0, 7). Teollisuusprosesseissa on luontainen vaara henkilö-, materiaali- ja ympäristövahingoille. Nykypäivänä onkin lähes mahdoton löytää konetta tai järjestelmää jossa ei olisi hätä-seis-painiketta. Turvajärjestelmien tärkeys on alati kasvava automaatiotekniikassa. Euroopan unionissa astui voimaan vuonna 2010 konedirektiivi, jossa säädetään ohjausjärjestelmien turvallisuutta. (PROFIsafe 2.)

Toiminnallisella turvallisuudella viitataan aktiivisiin turvajärjestelmiin. Aktiiviset turvajärjestelmät eroavat passiivisista siten, että ne ovat osa turvajärjestelmää, joka on riippuvainen laitteen toiminnasta. Esimerkiksi sähkömoottorin ylilämpenemissuoja, joka rajoittaa moottorin pyörimisnopeutta, kun se havaitsee, että moottori on liian lämmin. Moottorin suojaaminen siten, että se kestää kuumuutta, ei ole toiminnallista turvallisuutta, vaikkakin sen tarkoitus on sama. (IEC61508-0, 7.)

Aktiivisia ja passiivisia turvajärjestelmiä suunniteltaessa pitää tarkastella järjestelmää kokonaisuudessaan sekä ympäristöä, jonne se sijoitetaan. Järjestelmän suunnitteluvaiheessa pitää kartoittaa laitteiston aiheuttamat henkilöihin ja ympäristöön kohdistuvat mahdolliset vaarat. Vaarojen analysoinnissa selvitetään, tarvitaanko toiminnallista turvallisuutta, jotta saavutetaan haluttu turvallisuustaso ja turva yksittäisiin vaaratilanteisiin. On ensiarvoisen tärkeää, että järjestelmään sulautetaan myös passiivisia turvajärjestelmiä, eikä toiminnallinen turvallisuus saisi yksin vastata turvallisuudesta. (IEC61508-0, 7.)

Seuraavassa esimerkki toiminnallisesta turvallisuudesta: Kuvitellaan kone, jonka suojakuvun alla pyörii terä. Terään pääsee käsiksi huoltoa ja puhdistusta varten nostamalla suojakupua. Suojakuvussa on kytkin, joka laukaisee turvapiirin, kun kupua nostetaan. Tällöin moottorin syöttö katkeaa ja jarru aktivoituu. Tällä tavalla operaattori tai huoltomies ei pääse koskemaan pyörivää terää, eikä vaaratilannetta pääse syntymään. Turvallisuuteen pyrittäessä pitää suorittaa sekä vaara-analyysi että riskien kartoitus. Vaara-analyysissä todetaan terän puhdistamisessa piilevät vaarat. Tässä tapauksessa voidaan esimerkiksi todeta, että suojakuvun nostamisen yli 5 mm ei pitäisi olla mahdollista ilman, että turvapiiri laukeaa. Analyysi voisi myös todeta, että turvapiirin laukeami-

sen jälkeen terän pysähtymisen pitää tapahtua alle sekunnissa. Yhdessä nämä muodostavat turvafunktion. (IEC61508-0, 8.)

Riskien kartoittamisessa määritellään turvafunktioiden suoritusvaatimukset. Tarkoituksena on varmistaa, että turvafunktion eheystaso on riittävä kussakin vaaratilanteessa. Eheystaso määritellään vammojen vakavuuden ja vaaratilanteen ilmenemistaajuuden mukaan. Edellisessä esimerkissä vammat saattavat olla vain mustelma tai pahimmillaan, jopa operaattorin käden irtileikkautuminen. Otetaan huomioon myös, nostetaanko kupua mahdollisesti useita kertoja työpäivän aikana vai kerran kuukaudessa. (IEC61508-0, 8.)

Eheystasoa valittaessa pitää myös tietää, onko kyseessä alhainen vai korkea vaatimustaso. Alhaisella vaatimustasolla vaaratilannetaajuus on alle ja korkealla yli yksi vuodessa. On myös olemassa jatkuva vaatimustaso, joka tarkoittaa, että järjestelmä palaa aina turvalliseen normaalitilaan. (IEC61508-4, 21.) Alhaisella vaatimustasolla lasketaan järjestelmälle PFD-arvo, joka kuvaa yksittäisen vaarallisen tapahtuman todennäköisyyttä (kuva 1). Korkealla vaatimustasolla lasketaan järjestelmälle PFH-arvo, joka tarkoittaa, kuinka monta kertaa tunnissa vaarallinen tapahtuma esiintyy (kuva 1). Arvot saadaan laskemalla yhteen eri komponenttien vikaantumisten, inhimillisten erehdysten, ja erinäisten järjestelmään vaikuttavien häiriöiden todennäköisyyksiä. (IEC61508-1, 33 – 34.)

Safety integrity level (SIL)	Average probability of a dangerous failure on demand of the safety function (PFD_{avg})
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$
Safety integrity level (SIL)	**Average frequency of a dangerous failure of the safety function [h^{-1}] (PFH)**
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

Kuva 1. Alhaisen vaatimustason eheystaulukko (PFD) ja korkean vaatimustason eheystaulukko (PFH) (IEC61508-1, 33 - 34).

Toiminnallisen turvallisuuden siirtyessä mitä suurenevissa määrin sähköisiin ja ohjelmoitaviin piireihin, aiheuttaa se haasteita järjestelmien monimutkaisuudessa. Järjestelmästä on käytännössä mahdotonta löytää jokainen häiriönaiheuttaja, tai testata jokainen mahdollinen käyttäytymismalli. Vaikkakin turvallisuuden suorituskykyä on vaikea ennustaa, on testaus silti oleellista. Suunnittelussa haastavuutta luo vaarallisten häiriöiden estäminen ja niiden sattuessa seurausten hallitseminen. Tällaisia häiriöitä ovat satunnaiset laiteviat, jännitehäiriöt, ohjelmistoviat sekä inhimillisestä erehdyksestä ja ulkoisista tekijöistä aiheutuvat häiriöt. (IEC61508-0, 8 – 9.)

4.1 Standardit

Standardit eivät ole lakeja, vaan yhteisiä toimintatapoja. Standardisoinnilla pyritään parantamaan yhteensopivuutta, turvallisuutta, kuluttajan suojelua, ympäristöä sekä helpottamaan kauppaa (Standardit 1). Standardisointijärjestöt sijoittuvat kolmeen tasoon, jossa ylimmällä tasolla on kansainväliset IEC-, ISO- ja ITU-järjestöt, keskimmäisellä tasolla on eurooppalaiset järjestöt ja alimmalla kansallisen tason järjestöt. Eurooppalainen CEN-järjestö velvoittaa kaikkia jäsenmaita vahvistamaan sen laatimat standardit, jotta samat standardit olisivat voimassa koko Euroopan alueella. CEN:n

laatimista standardeista käytetään lyhennettä EN. Kansainvälisistä järjestöistä ISO on laajin ja siihen kuuluu standardisointijärjestöjä jopa 163 maasta. Noin kolmekymmentä prosenttia CEN:n standardeista perustuvat kansainvälisiin ISO-standardeihin. Tällöin standardille tulee nimeksi EN ISO. (Standardit 2, Standardit 3.)

Kattostandardina toiminnalliselle turvallisuudelle toimii IEC 61508-standardi. Standardi koskee turvalaitteita, jotka ovat toteutettu sähköisesti, elektronisesti tai ohjelmoitavalla elektroniikalla. Tällaiset turvalaitteet kantavat nimitystä E/E/PE. Standardia sovelletaan tällä tekniikalla toteutettuihin järjestelmiin, käyttösovelluksesta riippumatta. Se tähtää parantamaan E/E/PE-teknologian turvallisuutta ja taloudellisuutta, antamaan teknologiakehitykselle yleisen turvallisuuskehyksen ja takaamaan käyttäjille selkeän sekä turvallisen operoinnin. (IEC61508-0, 9.) IEC 61508 määrittelee myös SIL-tasot, jotka kuvaavat turvafunktion eheyttä. SIL1-taso on matalin ja SIL4-taso korkein eheystaso. (IEC61508-0, 11.) SIL4-tasoon vaaditaan todella laajamittaiset ja suojaavat turvallisuusjärjestelmät, jotta tiukat vaatimukset täyttyvät. Alhaisella vaatimustasolla SIL4-eheystason vaatimukset vaarallisen tapahtuman todennäköisyydelle on vähintään yksi kymmenestätuhannesta. Korkealla vaatimustasolla SIL4-tasoon vaaditaan, että vaarallinen tilanne tapahtuu korkeintaan 0.00000001 kertaa tunnissa, eli noin 11415 vuoden välein. (IEC61508-1, 33 – 34.)

Toinen tärkeä IEC-standardi on 61511, joka käsittelee toiminnallista turvallisuutta prosessiteollisuudessa. Siinä missä IEC 61508- koskee laitevalmistajia, säätää IEC 61511-standardi instrumentointijärjestelmien suunnittelua ja käyttöä. (IEC61511-1, 11.) Standardi tarkentaa vaatimuksia järjestelmän arkkitehtuurille, laitteen rakenteelle, sovellusten ohjelmistolle ja järjestelmäintegroinnille. Se koskee toiminnallista turvallisuutta, jolla pyritään turvafunktioiden instrumentoinnilla suojelemaan työntekijöitä, ulkopuolisia henkilöitä tai ympäristöä. IEC 61511 määrittelee myös instrumentoinnin SIL-tasojen kehykset, mutta ei erittele niiden vaatimuksia tiettyihin järjestelmiin. (IEC61511-1, 9 – 10.)

ISO 13849-standardi pyrkii riskien vähentämiseen koneturvallisuudessa. Standardissa määritellään suoritustasot (PL) turva-automaatiojärjestelmälle. PL-tasot valitaan PFH-arvojen, diagnostiikkakattavuuden (DC) ja vaarallisen vikaantumisen keskiajan ($MTTF_d$-arvo) avulla. DC on järjestelmän oma vianhavaitsemisdiagnostiikka joka on jaettu neljään tasoon (kuva 2). Se tarkoittaa sitä, millä todennäköisyydellä järjestelmää huomaa vaaralliset laitteistoviat. Koska 100 %:n vianhavaitseminen ei käytännössä ole mahdol-

lista, on se rajoitettu 99 %:iin. Alle 60 %:n arvot eivät merkittävästi heikennä järjestelmän luotettavuutta, joten ne jätetään arviosta ulkopuolelle (kuva 2). (ISO13849-1: 11, 16.)

PL	SIL (IEC 61508-1, for information) high/continuous mode of operation
a	No correspondence
b	1
c	1
d	2
e	3

DC	
Denotation	Range
None	DC < 60 %
Low	60 % ⩽ DC < 90 %
Medium	90 % ⩽ DC < 99 %
High	99 % ⩽ DC

Kuva 2. PL-suoritustasot ja diagnostiikkakattavuustaulukko (ISO13849-1: 16, 18).

Järjestelmän MTTF lasketaan jokaiselle komponentille erikseen annettujen vikaantumisaikojen avulla. MTTF-arvo lasketaan jokaiselle kanavalle erikseen ja se määrittää kanavan keskimääräisen vikaantumisvälin (kuva 3). Alle kolmen vuoden arvoja ei oleteta löytyvän, koska tämä tarkoittaisi että noin 30 % turva-automaatiojärjestelmistä pettäisi ensimmäisen vuoden jälkeen. Yli 100 vuoden arvoja ei voida ottaa huomioon, koska korkean riskin turva-automaatiojärjestelmät eivät saisi perustua yksinomaan komponenttien luotettavuuteen, vaan ne pitäisi rakentaa redundanttisiksi ja testauttaa tasaisin väliajoin. (ISO13849-1: 17.)

$MTTF_d$	
Denotation of each channel	Range of each channel
Low	3 years ⩽ $MTTF_d$ < 10 years
Medium	10 years ⩽ $MTTF_d$ < 30 years
High	30 years ⩽ $MTTF_d$ ⩽ 100 years

Kuva 3. $MTTF_d$-taulukko (ISO13849-1: 17).

4.2 Direktiivit

Työn ja toiminnallisen turvallisuuden kannalta tärkeä EU-direktiivi on koneturvallisuusdirektiivi 2006/42/EY. Direktiivissä säädetään ohjausjärjestelmien turvallisuutta ohjauslaitteiden, pysäyttämisen, käynnistämisen sekä ohjaus- ja toimintatapavalintojen kautta. Ohjausjärjestelmät pitää suunnitella siten, että ne kestävät tarkoitetut rasitukset eikä ohjelmisto-, laitteisto- ja logiikkaviat saa aiheuttaa vaaratilanteita, myös kohtuudella ennakoitavissa olevat inhimilliset erehdykset pitää ottaa huomioon, eivätkä ne saa aiheuttaa vaaratilanteita. Direktiivin tärkeimmät kohdat käsittelevät koneen odottamatonta käynnistystä, ominaisarvojen säilymistä käytön aikana, pysäytyskäskyn dominoivuutta, laitteistosta sinkoavien osien estämistä ja turvalaitteiden luotettavuutta. (2006/42/EY-Direktiivi:L 157/ 37 – 38.)

4.3 PROFIsafe

PROFIsafe on PROFIBUS and PROFINET International-yhteisön (PI) kehittämä turvakommunikointiväylä. Se on integroitu turvateknologia kaikkiin kappaletavara- ja prosessiautomaatiojärjestelmiin, jotka käyttävät PROFIBUS- tai PROFINET-tekniikoita. Väylän kommunikointi on itsenäistä tavallisesta väyläliikenteestä ja tarjoaa kustannustehokkaan sekä joustavan tavan toiminnalliselle turvallisuudelle. (PROFIsafe 1.) PROFIsafe noudattaa IEC 61508-standardia ja on maailmalla laajasti käytetty turvaväylä. Siitä on itsessään muodostunut kansainvälinen standardi (IEC 61784-3-3) ja se on saksalaisten sertifiointijärjestöjen IFA:n ja TÜV:n hyväksymä. Turvaväylä kattaa koko kommunikointimatkan kenttälaitteelta, ohjausyksikön kautta, toimilaitteelle ja integroi turvaväylän tavallisen kenttäväylän kanssa samaan kaapeliin. (PROFIsafe 2.)

PROFIsafe käyttää Black Channel-periaatetta, joka tarkoittaa, että tavallinen kenttäväylä ei häiritse turvaväylää millään tavalla. Viesti koostuu kahdesta erillisestä kerroksesta, turva- ja tavallisesta kerroksesta. Turvakerros on itsenäinen, oli kyseessä sitten kuparijohto, valokaapeli, langaton yhteys tai liitinalusta. Myöskään siirtonopeuksilla tai virheenhavaitsemislaitteistolla ei ole väliä, PROFIsafe näkee vain ”mustan kanavan”. Turvaväylän viestit ovat aina dominoivia ja ne varmistavat yhteyden aina signaalin lähteestä, sen prosessointitasolle asti. PROFIsafe-protokollaa voidaan käyttää SIL3-tasoisiin turvajärjestelmiin, IEC61508-standardin mukaisesti. (PROFIsafe 3.)

5 Kestotestaus ja vanhentaminen

5.1 Ohjelmistojen luotettavuus

Tietokoneen saapumisen jälkeen, kuusikymmentä vuotta sitten, olemme tulleet riippuvaisiksi koneista jokapäiväisessä elämässämme. Nykypäivänä tietokoneita löytyy lähes jokaisesta laitteesta, jota käytämme. Niitä löytyy rannekelloista, puhelimista, kodinkoneista, rakennuksista, autoista ja lentokoneista. Useilla aloilla tietokoneisiin pitää pystyä luottamaan varauksetta niiden luotettavan toiminnan kannalta sekä ihmishenkien ja ympäristötuhojen säästämiseksi. Esimerkiksi avaruussukkulassa on sukkulaan ohjelmoitu yli puoli miljoonaa riviä koodia, kun sitä ohjaavaan valvomoon on ohjelmoitu kolme ja puoli miljoonaa koodiriviä. Sukkulan toiminnan sekä jokaisen matkustajan turvallisuuden kannalta on ensiarvoisen tärkeää, että jokainen rivi toimii moitteetta. (Lyu 1996: 3.)

Jokainen tietokone nykyään sisältää massiivisen määrän ohjelmaa. Kehittyneen ja monimutkaisen ohjelman tarve on kasvanut nopeammin kuin niiden suunnittelu, implementointi, testaaminen tai ylläpito on mahdollista. Kun tietokoneille asetetaan näin kovia tavoitteita, myös niiden pettämisestä aiheutuvien kriittisten onnettomuuksien todennäköisyys kasvaa. Pienimmillään kodinkoneiden vikaantuminen aiheuttaa epämukavuutta, kun vakavimmillaan tietokoneiden pettäminen voi aiheuttaa ydinvoimalan reaktorin sulamisen ja suuren ympäristökatastrofin. On siis sanomattakin selvää, että ohjelmistojen luotettava toiminta on yhteiskunnassamme suuri huolenaihe. (Lyu 1996: 4.)

Ohjelmiston testaaminen on varsinkin tuotekehityksessä tärkeää, mutta myös koko tuotteen eliniän ajan tarpeellista. Kestotestissä tärkein testattava ominaisuus FSO-moduulin toiminnan sekä turvallisuuden kannalta on kuormituksen kestokyky. Ääriolosuhteiden muuttuessa pitää varmistaa, että moduuli ei laukaise turvatiloja turhaan mutta turvalliseen toimintaan pitää silti pystyä luottamaan. Moduulissa on paljon sisäistä diagnostiikkaa, joka valvoo kahden prosessorin välisiä eroavaisuuksia ja tiedon prosessointia. Kestotestissä valvotaan moduulin sisäisiä sekä turvaväylän viiveaikoja kovan kuormituksen alla. (Kostiainen 2014.)

5.2 Luotettavuustekniikka

Reliablility Engineering, eli luotettavuustekniikka, tarkoittaa systemaattista, tuotteen koko käyttöikää käsittelevää, sovellettavaa testaamista sekä kehitystä ja on täten keskeinen osa tuotteen eliniän hallintaa. Tavoitteena on varmistaa tuotteen tai prosessin toimintavarmuus määritetyllä käyttötavalla, sekä parantaa luotettavuutta. Realistisesti ajateltuna, kaikkia virheitä tuotteessa ei pystytä eliminoimaan, joten toinen luotettavuustekniikan tavoite on kartoittaa todennäköisimmät virheet ja häiriöt, jotta mahdolliset haitat voidaan minimoida. Tekniikan primääritavoite tulee aina olla luotettavuusongelmien identifioimisessa mahdollisimman aikaisessa vaiheessa. Mikäli virheet voidaan tunnistaa tuotekehitysvaiheessa, tulee niiden korjaaminen monin kerroin halvemmaksi, kuin jo markkinoilla olevan tuotteen kehittäminen tai korjaaminen. (Reliability Engineering 2014.)

On useita syitä, miksi luotettavuus on tärkeä ominaisuus tuotteessa, tärkeimmät ovat:

- Maine; tuotetta valmistavan yhtiön maine on erittäin läheisesti yhteydessä heidän valmistamien tuotteiden laatuun.
- Asiakastyytyväisyys; tuotteiden luotettavuus on elinehto tyytyväisille asiakkaille. Epäluotettava tuote vaikuttaa erittäin negatiivisesti asiakastyytyväisyyteen.
- Takuukustannukset; mikäli tuote ei kestä takuuaikaa, joudutaan se korvaamaan uuteen, mikä puolestaan syö voittoja.
- Asiakkaiden pysyminen; pyrkimys tuotteen laadun parantamiseen on olemassa oleville asiakkaille merkki siitä, että valmistaja on omistautunut tuotteelle ja pyrkii huolehtimaan asiakkaiden tyytyväisyydestä.
- Kustannusanalyysi; luotettavuustekniikasta saatuja tuloksia voidaan laskea muiden kustannusten kanssa yhteen kustannustehokkuuden laskemiseksi. Kustannusanalyysi voi todeta, että vaikka tuotteen valmistaminen on kalliimpaa, tulevat sen elinikäkustannukset halvemmaksi kuin, esimerkiksi kilpailijan vastaava tuote.
- Asiakasvaatimukset; Monet asiakkaat vaativat toimittajan suorittavan luotettavuustekniikkaa.
- Kilpailukyky; yhtiöt usein julkaisevat luotettavuustekniikan tuloksia saavuttaakseen etua kilpailijoihin, jotka eivät julkaise tuloksia tai joilla on heikommat tulokset. (Reliability Engineering 2014.)

Luotettavuutta ei saa sekoittaa laatuun. Vaikka tuote olisi laadukkaasti suunniteltu, voi se silti tuotannosta tullessa kenttäolosuhteissa olla epäluotettava. Epäluotettavuuden syy on tällöin todennäköisesti valmistuksen huono laatu. Tuotteen ollessa luotettava suunnittelunsa puolesta, voi siis lopullinen tuote kuitenkin olla täysin epäluotettava kentällä. Esimerkiksi kylmäjuotos voisi läpäistä alustavat testit valmistajalla, mutta myöhemmin kenttäkäytössä pettää, lämmönvaihteluiden tai tärinän takia. Tällaisesta virheestä ei voi yksinomaan syyttää suunnittelua, vaan syy voi myös olla ala-arvoisen valmistusprosessin. Aivan kuten ketju on yhtä vahva kuin sen heikoin lenkki, on tuote yhtä luotettava kuin sen suunniteltu luotettavuus ja valmistuksen laatu. (Reliability Engineering 2014.)

5.3 Vanhentaminen

Highly accelerated lifetime testing (HALT) tarkoittaa tuotteen tahallista altistamista kuluttaville olosuhteille haittapuolien ja heikkouksien löytämiseksi. Tällaisia olosuhteita voivat olla esimerkiksi altistaminen kylmyydelle tai kuumuudelle, nopeille lämpötilanvaihteluille tai tärinälle. (Qualmark 2 2014.) Kokeita suoritetaan uusien mallien, komponenttien ja valmistusprosessien parantamiseksi ja kehittämiseksi. HALT-kokeen tarkoitus on löytää kaikki mahdolliset vikaantumistavat jotta ne voidaan kitkeä ja näin ollen parantaa elinikää ja luotettavuutta. (Qualmark 1 2014.)

Kuluttavien olosuhteiden altistamisen yhteydessä suoritetaan käytännön kokeita, tuotteen toimivuuden toteamiseksi. Kun vika ilmenee, tallennetaan vikaantumistiedot ja tuotteeseen sillä hetkellä vaikuttaneet rasitukset. Vikatiedoista paikannettu, vikaantumisen aiheuttanut, syy korjataan ja kokeita jatketaan, yhä uusien vikojen paikallistamiseksi. Kun vikaantumisia ilmenee ja niitä korjataan, saadaan tuotteen elinikää ja luotettavuutta parannettua. (Qualmark 2 2014.)

6 Työssä käytetty laitteisto

6.1 Taajuusmuuttaja

Tässä insinöörityössä käytetty taajuusmuuttaja on ABB:n valmistama ACS880-01. Sitä käytetään asynkronisten AC-moottoreiden, synkronoitujen kestomagneettimoottoreiden sekä AC-induktioservomoottoreiden ohjaukseen. ACS880 on ABB:n uusin taajuusmuuttajamalli, joka tuotiin markkinoille vuonna 2012. Se on suunniteltu yleiskäyttöön ja sen rakenne on käyttötarkoituksesta riippuen muunneltavissa erinäisten kenttäväylä-, enkooderi- ja runkomoduuleiden avulla. (ACS880 Sales: 4.)

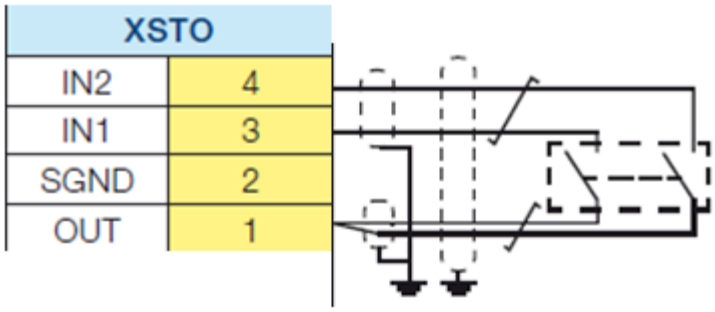
ACS880 mahdollistaa melkein kaikkien mahdollisten vaihtosähkömoottoreiden ohjauksen ja kaikkiin automaatioväyliin sekä etäkäyttöverkkoihin kytkeytymisen. Taajuusmuuttajalle voidaan syöttää kaikkea aina 230 V:sta, 690 V:n jännitteisiin ja yltää maksimissaan jopa 6 MW:n tehoon. Seinäasennettavalla ACS880-01-mallilla voidaan päästä 250 kW:n tehoon (kuva 4). (ACS880 Sales: 7, 10.) ACS880-ohjauskortilla on useita liitäntämahdollisuuksia tiedonkeruuseen, ohjaukseen ja väyläkytkentöihin. Laajojen liitäntämahdollisuuksien lisäksi siinä on kolme kytkentäpaikkaa, joihin voi halutessaan liittää kenttäväylä-, lisä-I/O-, tiedonkeruu- tai turvamoduuleita. (ACS880 single drives, catalog: 17.)

Kuva 4. ACS880-01-taajuusmuuttaja ilman koteloa. Liitettynä FPBA-01-PROFIBUS-kenttäväylämoduuli, FSO-11-turvamoduuli ja FEN31-enkooderi.

Taajuusmuuttajalle on tehty mahdolliseksi IEC-ohjelmointi. Se tarjoaa ohjelmoitavan logiikan toiminnot ilman lisälaitteita ja on täten täydellinen tapa räätälöidä taajuusmuuttaja prosessin vaatimusten mukaan. Ohjelmointi perustuu IEC61131-3-standardissa määriteltyyn CODESYS-ympäristöön ja samaan Control Builder Plus-ohjelmistoon, kuin mitä AC500-logiikka käyttää. Koska ohjelmointi voidaan toteuttaa taajuusmuuttajan sisällä mahdollistaa se säästön ylimääräisten laitteistojen hankintakustannuksista, nopeamman ja luotettavamman ohjauksen kentälle hajauttamisen myötä, paremman luotettavuuden komponenttien ja asennusten vähenemisellä, tutun ohjelmointiympäris-

tön ja asennukset saman suojakotelon alle, pienempään tilaan. (ACS880 CODESYS: 1 - 2.)

ACS880-taajuusmuuttajaperheessä on sisäänrakennettuna STO-turvatoiminto, joka aktivoituessaan katkaisee vääntömomentin moottorin akselilta, sitä käytetään hätä-seis-tilanteissa, sekä ylinopeus- ja ylilämpenemis-suojana (ACS880 integroidut turva-toiminnot 2014.) Toiminto täyttää kansainväliset IEC- sekä ISO-standardien mukaiset SIL 3-tasoon tarvittavat vaatimukset ja on TÜV-sertifioitu. Turvajärjestelmän keltainen XSTO-sovitin sijaitsee taajuusmuuttajan ohjauskortilla, vasemmassa alanurkassa (kuva 5). (ACS880 STO: 4 – 6.)



Kuva 5. ACS880-taajuusmuuttajan STO:n liitäntärakenne (ACS880 STO: 6).

STO on rakenteeltaan redundanttinen, siinä on kaksi toisistaan erillistä kanavaliityntää, joiden standardinmukainen vikasietokyky (HFT) on 1. Sovittimen IN1- ja IN2-pinnille syöttö voidaan toteuttaa joko ulkoisella tai sisäisellä virtalähteellä. Mikäli käytetään taa-juusmuuttajan sisäistä virtalähdettä, kytketään XSTO-sovittimessa ensimmäinen liityntä kolmanteen ja neljänteen. STO voidaan myös liittää mihin tahansa turvareleeseen, -logiikkaan tai -painikkeeseen. Signaalin katketessa, siirtyy taajuusmuuttaja turvatilaan alle 20 millisekunnissa. Mikäli kanavien signaalit eroavat yli 200 millisekuntia, turvapiiri laukeaa.(ACS880 STO: 6, 8, 10.)

6.2 Logiikkakokonaisuus

Työssä käytetty logiikkakokonaisuus koostuu PM583-ETH-logiikasta, SM560-S-turvalogiikasta ja CM579-PNIO-moduulista. ABB:n logiikat on suunniteltu moduläärisik-si siten, että kokonaisuus kootaan tarpeen mukaan. Logiikka kootaan liitinalustan pääl-le liittämällä haluttu keskusyksikkö mitoitetun laskentatehon ja tarvitun muistin mukaan

sekä valitsemalla tarvittavat kenttäväylämoduulit ja I/O-pisteet. Työssä käytetään TB521-liitinlevyä, jossa on kaksi lisämoduulipaikkaa keskusyksikön lisäksi. Ensimmäiselle liitinpaikalle, keskusyksikön viereen, kiinnitetään turvalogiikka, joka toimii PROFIsafe-väylän masterina. Toiselle liitinpaikalle kiinnitetään PROFINET-master, joka toimii kenttäväylämoduulina.

6.2.1 Prosessorikeskusyksikkö

PM583-ETH-keskusyksikkö sijoittuu laskentateholtaan noin keskikastiin ABB:n prosessorivertailussa. Keskusyksikön etupaneelista löytyy taustavalaistu 7-segmenttinäyttö, kolme tilaa indikoivaa LED-valoa, muistikorttipaikka, kahdeksan painonappia ja paikka litiumakulle (kuva 6). Sitä voidaan käyttää AC500-laitteistossa master-tyyppisesti useissa eri kenttäväylissä ja verkoissa, slave-tyyppisesti PROFIBUS-, DeviceNet- ja CANopen-väylissä liitinalustan FieldBusPlug-liitännän kautta tai täysin itsenäisenä prosessoriyksikkönä. (PM583-ETH.)

Kuva 6. PM583-ETH-prosessorikeskusyksikkö.

Prosessorikeskusyksikön litiumakku mahdollistaa RAM-muistin ja sisäisen kellon säilymisen virtakatkon aikana. Se ei vaadi akkua toimiakseen, mutta sen käyttö on suositeltavaa tallennetun tiedon säilyttämiseksi. Akun varausta valvotaan ja matalasta varauksesta varoitetaan noin kaksi viikkoa ennen varauksen loppumista. Muistikorttia käytetään käyttäjätietojen ja ohjelmien tallentamiseen, sekä prosessorin sisäisen ohjelmiston päivitykseen. Keskusyksikkö käyttää standardinmukaista tiedostojärjestelmää, mikä

mahdollistaa muistikortin lukemisen tavallisimmissa muistikortinlukijoissa, esimerkiksi tietokoneessa. (PM583-ETH.)

AC500-logiikan ohjelmointi tapahtuu PS501 Control Builder Plus -ohjelmiston avulla. Ohjelmistoon kuuluu konfigurointi-, ohjelmointi-, diagnostiikka-, huolto- ja valvomotyökalut sekä laaja kirjastohakemisto ohjelmointiin. Ohjelmointi perustuu CoDeSys-ympäristöön, mikä on määritelty omassa IEC61131-3 -standardissansa. PS501-ohjelmiston avulla voidaan myös parametrisoida etänä useita taajuusmuuttajia yhtä aikaa logiikan kautta. (AC500 PLC: 13.)

Liitinalustan voi valita kolmesta vaihtoehdosta; TB511 (kuva 7), yhdellä moduulipaikalla, TB521, kahdella moduulipaikalla tai TB541, neljällä moduulipaikalla. ETH-pääte tarkoittaa sitä, että alustassa on liityntä Ethernet-kommunikointia varten, liitinalustoja löytyy myös Arcnet-kommunikointiin. I/O-moduuleja voi liittää, liitinalustasta riippumatta, kymmenen keskusyksikön oikealle puolelle. Liitinalustasta löytyy Ethernet RJ-45-liitin PLC-PC-kommunikaatiolle, kaksi COM-porttia RS232-, RS485- ja Modbus-liitännille sekä FieldBusPlug-liitin Profibus DP, CANopen ja DeviceNet-väylille. (AC500 PLC: 14.) Liitinalustan käyttö mahdollistaa sen, että mikäli myöhemmin tarvitaan enemmän laskentatehoa, uutta väyläkommunikaatiota tai lisätä I/O-pisteitä, voidaan ne kätevästi toteuttaa pelkän prosessorin ja väylämoduulin vaihdolla tai lisäämällä I/O-moduuli liitinalustaan.

Kuva 7. TB511-ETH-liitinalusta.

6.2.2 Turvalogiikka

AC500-S on 1oo2-periaatteella toimiva, SIL 3 -tasoon yltävä turvalogiikka (kuva 8). 1oo2-periaate tarkoittaa redundanttista järjestelmää, jonka vikasietokyky on 1, eli yksi kahdesta riittää turvatilan laukaisemiseen. Jotta 1oo2 toteutuisi, logiikassa on kaksi mikroprosessoria, jotka suorittavat turvaohjelmaa omalla muistialueellaan. Syklin lopussa lopputuloksia verrataan ja eroavaisuuksien tai virheilmoitusten sattuessa järjestelmä palaa turvatilaan. Yhdenkin prosessorin vikaviesti riittää turvatilan laukeamiseen. Turvalogiikka liitetään osaksi AC500-logiikkakokonaisuutta kommunikointimoduulin ja tavallisen PLC:n väliin logiikkaräkkiin. Vaikka turvalogiikka liitetään samaan kokonaisuuteen, ei tavallinen logiikkaliikenne aiheuta sille häiriöitä, eikä virhe tavallisessa vaikuta turvalogiikan ohjelmakiertoon. Koska turvalogiikka on integroitu tavalliseen, pätevät siinä samat rakenteelliset, ohjelmointi-, ja konfigurointiominaisuudet. (AC500-S Safety User Manual: 14.)

Tieto liikkuu PROFIsafe-turvaväylässä ja kaikki kommunikointi tapahtuu tavallisen logiikan kautta käyttäen ”black channel” -periaatetta. Toimiakseen, turvalogiikka tarvitsee vierelleen PROFINET I/O-Masterin, jonka kautta myös turvaväylän liikenne ohjataan. Mikäli viestissä ilmenee virhe, siirtyy turvalogiikka SAFE STOP -tilaan, jonka seurauksena turvajärjestelmä menee turvatilaan, kun watchdog-aika on kulunut loppuun. (AC500-S Safety User Manual: 22, 24.)

Turvalogiikassa ei ole paristoa, joten kaikki ohjelmalliset funktiot ja muuttujat tunnistetaan, kun syöttöjännite kytketään päälle. Kaikki itsetestit ja diagnostiikkafunktiot sekä käynnistyksessä että ohjelmakierrossa suoritettavat, kuten prosessori- ja muistitestit sekä ohjelman sujuvuuden hallinta, ovat logiikassa implementoituina kansainvälisen IEC61508-standardin vaatimusten mukaan. SM560-S suorittaa yhtä tehtävää kerrallaan ohjelmakierron aikana. Ohjelmasyklille ei ole määritelty aikaa, mutta käyttäjä voi halutessaan ottaa käyttöön watchdog-funktion, joka valvoo ohjelmakiertoon käytettyä aikaa. Watchdog-ajaksi asetetaan suurin sallittu sykliaika, jonka ylittäessään ohjelmasykli pysäytetään ja järjestelmä ajetaan turvatilaan. (AC500-S Safety User Manual: 36 – 37.)

Kuva 8. SM560-S-turvalogiikka.

Tavallisella ja turvalogiikalla on omat sulautetut ohjelmat, boottiprojektit ja ohjelmakierrot, jotka suoritetaan itsenäisesti. Ainoa yhteinen ohjauksellinen ominaisuus on tavallisen PLC:n ”Run”- ja ”Stop” -painikkeet, jotka käynnistävät ja pysäyttävät sekä tavallisen että turvalogiikan. Tämä ei kuitenkaan laukaise turvapiiriä, eikä pysäytyskäsky näin ollen lopeta PROFIsafe-kommunikaatiota tai siirrä järjestelmää turvatilaan. (AC500-S Safety User Manual: 54.)

SM560-S-turvalogiikalla on PFH-arvo < 3 * 10^-9, mikä vastaa karkeasti turvalogiikan pettämistä noin 40000 vuoden välein. Logiikan käyttöikä on kuitenkin vain 20 vuotta, mikä tarkoittaa, että jokainen turvalogiikka pitää vaihtaa uuteen viimeistään viikkoa ennen sen käyttöiän päättymistä. (AC500-S Safety User Manual: 19 – 20.)

6.2.3 Logiikan kenttäväylämoduuli

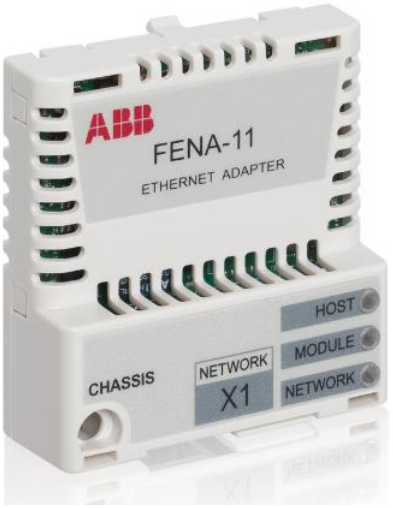
CM579-PNIO on AC500-logiikan kenttäväylämoduuli PROFINET-liikennöintiin (kuva 9). Se käyttää PROFINET IO real-time-protokollaa. Moduuli voidaan liittää kaikkiin liitinalustoihin ja se tukee kaikkia PM57x-, PM58x- ja PM59x-sarjan prosessorikeskusyksiköitä. Kommunikointi prosessorin kanssa tapahtuu liitinalustaan integroidun kommunikaatioväylän kautta. Moduuli käyttää dual-port RAM-muistia (DPRAM), mikä mahdollistaa yhtäaikaisen lukemisen ja kirjoittamisen. Siinä on myös sisäinen, 10/100 MBit/s-nopeutta tukeva, Ethernet-kytkin porttien sekä porttien ja prosessorin väliseen liikennöintiin. (CM579-PNIO.) Työssä käytettävässä logiikkakokonaisuudessa moduuli asennetaan liitinalustaan, turvalogiikan vasemmalle puolelle.

Kuva 9. CM579-PNIO-PROFINET-kenttäväylämoduuli

Kenttäväylämoduulin paneelista löytyy viisi LED-merkkivaloa, kaksi kiertokytkintä osoitteen asettamiseen ja kaksi RJ-45-porttia Ethernet-kommunikointia varten (CM579-PNIO).

6.3 Taajuusmuuttajan kenttäväylämoduuli

FENA-21 on ABB:n valmistama, F-moduulituoteperheeseen kuuluva, taajuusmuuttajaan kiinnitettävä Ethernet-kenttäväyläliitännäinen (kuva 10). Moduuli on yhteensopiva kaikkien ABB:n taajuusmuuttajien sekä IEEE 802.3- ja IEEE 802.3u-standardien kanssa. Se tukee Modbus/TCP- sekä UDP-, EtherNet/IP- ja PROFINET I/O-protokollia. (FENA-11 User´s Manual: 21 - 22.) Moduulin kautta voi lähettää taajuusmuuttajalle ohjauskäskyjä ja referenssiarvoja, säätää PID-arvoja, lukea tietoja sekä kuitata virheen taajuusmuuttajassa. Se tukee 10 ja 100 Mbit/s-siirtonopeuksia ja tunnistaa automaattisesti verkossa käytetyn nopeuden. (FENA User´s Manual: 35.)



Kuva 10. FENA-11-kenttäväylämoduuli.

FENA-21 eroaa FENA-11-moduulista siten, että siinä on kaksi RJ-45-porttia, mikä mahdollistaa taajuusmuuttajien liittämisen sarjaan. Se kiinnitetään yhteen taajuusmuuttajan vapaista liitäntäpaikoista ja kiristetään ruuvilla, joka löytyy moduulin vasemmasta alareunasta. Kiristysruuvi myös maadoittaa moduulin. Ethernet-johto kiinnitetään RJ-45-tyyppiseen X1-liittimeen. Mikäli halutaan useampi taajuusmuuttaja sarjaan, voidaan se FENA-21-moduulilla toteuttaa siten, että liitetään moduuli X2-liittimestä seuraavan taajuusmuuttajan moduulin X1-liittimeen. Moduuli toimii tällöin myös kytkimenä X1- ja X2-liittimien välillä. Oikeasta alareunasta löytyy kolme diagnostiikkamerkkivaloa, jotka osoittavat moduulin tilan. (FENA-11 User´s Manual: 36.)

6.4 Taajuusmuuttajan turvamoduuli

FSO-11 on ACS880-taajuusmuuttajaan kiinnitettävä turvamoduuli toiminnalliseen turvallisuuteen (kuva 11). Se tukee sekä ABB:n valmistamaa AC500-S-turvalogiikkaa että Siemensin SIMATIC S7-logiikkaa. FSO ei ohjaa taajuusmuuttajaa, vaan valvoo sen toimintaa ja käskee turvafunktioiden käynnistymisen. Pyyntö turvafunktiolle voi tulla ulkoisesta lähteestä, kuten painonapista, turvalogiikalta tai moduulin sisäisestä virheestä. Mikäli taajuusmuuttaja ei tottele turvafunktioiden täytäntöönpanokäskyjä, laukaisee turvamoduuli taajuusmuuttajan STO-funktion. Taajuusmuuttajan ollessa yhteydessä turvalogiikkaan, turvataan väyläyhteys PROFIsafe-yhteydellä. Logiikka on tällöin yhteydessä taajuusmuuttajan FENA-kenttäväyämoduuliin, joka kommunikoi FSO-moduulin kanssa. (FSO-11 User´s manual: 24.)

Laitteella voidaan luoda ACS880-järjestelmään seuraavat turvafunktiot:

- *Safe torque off* (STO) on vakio turvafunktio ACS880-taajuusmuuttajissa. Asettaa taajuusmuuttajan turvalliseen tilaan ja estää vahingossa käynnistymisen.
- *Safe brake control* (SBC) käytetään STO-toiminnon kanssa. Luo turvallisen lähdön ulkoisten mekaanisten jarrujen ohjaukseen.
- *Safe stop 1* (SS1) pysäyttää moottorin turvallisesti rampin avulla ja moottorin pysähdyttyä, käynnistää STO-funktion.
- *Safe stop emergency* (SSE) pysäyttää koneiston joko välittömällä STO-funktiolla, tai rampilla. Moduulin sisäinen monitorointi voi laukaista SSE-funktion ilman ulkoista signaalia.
- *Safely-limited speed* (SLS) estää moottoria ylittämästä säädettyä nopeusrajoitusta.
- *Variable Safely-limited speed* (SLS) vaatii turvalogiikan PROFIsafe-väylällä. Estää moottoria ylittämästä säädettyä rajaa, jota voidaan säätää turvaväylän kautta kesken ajon.
- *Safe maximum speed* (SMS) suojaa liian suurelta nopeudelta/taajuudelta. Ylä- ja alaraja voidaan konfiguroida erikseen, mikäli raja ylitetään, käynnistetään SSE-funktio.
- *Prevention of unexpected start-up* (POUS) estää koneiston vahingossa käynnistymisen. Aktivoi taajuusmuuttajan STO-funktion. (FSO-11 User´s manual: 14.)

Moduulissa on useita turva-I/O-liittimiä, joita voidaan käyttää ulkoisten turvalaitteiden liittämiseksi osaksi turva-automaatiojärjestelmää. Liittimillä voidaan monitoroida ja ohjata esimerkiksi painonappeja, hätä-seis-kytkimiä, turvaportteja ja valoverhoja. (FSO-11 User´s manual: 24.)

Kuva 11. FSO-11-turvamoduuli.

FSO-11-turvamoduulin musta liitin on 24V:n syöttö ja keltainen liitin kytketään taajuusmuuttajan XSTO-liittimeen. Musta liitin kyljessä toimii dataliittimenä ja kytketään taajuusmuuttajaan. Moduulin alalaidassa on I/O-liittimiä joilla voidaan toteuttaa redundanttisia tuloja ja lähtöjä. Siinä on 5 LED-merkkivaloa jotka indikoivat laitteen tilaa ja 20 merkkivaloa indikoi I/O-tiloja. Tilaa indikoivien valojen alla on maadoitusruuvi ja I/O-liittimien vieressä on tehdasasetuksiin palauttava reset-nappi. (FSO-11 User´s manual: 25.)

Turvalogiikan ohjatessa taajuusmuuttajaa, täytyy kenttäväyläliikenteen luotettavuus turvata PROFIsafe-turvaväylällä. PROFIsafe-protokolla sisältää monia ominaisuuksia haittojen minimoimiseen virheistä, jotka ilmenevät viesteissä niiden kulkiessa laajoissa ja monimutkaisissa verkoissa. Protokolla luo turvatun yhteyden viesteille kenttälaitteelta logiikalle, mikä tässä työssä tarkoittaa turvalogiikalta FSO-moduulille. Logiikalta tulevat viestit kulkevat FENA-moduulin kautta FSO-moduulille joka prosessoi viestit, suorittaa käsketyt tehtävät ja lähettää PROFIsafe viestin takaisin turvalogiikalle. (FSO-11 User´s manual: 69 – 72.)

FSO-11-moduuli käyttää lyhyttä, 128 bittistä PROFIsafe viestiä. Viesti koostuu kuudestatoista oktetista, joista kaksitoista on varattu tiedonsiirrolle, yksi status- sekä control-bitille ja kolme CRC-viestille. CRC, eli cyclic redundancy check, on osa jokaista turvaväylässä lähetettävää viestiä, tehtävänään valvoa viestin eheyttä.

7 Työn toteutus

Työ toteutettiin kahdelle jo olemassa olevalle, koulutuskäyttöön tarkoitetulle, ACS880 demoräkille [Liite 1]. Yksi räkki koostuu kahdesta ACS880-taajuusmuuttajasta ja kahdesta moottorista. Räkit muokattiin kestotestiä varten irrottamalla ylimääräinen laitteisto ja asentamalla tarvittavat moduulit. Koska kestotesti toteutetaan palovaarallisessa tilassa, ei kestotestiä voida suorittaa ympärivuorokautisesti, vaan se joudutaan käynnistämään aamulla töihin saavuttaessa ja sammuttamaan lähdettäessä. Räkeillä voidaan kuitenkin rakentaa toimiva kestotestiympäristö, mikä on helposti siirrettävissä ja muokattavissa.

Kestotesti on myöhemmin tarkoitus siirtää ABB:n moottoritehtaan kellarissa sijaitsevaan kestotestikäytävään, jotta testiä voi suorittaa vuorokauden ympäri tauotta. Kellariin ei valitettavasti pystytty asentamaan työn aikataulun puitteissa, joten päätimme toteuttaa insinöörityön räkkeihin ja myöhemmin, siirtää testi kokonaisuudessaan kestotestialueelle yhteistyössä High Power Drives-osaston kanssa. Kestotestikäytävä on suljettu ja suojattu paloturvallisuusjärjestelmällä, tarkoittaen että mahdollisessa vaaratilanteessa sähköt katkaistaan, eikä palo pääse leviämään kellarin ulkopuolelle. Kestotestikellariin myöhemmässä vaiheessa asennettava testikokonaisuus ei tule eroamaan räkkitoteutuksesta mitenkään muuten, kuin että testiä voidaan suorittaa ympärivuorokautisesti ja näin ollen kerätä enemmän dataa.

Räkit tarjoavat kätevän alustan kestotestiprojektille. Ne mahdollistivat myös työn pikaisen suunnittelun ja aloittamisen, koska ne löytyivät käyttämättöminä varastosta ja näin ollen pystyimme ne varaamaan omaan käyttöön. Räkeissä oli ohjausjännitteen kytkentää varten tuotu 24 VDC DIN-kiskoille asennetuille riviliittimille taajuusmuuttajien vasemmalle puolelle, mikä palveli projektia vallan mainiosti. Ohjausjännitteet tuotiin riviliittimiltä taajuusmuuttajan ohjauskortille, FSO-moduulille sekä logiikalle. DIN-kiskoa löytyi myös taajuusmuuttajien alapuolelta, johon logiikan liitinalusta saatiin asennettua.

Kestotestiräkeille asennettiin myös kenttäväylän yhteyden lisätestausta varten Hirchmann-kytkin, jonka kautta PROFINET-väylä kulkee. Logiikalta kytketään Ethernet-johto kytkimen ensimmäiseen porttiin ja kytkimen toisesta portista ensimmäisen taajuusmuuttajan FENA-moduulille. Kytkimen 24 V-ohjausjännite ohjattiin logiikkaan lisätyn rele-IO-moduulin kautta kulkemaan siten, että halutessa voidaan katkaista kytkimeltä syöttö. Näin voidaan simuloida kenttäväyläyhteyden katkeamista ja testata yhteyden

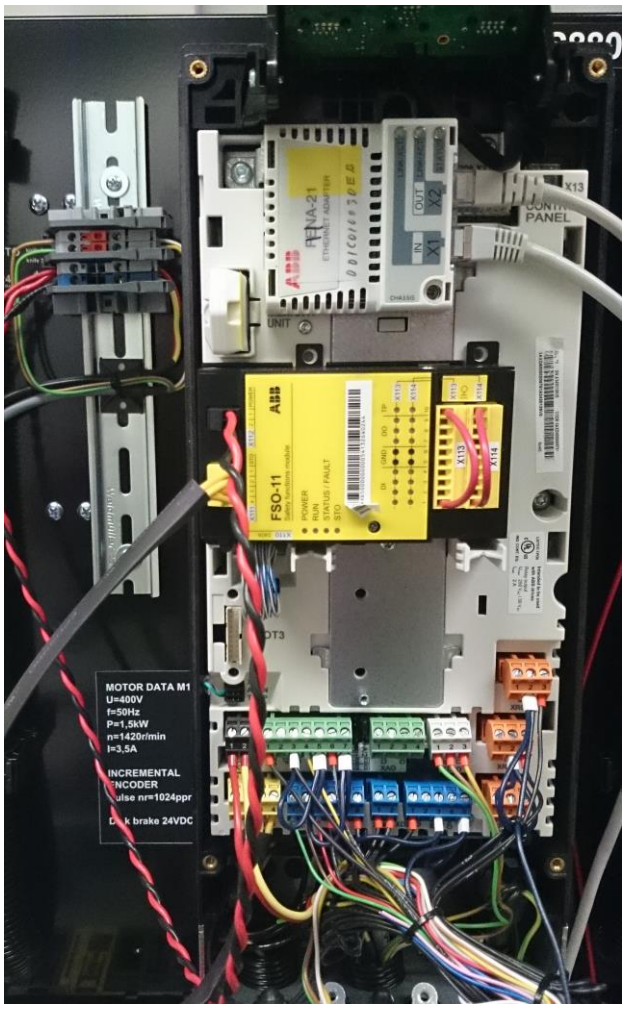
palaamisen jälkeen tapahtuvaa uudelleenyhdistämistä sekä mahdollisia vikatilanteita yhteyden katkeamiseen liittyen.

7.1 Asennukset

Asennustyötä edelsivät työpöydällä tehtävät alustavat toimenpiteet. Komponentit tilattiin uusina tehtaalta, joten kaikki päivitettiin uusimpiin ohjelmistoihin. Muistitikut, FENA-21- ja FSO-11-moduulit, ohjauskortit, paneelit ja logiikkaohjelmisto pitivät kaikki päivittää. Projektipohja luotiin neljälle taajuusmuuttajalle ja kaikille moduuleille sekä muistitikuille osoitettiin oikeat nimet ja osoitteet. Taajuusmuuttajiin asennettavat laitteet numeroitiin yhdestätoista neljääntoista, jotta projektissa luotu järjestys säilyisi asennuksessa.

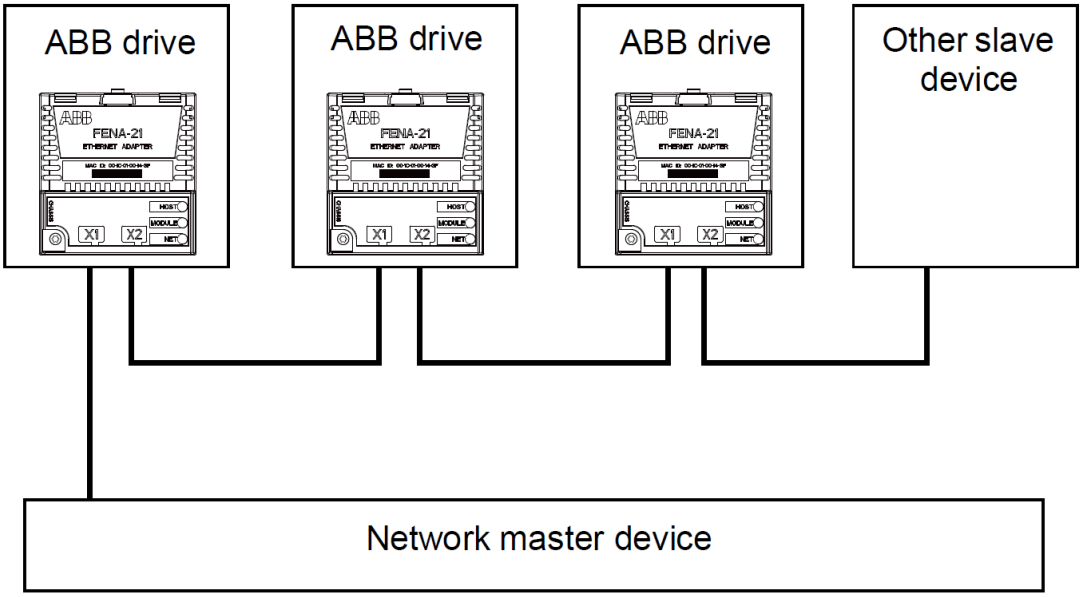
Insinöörityö toteutettiin kahdelle ACS880-demoräkille. Asennustyö aloitettiin vaihtamalla vanhoihin koulutuskäytössä olleisiin räkkeihin uudet ZCU-12-ohjauskortit. ZCU-12 on ZCU-11-ohjauskortin uudempi malli, joka mahdollistaa 160 kB:n CODESYS-muistin, taajuusmuuttajien linkittämisen ja sulautetun Modbus-liitännän. Uusi ohjauskortti tukee myös uutta suuremmalla muistilla varustettua ZMU-02-muistikorttia. (ACS880 firmware update: 3.) Ohjauskorttien vaihdon yhteydessä, jätettiin osa vanhoissa korteissa olleista enkoodereista ja kytkennöistä asentamatta, ainoastaan välttämättömät laitteistot asennettiin uusiin kortteihin. Ylimääräiset osat jätettiin räkkeihin roikkumaan, koska räkit palautetaan myöhemmin alkuperäiseen tarkoitukseensa, kun kestotesti saadaan asennettua lopulliseen sijoituspaikkaansa kestotestikäytävälle. Ohjausjännite korteille saadaan viereisistä riviliittimistä.

Kaikkien neljän testissä olleen taajuusmuuttajan uudelle ohjauskortille asennettiin ensimmäiselle liitinpaikalle FENA-21-Ethernet-kenttäväylämoduuli ja toiselle liitinpaikalle FSO-11-turvamoduuli (kuva 12). Uudet ZMU-02-muistikortit asetettiin ohjauskorttien vasemmassa ylälaidassa oleviin muistikorttipaikkoihin. FSO-11-turvamoduuli liitettiin taajuusmuuttajan XSTO-liittimeen ja datakaapeli taajuusmuuttajassa olevaan X12 Safety Option-dataliittimeen. Taajuusmuuttajien vieressä olevista riviliittimistä saadaan FSO-moduulin ohjausjännite. Turvamoduulin I/O-liittimissä oli tehtaan jäljiltä hyppylanka test pulse- ja DI1-liittimien välillä, joka sai jäädä, mutta jolla ei testissä sinänsä ole merkitystä.



Kuva 12. Esimerkki yhden taajuusmuuttajan laitteistosta kestotestissä.

Projektin taajuusmuuttajat liitettiin PROFINET-väylään FENA-21-kenttäväylämoduulilla. Logiikan CM579-PROFINET-moduuli toimii järjestelmän masterina, johon liitettiin neljä taajuusmuuttajaa sarjaan (kuva 13). Moduulit asennettiin taajuusmuuttajiin valmiiksi numeroidussa järjestyksessä sarjaan siten, että ensimmäisen moduulin X1-liittimelle saapuu viesti logiikalta ja X2-liittimeltä lähetetään viesti seuraavalle FENA-moduulille. Sama toistuu muilla taajuusmuuttajilla sillä poikkeuksella, että X1-liittimelle saapuu viesti edelliseltä moduulilta. Kytkennät on toteutettu suojaamattomalla Cat 5e-Ethernet-kaapelilla.



Kuva 13. FENA-21-moduulin sarjaankytkentä. (FENA-11 User´s Manual: 34)

Toiseen räkeistä asennettiin logiikka räkissä jo olemassa olleeseen DIN-kiskoon, turvakytkimen vasemmalle puolelle. Logiikan liitinalusta kiinnitettiin kiskoon ja siihen liitettiin PM583-ETH-prosessori, SM560-S-turvalogiikka ja CM579-PNIO-PROFINET-moduuli. Ohjausjännite logiikalle saatiin taajuusmuuttajan vasemmalla puolella olevista riviliittimistä. Yhteys logiikkaan muodostetaan Ethernet-johdolla, joka kytketään logiikassa sille varattuun PC-liityntään tarkoitettuun RJ-45-liittimeen. Tietokoneen päässä käytetään ASIX USB 3.0 to Gigabit Ethernet -adapteria. Kenttäväylä kytketään PROFINET-moduulin PNIO1-portista ensimmäiselle taajuusmuuttajan FENA-moduulille.

Kenttäväylän katkeamisen simulointia varten rakennettiin rele-kytkin-yhdistelmä, joka koostuu Hirchmann-merkkisestä kytkimestä ja logiikkaan liitettävästä DX522-rele-I/O-moduulista. PROFINET-väylä kytkettiin logiikalta kytkimen ensimmäiseen porttiin ja jatkettiin kytkimen toisesta portista ensimmäiselle taajuusmuuttajalle. Kytkimen ohjausjännite toteutettiin siten, että nollajohdin kytkettiin suoraan riviliittimeltä, kun taas vaihe ohjattiin relemoduulin COM- ja NC-liittimen kautta kytkimelle. Näin saatiin logiikkaohjaus kytkimen jännitteelle, mikä tarkoittaa että kytkin voidaan sammuttaa ja väylä katkaista, näin halutessa logiikan ohjelmassa.

Työn sydän, FSO-11-turvamoduulit, joita varten kestotesti kehitettiin, liitettiin taajuusmuuttajien liityntäpaikalle kaksi. Myös turvamoduulit olivat valmiiksi numeroitu projektin

järjestykseen ja asennettiin maadoitusruuvin avulla taajuusmuuttajien ohjauskorteille numerojärjestyksessä. Tiedonkulku taajuusmuuttajan ja turvamoduulin välille toteutettiin lyhyellä datakaapelilla, joka kiinnitettiin moduulin vasemmasta reunasta taajuusmuuttajan ohjauskortille, toisen kiinnityspaikan alta löytyvään Safety Option-liittimeen. Tämän yhteyden kautta turvamoduuli keskustelee taajuusmuuttajan ja väylän kautta logiikan kanssa. Moduulin ohjausjännite tuotiin niin ikään räkkeihin valmiiksi asennetuilta ohjausjänniteriviliittimiltä kierretyllä puna-musta-parikaapelilla.

Turvafunktioiden turvatilojen täytäntöönpano toteutuu STO-käskyllä, joka saadaan turvamoduulilta taajuusmuuttajalle keltaisella STO-kaapelilla. Se koostuu neljästä, mustalla litteällä kumisuojuksella päällystetystä johtimesta, jonka päissä on keltaiset liittimet. Kaapeli liitettiin turvamoduulissa sille tarkoitetulle paikalle, virtaliittimen viereen ja taajuusmuuttajan ohjauskortissa XSTO-paikalle. FSO-moduulin turvafunktion toteutuessa, tai sen sisäisen diagnostiikan pettäessä, voi STO-yhteyden katkaisemisella laukaista taajuusmuuttajan sisäisen turvatoiminnon, joka ajaa sen seis-tilaan ja estää sitä käynnistymästä ennen kuin yhteys on palannut ja virhe kuitattu.

Haastetta työhön toi tuotekehitysympäristössä työskentely. Koska uusia versioita laitteistoihin valmistui jatkuvasti, piti ne päivittää myös kestotestialustan laitteisiin. Kun kestotestiä alettiin kehittää, eivät kaikki versiot vielä tukeneet tarvittavia ominaisuuksia. Esimerkiksi vasta taajuusmuuttajan ohjauskortin versio numero 1.70.2.0, joka valmistui kesken projektin, korjasi kriittisiä parametreja PROFIsafe-kommunikointiin.

Työssä käytettyjen laitteiden ja ohjelmistojen versiot ovat:

- ACS880; ohjauskortti ZCU-12, AINF6 v1.70.2.0
- Ohjauspaneeli; GPAPI v4.50
- PS501 Control Builder Plus; v2.3.0 Release Build 339
- Automation Builder; v1.0.1
- AC500-S; PM583 v2.3.3
- GSDML; ABB FENA v3.0
- FSO-11; FW 2.50 rev. Q.

Laitteistojen ohjelmaversiot tulevat elämään paljonkin kestotestin kehittyessä. Yllä luetellut versionumerot ovat insinöörityöhön valmistetun kestotestipohjan viimeisimmät ja kestotesti on todettu niillä toimivaksi.

7.2 Logiikkaohjelmointi

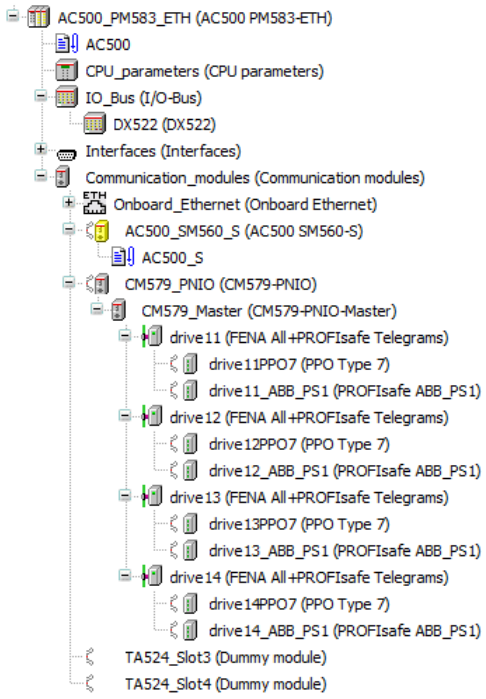
Kestotestin ohjelman tarkoituksena on turvalogiikkaohjauksella kytkeä vuorotellen FSO-moduulin turvafunktioita aktiiviseksi ja pois, samalla tarkkaillen järjestelmän vakautta ja eheyttä. Ohjelma rakennetaan tilakoneeksi, jossa seuraavaan tilaan siirtymistä ohjataan ajastimen avulla. Jokaisessa tilassa kytketään yksi turvafunktio aktiiviseksi ajastimen laskiessa aikaa seuraavaan tilaan siirtymistä varten. Joissakin tapauksissa turvafunktioita aktivoidaan useampi samaan aikaan, jotta väylälle ja siihen kytkettyihin laitteistoihin saadaan mahdollisimman suuri kuorma aikaiseksi.

Ohjelmassa on syklinen tiedonsiirto tavallisen ja turvalogiikan välillä. Koska turvaohjelma on täysin itsenäinen sekä tavallisesta ohjelmasta riippumaton ja koska taajuusmuuttajien ohjaukseen sekä tiedon tallentamiseen tarvitaan tietoa turvaohjelman toiminnasta, tarvitaan erillinen sykliseen tiedonsiirtoon tarkoitettu ominaisuus. Syklisessä tiedonsiirrossa jokaisen turvaohjelmassa suoritetun ohjelmakierron lopussa kirjoitetaan taulukko, joka lähetetään tavallisen ohjelman puolelle pakettina.

Control Builder-ohjelmassa rakennettiin projekti, johon lisättiin tässä työssä käytetyt laitteistot. Projektin kokoaminen aloitettiin valitsemalla PM583-prosessori annetusta listasta. Seuraavaksi liitettiin logiikan liitinlevyn mukaisessa järjestyksessä komponentit ohjelmassa löytyville avoimille liitinpaikoille. Ensimmäiselle liitinpaikalle sijoitettiin SM560-S-turvalogiikka. Koska turvaohjelma on salasanasuojattu, pyytää ohjelma salasanaa turvalogiikan liittämisen yhteydessä. Toiselle vapaalle paikalle liitettiin logiikan kenttäväylämoduuli, jolloin ohjelma luo moduulin alle automaattisesti myös PROFINET-masterin väyläkommunikointia varten.

Masterin alle kootaan väylässä olevien orjien kommunikointia varten tarvittavat komponentit (kuva 14). Tässä tapauksessa piti FENA- sekä PROFIsafe-kommunikointia varten asentaa XML-tiedostoformaatissa oleva GSDML-tiedosto, joka loi ohjelmaan kirjaston tarvittavista tiedoista ja mahdollisti logiikan kommunikoinnin väylään. Kun kirjasto oli luotu, lisättiin PROFINET- sekä turvalogiikkakommunikointiin tarkoitettu, väy-

lässä orjana toimiva kommunikointiväylä. Väylään lisättiin syklisen väyläliikenteen viestien muodon ja suuruuden määrittävä lohko, tässä tapauksessa suurin mahdollinen, PPO7. PROFIsafe-kommunikointiin lisättiin erillinen viestilohko.



Kuva 14. Control Builder-ohjelmassa rakennettu projektipuu.

Jokaisen orjan PPO- ja PS1-lohkoihin kartoitettiin projektissa tarvittavat muuttujat, joita logiikkaohjelmassa kutsutaan. PPO-lohkoon kartoitettiin muuttujat sekä niiden osoitteet omaan sarakkeeseensa. Näillä muuttujilla voidaan logiikkaohjelmoinnissa lukea taajuusmuuttajan arvoja ja ohjata taajuusmuuttajille haluttuja ohjearvoja. PS1-lohkoon parametroitiin turvaväylälle käytetty SIL-taso, lähde- ja pääteosoite, watchdog-aika ja väylän CRC-diagnostiikka-aika. Muuttujien kartoitukset tehdään kahdelle eri välilehdelle. Turvaohjelmointiin kartoitettiin oma välilehtensä, jossa kartoitetut muuttujat ovat käytettävissä ainoastaan turvaohjelmointiin. Toiselle välilehdelle kartoitettiin muuttujat osoitteineen FSO-kommunikointia varten, joita käytettiin tavallisen ohjelman puolella.

Ohjelmalle laskettiin watchdog aika siten, että lasketaan FSO-moduulin viiveaika, FENA:n ja PROFINET-väylän viiveaika sekä turvalogiikan viiveaika yhteen. Koska FENA-moduuli ja PROFINET-väylän viiveajat vaikuttavat molempiin suuntiin, otetaan se huomioon kaksi kertaa. Watchdog-aikaa laskettaessa pitää ottaa huomioon, että laskettuun aikaan lisätään 30 prosenttia, jotta turhilta virheiltä vältytään.

Teoriassa turvaväylän watchdog-aika lasketaan seuraavasti: 1.3 x (50 ms + (3 ms + 4 ms) + 6 ms + (4 ms + 3 ms)) = 91 ms, jossa 50 ms on FSO-moduulin viiveaika, 3 ms on FENA-moduulin viiveaika, 4 ms on PROFINET-väylän viiveaika ja 6 ms on turvalogiikan viiveaika. Ensimmäisen vaiheen kestotesteissä käytetään kuitenkin 200 ms watchdog-aikaa, jota kestotestin kehityksessä aletaan portaittain alentaa todellisen suorituskyvyn mittaamiseksi.

Syklisen tiedonsiirron mahdollistamiseksi täytyy projektipuusta valita keltainen turvalogiikan, syklisen tiedonsiirron välilehti. Välilehdellä määritellään syklisen tiedonsiirron määrä, joka tässä tapauksessa asetettiin 160 tavun suuruiseksi. Määritellyn tiedonsiirron määrän pitää olla identtinen ohjelmassa siirrettävän tietomäärän kanssa. Ohjelmassa siirrettävä tieto järjestetään 80 sanan taulukkoon. Tulevaa testaamista varten, väylän katkaisemiseen asennettu kytkin, lisättiin projektipuun I/O-väylään. DX522-moduulille kartoitetaan muuttuja relelähtönä, jota tullaan ohjaamaan logiikkaohjelmasta. Kaksi viimeistä liitinpaikkaa jätettiin projektissa tyhjäksi.

7.2.1 Tavallinen ohjelma

Ohjelmointi aloitettiin avaamalla lähdekooditiedosto projektipuussa sijaitsevan keskusyksikön alapuolelta, mikä aukaisee CoDeSys-ohjelmointityökalun. Projektiin lisättiin tarvittavat kirjastot. Ohjelma koostuu pääohjelmasta ja pienemmistä kutsuttavista lohkoista. Ohjelma löytyy kokonaisuudessaan liitteenä insinöörityön lopusta.

Tavallisen ohjelman alussa luetaan logiikan prosessorilta aika, joka kirjoitetaan muuttujaan ”SysTim” (kuva 15). Aika luetaan mikrosekunneissa ja tallennetaan 64-bittisenä. 64-bittinen arvo vaatii kaksi DWORD-sanaa, joista ensimmäinen on ala-arvo ja toinen ylä-arvo. Ala-arvon pyörähdettyä ympäri, kasvaa ylä-arvo yhdellä. Tätä muuttujaa käytetään ohjelmassa myöhemmin väliaikatietojen tallennukseen, vaikka prosessissa ei olisi virhettä.

```
0003 (* System Time in high and low DWORD (64 bit) *)
0004 SystemTimer(SystemTime := SysTim);
```

Kuva 15. Prosessorilta luettu aika tallennetaan 64-bittiseksi muuttujaksi.

Toisena ohjelmassa luetaan syklisen tiedonsiirron taulukosta arvoja, jotka tallennetaan omiksi muuttujiksi tavallisen ohjelman puolelle (kuva 16). Muuttujia käytetään ohjelmassa taajuusmuuttajien ohjaukseen ja tietojen tallennukseen virhetilanteissa. Ohjelmassa voisi lukea suoraan muistipaikoilta arvoja myöhemminkin, mutta omiksi muuttujiksi tallennus helpottaa ohjelman seuraamista ja selventää kokonaisuutta. Kirjoitus myös kuormittaa logiikkaa, mikä on erittäin toivottavaa kestotestiympäristössä.

```
0006 (* Cyclic data exchange *)
0007 StateCheck := I_awCyclicNoneSafe[1];
0008 SLS2req := I_awCyclicNoneSafe[2];
0009 SLS1req := I_awCyclicNoneSafe[3];
0010 SS1req := I_awCyclicNoneSafe[4];
0011 SSEreq := I_awCyclicNoneSafe[5];
0012 POUSreq := I_awCyclicNoneSafe[6];
```

Kuva 16. Syklisen tiedonsiirron tallennus muuttujiin.

Taajuusmuuttajaa ohjataan JOS-lausekkeella, joka on riippuvainen taajuusmuuttujalta luettavista virheistä. Kun turvamoduuli kytkee taajuusmuuttajan turvatilaan ajavan turvafunktion, tai kun taajuusmuuttajan STO-funktio on aktiivinen, lähetetään taajuusmuuttajalle ohjaus-sana 1024, joka pysäyttää käynnin ja siirtää taajuusmuuttajan seistilaan. Kuittauksen jälkeen ja virheen poistuessa, aloitetaan ”startTimer”-laskuri ja taajuusmuuttajalle lähetetään 1150-ohjaus-sana, joka on käyntiin valmistava käsky. Laskurin päästyä asetetun rajan yli, lähetetään käyntikäsky taajuusmuuttajalle, joka käynnistää moottorin (kuva 17). Laskuria tarvitaan lausekkeessa koska ohjelmakierto on liian nopea, eikä taajuusmuuttaja ehdi rekisteröidä eri käskyjä jos ne tulevat ilman viivettä. Taajuusmuuttajan käynnistys vaatii aina ohjaus-sanan kierrätystä käyntiin valmistavan 1150-käskyn kautta 1151-käyntikäskyyn.

```
0083 (* Drive start command word *)
0084 IF (drive11STOact = 1 OR SS1req = 1 OR POUSreq = 1) THEN
0085     drive11Command := 1024;
0086     startTimer := T#0s;
0087 ELSIF (drive11STOact = 0 AND startTimer > T#500ms) THEN
0088     drive11Command := 1151;
0089 ELSE
0090     drive11Command := 1150;
0091     startTimer := startTimer + T#3ms;
0092 END_IF;
```

Kuva 17. Taajuusmuuttajan ohjaus-sanaa ohjaava JOS-lauseke.

JOS-lausekkeessa ohjataan ainoastaan ensimmäistä taajuusmuuttajaa. Ohjaus-sanat muille taajuusmuuttajille saadaan yksinkertaisella kopiointimenetelmällä, jolla ensimmäisen taajuusmuuttajan ohjaussanan arvo kirjoitetaan kolmen muun ohjaussanojen muuttujiin. Ohjauksessa luetaan ainoastaan ensimmäisen taajuusmuuttajan tila-arvoja, jolloin mikäli jossakin kolmessa muussa taajuusmuuttujassa tapahtuu arvaamaton turvatilaan siirtyminen, jää se odottamaan ohjelmakierron alusta aloittamista. Virhetilassa olevan taajuusmuuttujan tiedot kuitenkin tallennetaan myöhempää analysointia varten.

Mahdollisimman suuren kuormituksen takaamiseksi päädyttiin jatkuvaan referenssiarvon kierrättämiseen. Luotiin muuttuja, jonka tehtävä on muuttaa arvoaan jatkuvasti, joka puolestaan syötetään taajuusmuuttajien viestikehysten jokaiselle paikalle. Näin ollen taajuusmuuttajat saavat jatkuvasti uuden arvon ja joutuvat prosessoimaan suuren määrän dataa koko viestikehyksen laajuudelta. PPO7-viestilohkossa on kaksitoista kanavaa sekä sisään että ulos, jokainen kanava on kooltaan yksi sana.

Referenssiarvoja kierrättävä muuttuja tehtiin laskurityyppisesti, jonka arvoa kasvatettiin yhdellä jokaisella ohjelmakierrolla (kuva 18). Muuttujan ylä-arvoksi asetettiin 20000, jolloin laskuri kääntää muuttujaa kasvattavan arvon negatiiviseksi, näin ollen muuttujan arvo alkaa laskea. Muuttujan saavuttaessa nolla-pisteen, kääntyy arvo jälleen positiiviseksi kasvattaen muuttujaa. Alati kasvava ja laskeva muuttujan arvo kopioidaan kaikkien neljän taajuusmuuttajan yhteentoista referenssiarvoon. Ensimmäinen kanava viestikehyksestä on varattu ohjaus-sanalle.

```
0099 (* Cycle drive reference values *)
0100 cyclicTempValue := cyclicTempValue+cyclicDelta;
0101
0102 IF cyclicTempValue > 20000 THEN
0103     cyclicTempValue := 20000;
0104     cyclicDelta := -cyclicDelta;
0105 ELSIF cyclicTempValue < cyclicDelta THEN
0106     cyclicDelta := -cyclicDelta;
0107     cyclicTempValue := ABS(cyclicDelta);
0108 END_IF;
```

Kuva 18. Taajuusmuuttajille referenssiarvoja kierrättävä laskuri.

PROFIsafe-tilasanasta löytyvälle vasteajalle tehtiin takautuva muisti, jossa tallennettiin viimeisen viiden sekunnin korkein arvo, edellisen viiden sekunnin korkein arvo sekä koko ohjelman käynnissäoloajan korkein arvo. Tallennukseen luotiin laskuri, joka laskee aikaa nollasta ylöspäin. Laskurin laskiessa aikaa, tallennetaan suurin PROFIsafe-

vasteaika muuttujaan, joka mittaa korkeinta vasteaikaa viimeisen viiden sekunnin ajalta. Laskurin päästessä viiden sekunnin rajan yli, tallennetaan viimeisen viiden sekunnin korkein arvo edellisen viiden sekunnin arvoon. Samalla viimeisin arvo nollataan ja laskuri aloittaa alusta.

Takautuvasti tallennettavaa, vanhempaa viiden sekunnin arvoa verrataan koko ohjelmakierron korkeimpaan aikaan ja sen ylittäessään, tallennetaan vanha arvo uudeksi kaikkien aikojen korkeimmaksi ajaksi (kuva 19). Jokaisen taajuusmuuttajan vasteajat tallennetaan omiksi muuttujiksi. Kaikki kolme vasteajan historiatietoja sisältävää muuttujaa tallennetaan taajuusmuuttajien virhetilanteessa kirjoitettavaan taulukkoon.

```
0162 (* Response time recording *)
0163 IF drive11PROFIsafeStatusResTim > drive11NewHighValue THEN
0164 drive11NewHighValue := drive11PROFIsafeStatusResTim;
0165 END_IF;
```

Kuva 19. PROFIsafe-vasteaikojen seuranta.

Virhetilanteessa taajuusmuuttajan ja turvaväylän tiedot kirjoitetaan taulukkoon myöhempää muistikortille tallennusta varten. Taulukkoon kirjoittaminen aloitetaan JOS-lausekkeella, mikäli taajuusmuuttajan tilasanan kolmas bitti, joka indikoi virhettä taajuusmuuttajassa, muuttuu todeksi (kuva 20). Lausekkeen ehdoksi tarvitaan myös vanhan virheen tunnistava muuttuja, jotta kirjoitusta ei tehdä jokaisella ohjelmakierrolla. Jokaiselle taajuusmuuttujalle luodaan taulukko globaaleihin muuttujiin, johon tiedot tallennetaan. Taulukkoon kirjoitettava sana saa osoitteensa muuttujasta, jota kasvatetaan yhdellä jokaisella rivillä. Taulukon kirjoittamisen jälkeen muuttujaan lisätään arvo, jotta seuraavan virheen sattuessa taulukkokirjoittaminen osataan aloittaa seuraavasta vapaasta kohdasta.

```
0206 (* In case of fault, write data to table *)
0207 IF drive11Status.3 AND oldError11 = FALSE THEN
0208     Drive11_write_table[d11] := drive11FaultNum;
0209     Drive11_write_table[d11 + 1] := VersionNumber;
0210     Drive11_write_table[d11 + 2] := LoopNum1;
0211     Drive11_write_table[d11 + 3] := LoopNum2;
0212     Drive11_write_table[d11 + 4] := drive11Status;
0213     Drive11_write_table[d11 + 5] := drive11PSStatus;
```

Kuva 20. Vian sattuessa kirjoitetaan taajuusmuuttujan tiedot taulukkoon.

Ensimmäinen tallennettava tieto on virheen numero, jota kasvatetaan jokaisen virheen jälkeen. Toisena tallennetaan versionumero, jotta virheiden historiassa voidaan jäljittää missä logiikan versiossa virhe on tapahtunut. Muita tallennettavia tietoja ovat turvaohjelman tilakoneen kierrosnumero, taajuusmuuttajan tilasana, PROFIsafe tila- ja ohjaussanat, turvaväylän tilatiedot, vasteaikojen historiamuuttujat ja logiikan prosessorin aika. Taulukkoon kirjoitettavia tietoja virheen sattuessa on ensimmäisessä kestotestiversiossa yhteensä 27. Kirjoittamisen jälkeen lisätään virhenumeroon yksi ja taulukon järjestysnumeroon 28.

Virheen kuittaus taajuusmuuttajassa tapahtuu ohjaussanan seitsemännellä bitillä. Mikäli taajuusmuuttajan tilasanan kolmas bitti on aktiivinen, kuitataan virhe taajuusmuuttajassa lähettämällä ohjaus-sana, jossa seitsemäs bitti on tosi (kuva 21). Jokaiselle taajuusmuuttajalle luotiin oma JOS-lauseke, jossa kyseisen laitteen virhe kuitataan. Virheen kuittauksen jälkeen taajuusmuuttaja jää odottamaan turvaohjelman tilakoneen alusta-aloittamista.

```
0373 (* Reset fault*)
0374 IF drive11Status.3 AND oldError11 THEN
0375 drive11Command.7 := TRUE;
0376 END_IF;
```

Kuva 21. Taajuusmuuttajan virheen kuittaus vikatilanteessa.

Virhetilanteissa taulukoonkirjoittamisen lisäksi tarvitaan väliaikatietoja kestotestin tilasta. Arvioitiin, että väliaikatietojen tallennus noin kerran tunnissa on riittävä prosessin toiminnan valvomiseksi. Luotiin lauseke, joka vertaa arvoa logiikan prosessorin aikaan ja arvon ylittäessä ajan, kirjoitetaan väliaikatiedot omaan taulukkoonsa. Prosessorin ajan ylittäessä muuttujan arvon lisätään muuttujaan tuntia vastaava luku, mikä jää odottamaan prosessorin ajan ylitystä. Prosessorin ajan ollessa kahdessa dword-sanassa, joudutaan alemman arvon ylitystä valvomaan ja sen ylittyessä lisätäämään ylempään arvoon yksi, jolloin alempi arvo alkaa alusta. Laskuri muuttaa kirjoituksen sallivan muuttujan todeksi, jolloin kirjoitus alkaa (kuva 22).

Väliaikatietojen taulukko on kooltaan virhetaulukoita suurempi. Siihen tallennetaan 86 arvoa. Tallennettavat arvot ovat samoja, joita virhetaulukoihin tallennetaan, mutta tässä tapauksessa kaikkien taajuusmuuttajien arvot tallennetaan samaan taulukkoon. Taulukon kirjoittamisen jälkeen, kirjoituksen salliva muuttuja käännetään epätodeksi, jolloin

se jää odottamaan laskurin täyttymistä. Myös tässä tapauksessa taulukon järjestysnumeron arvoa kasvatetaan, jotta seuraava kirjoitus alkaa oikeasta kohdasta.

```
0390 (* Trigger for "All OK" table *)
0391 SysTimLog := systim.ulLow;
0392
0393 IF SysTimLog > SysTimTrigger THEN
0394     IF SysTimTrigger + 3600000000 > 4294967295 THEN
0395     SysTimTrigger := 3600000000;
0396     SysTimTriggerHigh := SysTimTriggerHigh +1;
0397     ELSE
0398     SysTimTrigger := SysTimTrigger + 3600000000;
0399     END_IF;
0400 WriteAllOK := TRUE;
0401 END_IF;
```

Kuva 22. Väliaikatietojen tallennukseen tehty ajastin.

Virhetaulukoiden ja väliaikataulukon täyttyessä, täytyy ne kirjoittaa muistiin myöhempää analysointia varten. Taulukoihin mahtuu kolme kirjoitusta, jonka jälkeen se kokonaisuudessaan kirjoitetaan muistikortille. Virhetaulukoissa järjestysnumeron muuttuessa 85:en, toimii se ehtona kirjoitukselle, joka kutsuu kirjoituslohkoa. Väliaikataulukon ollessa hieman suurempi, tapahtuu sen kirjoitus vasta järjestysnumeron noustessa 349:an. Jokaisen taulukon SD-muistikortille kirjoittamiseen on oma lohko, jota kutsutaan ehtojen täyttyessä (kuva 23).

Taulukon muistiin kirjoituksessa kutsutaan kirjoituslohkoa, jonka kirjoituksen aloittava muuttuja muutetaan todeksi. Lohkolta saatu tieto kirjoituksen onnistumisesta, palauttaa taulukkokirjoitukseen tarkoitetun järjestysmuuttujan oletusarvoon, jolloin taulukon kirjoittaminen alkaa jälleen alusta. Lohkon tilatietojen hyväksikäyttö vaatii sen, että lohkolta saadut arvot tallennetaan pääohjelmakierrossa omiksi muuttujikseen ennen SD-kortille kirjoitusta.

```
0507 (* WRITE TABLES TO SD CARD *)
0508 IF d11 = 85 THEN
0509 Write_d11(
0510     oldFailStatus:= ,
0511     startCommand:= ,
0512     sDevName:= ,
0513     SD_Write11:= TRUE,
0514     WriteDone=> );
0515     IF WriteDone11 OR  WriteError11 THEN
0516     d11 := 1;
0517     END_IF;
0518 END_IF;
```

Kuva 23. Ensimmäisen taajuusmuuttajan virhetaulukon tallennus SD-muistikortille.

Ehtojen täyttyessä ja kirjoituksen alkaessa kutsutaan lohkoa, johon on luotu CoDeSys-kirjastosta poimittu valmis SD-kortille kirjoituslohko. Lohkoon on määritelty luotavan tiedoston nimi, sen koko ja aloitusosoite. Vastaavasti lohkosta saadaan tieto siitä, onko kirjoitus valmis tai virheellinen. Mikäli lohkossa tapahtuu virhe, kertoo se virhenumeron muodossa mistä syystä kirjoitus on epäonnistunut. Lohkon käynnistävä muuttuja on listattu tulomuuttujiin ja lohkosta saatavien tilatietojen muuttujat lähteviin muuttujiin.

7.2.2 Turva-automaatio-ohjelma

Turvaohjelman ohjelmointi aloitettiin avaamalla projektipuussa, keltaisen turvalogiikan alla oleva turvaohjelman lähdekooditiedosto. Tämä avaa CoDeSys-ohjelmointityökalun turvaohjelmalle tarkoitetulla keltaisella taustalla. Ohjelmointi ei eroa tavallisesta ohjelmoinnista muuta kuin kartoitettujen muuttujien osalta. Turvaohjelmassa on myös valmiiksi kirjastoja PROFIsafe-väylän diagnostiikkaa ja viestirakenteita varten.

¨Turvaohjelman tarkoitus on ohjata FSO-moduulin turvafunktioita päälle ja pois eri tiloissa. Tiloja on tällä hetkellä kymmenen mutta niiden lukumäärä tullaan kasvattamaan, kun kestotestiä rakennetaan, uusia versioita turvamoduulista kehitetään ja asiakkailta saadaan tärkeää tietoa ongelmakohdista sekä virheistä. Turvaohjelma löytyy kokonaisuudessaan liitteenä insinöörityön lopusta.

Turvaohjelma alkaa aina watchdog-funtiolla, joka valvoo ohjelmasyklin suorittamiseen käytettyä aikaa (kuva 24). Ohjelmassa watchdog ajaksi asetettiin 500 ms mutta sitä on kestotestiä kehitettäessä tarkoitus ajaa alemmaksi, jotta todellinen suorituskyky saa-

daan selvitettyä. Mikäli ohjelmakierto jostakin syystä hidastelee tai jumiutuu ja watchdog aika ylittyy, siirtyy järjestelmä turvatilaan.

```
0001 (* Toggle Watchdog *)
0002 SF_WDOG(
0003   EN:=TRUE,
0004   WDOG:=500,
0005   RESET:=FALSE,
0006   DONE=>,
0007   ACT_TIME=>,
0008   MAX_TIME=>);
```

Kuva 24. Ohjelman alkuun asetettu watchdog-aika.

Toisena turvaohjelmassa tulee syklisen tiedonsiirron vastaanotto (kuva 25). Funktiolle on määritelty muistipaikat lukemiseen, vaikkakin tässä ohjelmassa ei tavallisen ohjelman puolelta lähetetä mitään turvaohjelman puolelle. Muistipaikat vastaanotettavalle tiedolle on asetettu globaaleihin muuttujiin nimellä G_awCyclicNoneSafe_Rec. Vastaanotettavat tiedot kirjoitetaan 80 sanan taulukkoon, mikä jokaisella kierrolla lähetetään ja luetaan taulukosta.

```
0010 (* Cyclic data exchange receive *)
0011 fbCyclicNoneSafeRec(
0012     EN:=TRUE ,
0013     DATA:= ADR(G_awCyclicNoneSafe_Rec),
0014     DATA_LEN:= SIZEOF(G_awCyclicNoneSafe_Rec),
0015     DONE=> ,
0016     ERR=> ,
0017     ERNO=> );
```

Kuva 25. Syklisen tiedonsiirron vastaanottoon käytetty funktio.

Seuraavaksi turvaohjelmaan asetettiin PROFIsafe-turvaväylän automaattinen kuittaus. PROFIsafe väylän kuittaaminen tapahtuu siten, että PROFIsafe-väylässä kulkevista sisäisistä diagnostikkabiteistä, PROFIsafe-pinosta, poimitaan bitti, joka pyytää kuittausta virheeseen. Tämän jälkeen aktivoidaan PROFIsafe-pinossa virheenkuittausbitti, joka kuittaa virheen väylässä (kuva 26). PROFIsafe-väylässä on jatkuva sisäinen diagnostiikka, jolla voidaan tarkkailla väylän sekä väylässä olevien orjien suorittamista ja eheyttä.

```
0019 (* Automatic  operator acknowledge for PROFIsafe *)
0020 IF drive11_ABB_PS1.OA_Req_S = TRUE THEN
0021     drive11_ABB_PS1.OA_C := TRUE;
0022 ELSE
0023     drive11_ABB_PS1.OA_C := FALSE;
0024 END_IF
```

Kuva 26. Automaattinen kuittaus PROFIsafe-turvaväylälle.

Myös FSO-moduulille tarvitaan automaattinen kuittaus. Virhetilanteessa FSO-moduuli vaatii käyttäjältä kuittauksen, mikä voidaan myös toteuttaa ohjelmallisesti. Moduulin havaitessa sisäisen diagnostiikkavirheen tai turvafunktion pettäessä, siirtyy moduuli turvatilaan, mikä vaatii kuittauksen ajon jatkamiseksi. Moduulille on parametroitu muuttujat Control Builder -ohjelman puolella. Moduulin virheen kuittaus toteutetaan JOS-lausekkeella, jossa kuittauspyynnöstä lähetetään kuittauskäsky moduulille (kuva 27). Kuittaustilanteessa pitää huomioida että myös taajuusmuuttajan ohjaus-sanaa pitää kierrättää. Syklisessä tiedonsiirrossa lähetetty tieto moduulin tilasta ohjaa taajuusmuuttajalle lähetettävää käynnistyskäskyä tavallisen ohjelman puolella.

```
0042 (* Automatic acknowledge for FSO-11 *)
0043 IF drive11_SF_end_ack_req = TRUE THEN
0044     drive11_SF_end_ack := TRUE;
0045 ELSE
0046     drive11_SF_end_ack := FALSE;
```

Kuva 27. Automaattinen kuittaus FSO-moduulille.

Tilakonetta varten asetettiin ajastin, joka laskee kulunutta aikaa. Ajastin sijoitettiin kokonaan JOS-lausekkeen ulkopuolelle. Lausekkeen sisään ensimmäiseksi määrättiin ajastin nollattavaksi, jolloin ajastin alkaa kasvattaa arvoaan alusta. Jokaisessa tilassa ajastimelle annetaan arvo, joka määrittää tilan pituuden. Ensimmäiseen tilaan lisättiin laskuri, jonka tarkoituksena on pitää lukua, montako kertaa tilakone on ajettu läpi (kuva 28).

```
0064 (* State program where safety functions are activated *)
0065 ProgTimer := ProgTimer + T#1ms;
0066
0067 IF StateCheck > 0 AND ProgTimer > StateChangeTime THEN
0068     ProgTimer := T#0s;
0069     IF StateCheck = 1 THEN (* all on*)
0070     LoopNum := LoopNum + 1;
0071     drive11SLS2req := TRUE;
0072     drive11SLS1req := TRUE;
0073     drive11SS1req := TRUE;
0074     drive11SSEreq := TRUE;
0075     drive11POUSreq := FALSE;
0076     drive11STOreq := TRUE;
0077     drive11SLS4req := TRUE;
0078     drive11SLS3req := TRUE;
0079     drive11VarSLSreq := TRUE;
0080     drive11NegScale := TRUE;
0081     drive11PosScale := TRUE;
0082     StateChangeTime := T#5000ms;
```

Kuva 28. Tilakoneen ensimmäinen tila.

Ensimmäisessä tilassa kytketään kaikki turvafunktiot paitsi POUS yhtä aikaa päälle. Toisessa kytketään POUS, kun kaikki muut kytketään pois. Tämä siksi, että kymmenennestä tilasta siirryttäessä, moottorin vielä pyöriessä, suoraan POUS-tilaan syntyy FSO-moduulissa vakava virhe, jota ei pysty automaattisesti kuittaamaan. Kolmas tila antaa moottorille käyntiluvan kytkemällä kaikki turvafunktiot pois päältä.

Neljännessä tilassa kytketään moottorin käyntiä rajoittava SLS2-funtio aktiiviseksi, näin simuloidaan täydestä vauhdista hidastus halutulle rajalle, mikä tässä tapauksessa on 400 kierrosta minuutissa. Viides tila simuloi turvallista pysäytystä, eli SS1-turvafuntion aktivointia. SS1 pysäyttää moottorin rampilla ja kytkee STO-funktion päälle. Kuudennessa, seitsemännessä ja kahdeksannessa tilassa käytetään SLS1, -2, ja -3 turvafunktioita. Yhdeksäs tila kytkee kaikki neljä SLS-funktiota päälle, jolla pyritään testaamaan päällekkäisten turvafunktioiden vaikutusta turvamoduulin toimintaan.

Viimeisessä, eli kymmenennessä tilassa kytketään Variable SLS-turvafunktio aktiiviseksi. Tässä turvafunktiossa voidaan säätää sitä rajaa, johon turvamoduuli rajoittaa moottorin nopeuden. Koska myös tätä funktiota halutaan testata mahdollisimman monipuolisesti, joudutaan rajaa muuttamaan säännöllisesti. Viimeisessä tilassa lisätään siis tuhat funktion arvoon jokaisella kierroksella. Kun tarpeeksi monta kierrosta tilakonetta on pyörähtänyt ympäri, asetetaan arvo takaisin alkuperäiseen, jolloin arvoa aletaan kasvattaa uudestaan (kuva 29).

```
0208 IF drive11VarSLSLimit = 15000 THEN (* when 15000 reached, back to 1000*)
0209     drive11VarSLSLimit := 1000;
0210 END_IF
```

Kuva 29. Variable SLS-turvafuntion rajan kierrätys.

Tilakone muuttaa ainoastaan ensimmäisen taajuusmuuttajan arvoja. Muille taajuusmuuttajille arvot saadaan yksinkertaisella kopioinnilla. Turvafunktiot kopioidaan JOS-lausekkeen jälkeen kaikille taajuusmuuttajille, jolloin tilakoneessa riittää ainoastaan yhden taajuusmuuttajan turvafunktioiden ohjaus.

Koska syklinen tiedonsiirto toteutetaan tallentamalla arvot kahdeksankymmenen WORD-muuttujan taulukoihin, joudutaan siirrettävät tiedot muuttamaan WORD-muotoon. Muutettavia arvoja oli BOOL-, TIME- ja DWORD-muodossa. BOOL-, ja TIME-muotoiset muuttujat voidaan kääntää suoraan, kun DWORD-muuttuja on kooltaan suurempi ja joudutaan tallentamaan kahteen sanaan. DWORD muutetaan lähetyksen ajaksi kahteen sanaan käyttämällä ROL-komentoa, jotka tavallisen ohjelman puolella muutetaan takaisin yhdeksi DWORD-muuttujaksi (kuva 30).

```
0267 LoopNumSend := DWORD_TO_WORD(LoopNum);
0268 LoopNumSend2 := DWORD_TO_WORD(ROL(LoopNum, 16));
```

Kuva 30. DWORD-muodossa olevan muuttujan kääntäminen WORD-muotoon.

Tässä tapauksessa LoopNum-muuttuja on DWORD-muodossa, koska kestotestin ollessa käynnissä pitkiä aikoja on todennäköistä, että yksinkertainen WORD-muuttuja ei riitä laskemaan kuluneita kierroksia. WORD-muuttujan maksimiarvo on 65535, kun DWORD-muuttujan on 4294967295. Yhden ohjelmakierron kestäessä 85 sekuntia, voisi WORD-muuttujalla näin ollen tallentaa kierroksia noin 64 päivää, mikä kestotesteissä on kohtalaisen lyhyt aika. DWORD-muodossa looppeja voidaan laskea jopa yli 11000 vuotta.

Kun lähetykseen kerättävät tiedot on muutettu WORD-muotoon, voidaan aloittaa taulukkoon kirjoittaminen. Kahdeksankymmenen muuttujan taulukkoon päädyttiin siten, että lähetettävien muuttujien lukumäärä on 73, joista suurin osa, jopa 55, on eri taajuusmuuttajien PROFIsafe-status-sanojen eri bittejä. Status-sanojen bittejä ei sinänsä tarvita ohjelmassa muuta kuin virheen sattuessa, taajuusmuuttajan tietojen tallennukseen.

Taulukkokirjoituksen jälkeen jäljelle jää taulukon lähettäminen tavallisen ohjelman vastaanotettavaksi (kuva 31). Lähetys tapahtuu samalla periaatteella kuin vastaanotto, sillä erotuksella, että tällä kertaa käytetään taulukkoa G_awCyclicNoneSafe_Send, joka on kirjoitettu edellisillä riveillä. Taulukkokirjoitus ja -lähetys tehdään jokaisella ohjelmakierrolla, mikä kuormittaa prosessoria. Tarpeellisuutensa lisäksi, syklisen tiedonsiirron aiheuttama rasitus palvelee kestotestitarkoitusta erinomaisesti.

```
0403 (* Cyclic data exchange send *)
0404 fbCyclicNoneSafeSend(
0405     EN:=TRUE,
0406     DATA:= ADR(G_awCyclicNoneSafe_Send),
0407     DATA_LEN:= SIZEOF(G_awCyclicNoneSafe_Send),
0408     DONE=> ,
0409     ERR=> ,
0410     ERNO=> );
```

Kuva 31. Sylkisen tiedonsiirron lähettämiseen käytetty funktio.

8 Mittaustulokset

Alkuperäinen tavoite mittaustulosten keräämisestä jäi pahasti kesken ohjelman viivästyksien takia. Ohjelma todettiin toimivaksi aluksi käsin syöttämällä arvoja taulukkoon ja kirjoittamalla muistikortille. Ohjelma kirjoitti taulukkoon tietoa simuloimalla virheitä taajuusmuuttajille, joten ohjelma toimii, emme kuitenkaan ehtineet pitää kestotestiä päällä tarpeeksi pitkiä aikoja, että olisimme saaneet kirjoituksia aikaiseksi. Osasyy tälle on, että kestotesti rakennettiin demoräkeille sellaiseen laboratorioon, jossa kestotesti pitää yöksi sammuttaa. Kestotestin myöhemmässä vaiheessa, kun testi rakennetaan alueelle missä testiä voidaan suorittaa ympärivuorokautisesti, on mittaustiedon keruulle paremmat edellytykset.

Ensimmäiset kestotestin koekäytöt ovat todistaneet, että FSO-turvamoduuli toimii loistavasti. Virheitä tulee todella harvoin, eikä kestotestissä ole ilmennyt tavallisen käytön aikana ongelmia. Turvafunktioiden luotettavuuden lisäksi käytettävyysaika on ollut erinomaista, eikä vääriä hälytyksiä tai virheitä ole ilmennyt. Käytettävyysaikaa saadaan kuitenkin paremmin testattua, kun testi on käynnissä tauotta useita viikkoja ja kuukausia. Tehtaiden turhien seisakkien välttämiseksi käytettävyysaika on turvafunktioiden ohella yksi tärkeimmistä tekijöistä.

Erittäin positiiviset tulokset johtuvat oletettavasti osittain myös watchdog-ajan liian suuresta arvosta. Kestotestin kehitysvaiheessa ajasta määritettiin riittävän suuri, että muun prosessin toiminta voitiin varmistaa, mikä onnistui parhaiten ilman ylimääräisiä väylä- ja ohjelmadiagnostiikkavirheitä. Kestotestiä kehitettäessä watchdog-aikaa leikataan, jolloin väylän ja turvaohjelman todellinen suorituskyky saadaan selvitettyä. Ohjelman käynnissäolon aikana PROFIsafe-vasteajat ovat kuitenkin pysyneet maltillisina, eikä suuria heittoja ajoissa ole ilmennyt.

Ohjelma kirjoittaa tiedot muistikortille dat-muotoiseen tiedostoon, jota voi lukea esimerkiksi Windows-käyttöjärjestelmän tekstieditorilla (kuva 32). Tiedostoon muodostuu puolipisteellä erotellut alkiot, johon jokainen sana järjestyksessä kirjataan. Samaan tiedostoon voidaan kirjoittaa useita kertoja, jolloin uusi tieto tulostuu vanhan tiedon jatkoksi. Mikäli tiedostossa on vanhaa tietoa, tallentaa logiikka vanhan tiedon bak-tiedostoon, joka toimii varmuuskopiona. Tämä toimii luotettavasti mutta kortilta lukeminen on aikaa vievää ja se joudutaan suorittamaan paikan päällä sammuttamalla logiikka.

1;0;0;1111;4736;32;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;1212;47
36;0;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;1313;4736;32;0;0;0;0;
0;1;0;0;0;0;0;0;0;0;0;0;0;1414;4736;0;0;0;0;0;0;1;0;0;0;0;0;
0;0;0;0;0;0;0;0;1;1;0;1111;4800;0;128;0;0;0;0;0;0;0;0;0;0;1;30
;0;30;33;59;1212;4800;32;128;0;0;0;0;0;0;0;0;0;0;0;1;21;0;33;3
3;49;1313;4800;0;160;0;0;0;0;0;0;0;0;0;0;0;0;29;0;32;33;41;141
4;4800;0;160;0;0;0;0;0;0;0;0;0;0;0;0;21;0;33;33;49;1;1;0;1111;
5687;0;128;0;0;0;0;0;0;0;0;0;0;1;30;0;33;33;59;1212;5687;32;
128;0;0;0;0;0;0;0;0;0;0;0;1;30;0;32;33;49;1313;5687;0;160;0;0;
0;0;0;0;0;0;0;0;0;0;0;30;0;40;33;41;1414;5687;32;160;0;0;0;0;0;0
;0;0;0;0;0;0;30;0;33;33;49;1;1;0;1111;5943;32;160;0;0;0;0;0;0;
0;0;0;0;0;0;29;0;32;33;59;1212;5943;32;160;0;0;0;0;0;0;0;0;0;0
;0;0;29;0;30;33;49;1313;5943;32;128;0;0;0;0;0;0;0;0;0;0;1;30
;0;30;32;41;1414;5943;0;128;0;0;0;0;0;0;0;0;0;0;1;32;0;32;33
;49;0

Kuva 32. Väliaikatietojen luku SD-muistikortilta Windows-tekstieditorilla.

Tallennetun tiedon lukeminen muistikortilta vaatii muistilistan, jotta arvosta selviää, mikä tieto on kyseessä. Tällä hetkellä tieto on pakattua, eikä kunkin arvon merkitystä näe pelkän taulukon avulla. Väliaikatietojen tallennuksessa on kunkin taajuusmuuttajan tiedot eroteltu siten, että ensimmäisen laitteen arvot alkavat 1111-arvoisen alkion jälkeen. Toisen arvot alkavat 1212-alkion, kolmannen 1313-alkion ja neljännen 1414-alkion jälkeen. Tulevaisuudessa tarkoituksena on joko uudenlainen tallennustyyli tai tämänhetkisen tiedon avaaminen Excel-tiedostoon, johon on syötetty kunkin arvon tiedot omaan sarakkeeseensa.

9 Päätelmät

Tavoitteet insinöörityölle ovat suurimmilta osin täyttyneet, vaikkakin joitakin kompromisseja jouduttiin tekemään. Työ tehtiin demoräkeille väliaikaiseen laboratorioon, lopullisen sijoituspaikan ollessa epäselvä. Räkeille rakennettu laitteisto on valmis siirrettäväksi kestotestikäytävälle, ohjelmisto on valmis sekä tiedonkeruu onnistuu muistikortille. Mittaustuloksia ei ole ehtinyt kerääntymään, mutta edellytykset niiden keruulle ovat olemassa. Johdannossa haastavimmaksi osuudeksi arvioitu PROFIsafe-yhteyden ylläpito osoittautui oikein toimivaksi ja suuremmilta ongelmilta vältyttiin.

Demoräkit muodostivat erittäin hyvän alustan kestotestien kehittämiselle. Räkeistä löytyneet valmiit riviliittimet, moottorit ja DIN-kiskot auttoivat suunnattomasti testiympäristön kokoamisessa. Insinöörityön tarkoituksena oli valmistella testi sellaiseen vaiheeseen, että se voidaan siirtää sellaisenaan kestotestikäytävälle ABB:n moottoritehtaan kellariin. Vaikka alusta on alati päivittyvä ja kehitettävä, onnistui kestotestin kehitys tähän pisteeseen onnistuneesti ja siirto voidaan toteuttaa.

Työ aloitettiin teoriaosan kokoamisella, jonka aikana laitteet tilattiin ABB:n tehtailta Suomesta ja Saksasta. Työnjako osoittautui järkeväksi, koska kaikkien laitteiden saapumisessa kului noin kaksi viikkoa. Teoriaosuus saatiin kasattua suurimmilta osin tuona aikana ja työn kokoaminen voitiin aloittaa. Asennuksissa suurin haaste oli laitteistojen ohjelmistopäivityksissä, joita tehtiin työn aikana useita kertoja. Fyysinen asennustyö oli helpohkoa ammattikorkeakoulun tarjoaman hyvän koulutuksen pohjalta. Työssä suurempi haaste oli myös laajuudeltaan suurempi ohjelmointiosuus, jossa kului runsaasti aikaa uuden opiskeluun ja ongelmien ratkomiseen.

Kestotesti rakennettiin insinöörityötä varten neljälle taajuusmuuttajalle. Kehitetty ohjelma antaa hyvät valmiudet laajentaa testiä useammalle laitteelle, mikä tullaan tulevaisuudessa myös toteuttamaan. Kestotestiä tullaan kehittämään siten, että samassa väylässä olisi mahdollisimman monipuolisesti eri laitteita ja versioita, jotta väylän todellinen suorituskyky ja monipuolisuus saadaan selvitettyä. Kestotestiin tullaan lisäämään FSO-moduulin uudet versiot heti kun mahdollista.

Tavoitteissa ollut vikojen etäluku toteutetaan kestotestin myöhemmässä vaiheessa. Etäkäyttöön tarkoitettu ABB:n NETA-21-etävalvontatyökalu ei ollut käytettävissä, joten tiedon tallennus toteutettiin muistikortille. Kestotestin lopulliseen sijoituspaikkaan tul-

laan todennäköisesti rakentamaan etäkäyttö, jolloin myös tallennetut vikatiedot voisi lukea toimistolta selaimella. Tällä hetkellä vikatietoja luetaan siirtämällä logiikasta muistikortti tietokoneeseen ja tarkastelemalla tallennettua tietoa suoraan kortilta.

Kestotestit tullaan toteuttamaan yhteistyössä muiden osastojen kanssa siten, että osa testeistä implementoidaan ATF-testiryhmän järjestelmiin ja osa rakennetaan HPD-ryhmän kanssa. ATF, eli automated testing framework, on ohjelma, joka testaa kaikki valmistuneet versiot automaattisesti joka päivä. Toisin sanoen jokainen versio voidaan tuoreeltaan testata, kun ATF-ohjelmisto lataa uusimman version joka aamu ja aloittaa testit. Tämä ei kuitenkaan ole perinteinen kestotesti koska järjestelmä pyörii ainoastaan päivällä.

HPD, eli high power drives on osasto, joka vastaa suurempitehoisten kaappitaajuusmuuttajien kehityksestä. Heillä on laaja testiympäristö, johon tässä insinöörityössä kehitetty kestotesti siirretään myös suurempitehoisten taajuusmuuttajien testausta varten. Muiden osastojen kanssa tehty yhteistyö osoittaa sen, että toiminnallinen turvallisuus on tullut jäädäkseen ja sitä kehitetään hyvin vahvasti jokaisella rintamalla. Toisaalta yhteistyö tuo myös paljon haasteita, mutta turvamoduulit on tarkoitettu koko ACS880-tuoteperheen käytettäviksi ja yhteensopiviksi useilla alustoilla. Aluksi joudutaan hyväksikäyttämään muiden osastojen valmiita testijärjestelmiä, mutta myöhemmin tulevaisuudessa toivottavasti saadaan turvamoduuleille oma, itsenäinen testausjärjestelmä.

Uusien versioiden jatkuva valmistuminen aiheutti päänvaivaa, koska jokaisessa versiossa oli jotakin uutta kestotestissä hyväksikäytettävää. ABB:lla on valtava tuotekehitysosasto, mikä näkyi vahvasti myös uusien ohjelmaversioiden valmistumisnopeudessa. Suurin osa versioista on talon sisäisiä, joten ne eivät näy asiakkaille, talon sisällä versionhallinta on kuitenkin haastavaa.

Lähitulevaisuudessa on valmistumassa uudet FSO-moduulit, FSO-12 ja FSO-21, jotka tuovat kestotestiin lisähaastetta uusien turvafunktioiden lisäämisessä logiikkaohjelmaan. FSO-21-moduulin julkaisussa uusina turvafunktioina tulee Safe Speed Monitor (SSM), Safe Accelleration Range (SAR) ja Safe Direction (SDI). Nimensä mukaisesti SSM valvoo moottorin nopeutta, SAR takaa turvalliset rajat kiihdytykselle ja SDI estää väärään suuntaan käymisen. Myöhemmässä vaiheessa FSO-21-moduuliin lisätään toinen safe stop-funktio ja turvallisen pysäytyksen safe operating stop (SOS). FSO-21 eroaa FSO-11:sta ja -12:sta siten, että sitä voidaan käyttää myös enkooderilla, mikä

mahdollistaa tarkemman valvonnan ja enemmän turvafunktioita. FSO-11 ja -12 toimivat ainoastaan ilman enkooderia, jolloin arvot ovat arvioituja, eikä yhtä korkeaa turvatasoa pystytä saavuttamaan.

Koska tässä insinöörityössä käytetyt moduulit ovat vielä niin uusia, ovat useimmin kysytyt kysymykset projektipalavereissa olleet ”mitä halutaan testata?” ja ”mitä laitteistoja tarvitsemme?”. Nämä kestotestit ja siinä käytetyt moduulit ovat vain alkua ja niitä tullaan kehittämään paljon tulevaisuudessa. Tulevaisuudessa turva-automaatio tulee olemaan suuri osa teollisuutta ja integroituna yksinkertaisimpiinkin järjestelmiin. On selvää, että koko automaatioala tulee muuttumaan lähivuosina ja on mielenkiintoista nähdä mihin se kehittyy sekä olla osallisena sen kehityksessä.

Lähteet

2006/42/EY-Direktiivi. 2006. Verkkodokumentti. <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:157:0024:0086:fi:PDF>. Luettu 10.2.2014.

ABB Oy 1. 2014. Who we are - ABB in bief. Verkkodokumentti. <http://new.abb.com/about/abb-in-brief>. Luettu 14.2.2014.

ABB Oy 2. 2014. Suomalaiset juuret: Strömbergin jalanjäljillä vuodesta 1889. Verkkodokumentti. <http://new.abb.com/fi/abb-lyhyesti/historia/suomalaiset-juuret>. Luettu 14.2.2014

ABB Oy 3. 2014. Who we are - Group structure. Verkkodokumentti. <http://new.abb.com/about/abb-in-brief/group-structure>. Luettu 14.2.2014.

ABB Oy 4. 2014. ABB Suomessa. Verkkodokumentti. <http://new.abb.com/fi/abb-lyhyesti/suomessa>. Luettu 14.2.2014.

AC500. The scalable PLC for customized automation. Verkkodokumentti. <http://vfservis.cz/files/002078_prospekt_ac500_en.pdf>. Luettu 11.3.2014.

AC500 PLC. The ideal solution for water. Flexibility, reliability & efficiency. Esite. <http://www05.abb.com/global/scot/scot397.nsf/veritydisplay/54122cd9e6ddae86c1257c2100539daa/$file/1SBC125046B0201-AC500%20PLC%20ideal%20solution%20for%20water_br.pdf>. Luettu 11.3.2014.

AC500-S Safety User Manual. 2013. <http://abblibrary.abb.com/global/scot/scot397.nsf/veritydisplay/0c5bd7666fe5e300c1257c2100539e83/$file/3adr025091m0202.pdf>. Luettu 18.2.2014.

ACS880 CODESYS. 2013. PowerPoint-esitys. <Drive CODESYS programming for ACS880 rev3.pptx>.

ACS880 firmware update. 2014. Primary control program 1.62 & roadmap for coming releases. <http://abblibrary.abb.com/global/scot/scot201.nsf/veritydisplay/8147fb254355db0ac1257c6f0040f95e/$file/abb_acs880_sw_update_partner_webinar_2014_01_28.pdf>. Luettu 18.3.2014.

ACS880 integroidut turvatoiminnot. 2014. Verkkodokumentti. <http://www.abb.fi/cawp/seitp202/6a86e45a4b44ebf0c1257992001ffc3d.aspx>. Luettu 7.2.2014.

ACS880 Sales. 2013. PowerPoint-esitys. <ACS880_sales_presentation_v_2013_01_31.ppt>

ACS880 Single drives, catalog. Verkkodokumentti. <http://abblibrary.abb.com/global/scot/scot201.nsf/veritydisplay/78a02f334321988fc1257c6e0040a0ad/$file/16957_ACS880_single_drives_3AUA0000098111_EN_RevI_lowres.pdf>. Luettu 10.2.2014

ACS880 STO. 2012. PowerPoint-esitys. <edt_2012_acs88 fs.ppt>

CM579-PNIO. Control Builder Plus contents help, CM579-PNIO Communication Module.

FENA-11 User´s Manual. 2014. FENA-01/-11/-21 Ethernet adapter module. <http://abblibrary.abb.com/global/scot/scot201.nsf/veritydisplay/8d15c812cd6c71a2c1257c6f005f46e8/$file/en_fena01_11_21_um_b_a4.pdf>. Luettu 12.2.2014

FSO-11 User´s manual. User´s manual, FSO-11 safety functions module. Draft 30.01.2014.

IEC61508-0. Functional safety of electrical/electronic/programmable electronic safety-related systems. Functional safety and IEC61508. 2005. Geneva: The International Electrotechnical Comission

IEC61508-1. Functional safety of electrical/electronic/programmable electronic safety-related systems. 2010. General requirements. Geneva: The International Electrotechnical Comission

IEC61508-4. Functional safety of electrical/electronic/programmable electronic safety-related systems. 2010. Definitions and abbreviations. Geneva: The International Electrotechnical Comission

IEC61511-1. Functional safety - Safety instrumented systems for the process industry sector. 2003. Framework, definitions, system, hardware and software requirements. Geneva: The International Electrotechnical Comission

ISO13849-1. Safety of machinery - Safety-related parts of control systems. 2006. General principles for design. Geneva: The International Organization for Standardization.

Kostiainen, Anne. DI. 2014. ABB Oy Drives, Helsinki. Keskustelu. 11.4.2014.

Lyu, R. Michael. 1996. Handbook of software reliability engineering. Amerikan yhdysvallat: The McGraw-Hill Companies, Inc.

PM583-ETH. Control Builder Plus contents help, CPUs PM57x PM58x and PM59x.

PROFIsafe 1. 2014. PROFIsafe. Verkkodokumentti. <http://www.profibus.com/technology/profisafe/>. Luettu 12.2.2014.

PROFIsafe 2. 2014. Overwiew. Verkkodokumentti.
<http://www.profibus.com/technology/profisafe/overview/>. Luettu 12.2.2014.

PROFIsafe 3. 2014. Benefits. Verkkodokumentti.
<http://www.profibus.com/technology/profisafe/benefits/>. Luettu 12.2.2014.

Qualmark 1. 2014. Why HALT Works. Verkkodokumentti.
<http://www.qualmark.com/technical-halt-hass>. Luettu 7.2.2014.

Qualmark 2. 2014. Performing HALT Testing. Verkkodokumentti.
<http://www.qualmark.com/technical-performing-halt>. Luettu 7.2.2014.

Reliability Engineering. 2014. About Reliability Engineering. Verkkodokumentti.
<http://www.weibull.com/basics/reliability.htm>. Luettu 18.2.2014.

Standardit 1. 2014. Mitä standardisointi on?. Verkkodokumentti.
<http://www.sfs.fi/standardien_laadinta/mita_standardisointi_on>. Luettu 11.2.2014

Standardit 2. 2014. Eurooppalainen standardisointi. Verkkodokumentti.
<http://www.sfs.fi/standardien_laadinta/mita_standardisointi_on/standardisoinnin_maailmankartta/eurooppalainen_standardisointi>. Luettu 11.2.2014

Standardit 3. 2014. Kansainvälinen standardisointi. Verkkodokumentti.
<http://www.sfs.fi/standardien_laadinta/mita_standardisointi_on/standardisoinnin_maailmankartta/kansainvalinen_standardisointi>. Luettu 11.2.2014

Demoräkit

Tavallisen ohjelman pääohjelma

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     VersionNumber: WORD := 1;
0004
0005     cyclicTempValue: INT := 10;
0006     cyclicDelta: INT := 1;
0007
0008     StateCheck: WORD;
0009     SLS2req: WORD;
0010     SLS1req: WORD;
0011     SS1req: WORD;
0012     SSEreq: WORD;
0013     POUSreq: WORD;
0014     STOreq: WORD;
0015     SLS4req: WORD;
0016     SLS3req: WORD;
0017     VarSLSreq: WORD;
0018     ProgTimer: WORD;
0019     NegScale: WORD;
0020     PosScale: WORD;
0021     VarSLSLimit: WORD;
0022     drive11STOact: WORD;
0023     startTimer: TIME;
0024     LoopNum: DWORD;
0025     LoopNum2: WORD;
0026     LoopNum1: WORD;
0027
0028     drive11FaultNum: WORD := 1;
0029     drive12FaultNum: WORD := 1;
0030     drive13FaultNum: WORD := 1;
0031     drive14FaultNum: WORD := 1;
0032
0033     d11: WORD := 1;
0034     d12: WORD := 1;
0035     d13: WORD := 1;
0036     d14: WORD := 1;
0037     oldError11: BOOL;
0038     oldError12: BOOL;
0039     oldError13: BOOL;
0040     oldError14: BOOL;
0041
0042     drive11PROFIsafeStatusCRC: WORD;
0043     drive11PROFIsafeStatusActFV: WORD;
0044     drive11PROFIsafeStatusConsnrR: WORD;
0045     drive11PROFIsafeStatusDeviceFault: WORD;
0046     drive11PROFIsafeStatusFVactivated: WORD;
0047     drive11PROFIsafeStatusHostCRC: WORD;
0048     drive11PROFIsafeStatusHostTimeout: WORD;
0049     drive11PROFIsafeStatusiParEN: WORD;
0050     drive11PROFIsafeStatusiParOK: WORD;
0051     drive11PROFIsafeStatusOAC: WORD;
0052     drive11PROFIsafeStatusOAReq: WORD;
0053     drive11PROFIsafeStatusToggleD: WORD;
0054     drive11PROFIsafeStatusResTim: WORD;
0055     drive11PROFIsafeStatusWDTimeout: WORD;
0056     drive12PROFIsafeStatusCRC: WORD;
0057     drive12PROFIsafeStatusActFV: WORD;
0058     drive12PROFIsafeStatusConsnrR: WORD;
0059     drive12PROFIsafeStatusDeviceFault: WORD;
0060     drive12PROFIsafeStatusFVactivated: WORD;
0061     drive12PROFIsafeStatusHostCRC: WORD;
0062     drive12PROFIsafeStatusHostTimeout: WORD;
0063     drive12PROFIsafeStatusiParEN: WORD;
0064     drive12PROFIsafeStatusiParOK: WORD;
0065     drive12PROFIsafeStatusOAC: WORD;
0066     drive12PROFIsafeStatusOAReq: WORD;
```

```
0067    drive12PROFIsafeStatusToggleD: WORD;
0068    drive12PROFIsafeStatusResTim: WORD;
0069    drive12PROFIsafeStatusWDTimeout: WORD;
0070    drive13PROFIsafeStatusCRC: WORD;
0071    drive13PROFIsafeStatusActFV: WORD;
0072    drive13PROFIsafeStatusConsnrR: WORD;
0073    drive13PROFIsafeStatusDeviceFault: WORD;
0074    drive13PROFIsafeStatusFVactivated: WORD;
0075    drive13PROFIsafeStatusHostCRC: WORD;
0076    drive13PROFIsafeStatusHostTimeout: WORD;
0077    drive13PROFIsafeStatusiParEN: WORD;
0078    drive13PROFIsafeStatusiParOK: WORD;
0079    drive13PROFIsafeStatusOAC: WORD;
0080    drive13PROFIsafeStatusOAReq: WORD;
0081    drive13PROFIsafeStatusToggleD: WORD;
0082    drive13PROFIsafeStatusResTim: WORD;
0083    drive13PROFIsafeStatusWDTimeout: WORD;
0084    drive14PROFIsafeStatusCRC: WORD;
0085    drive14PROFIsafeStatusActFV: WORD;
0086    drive14PROFIsafeStatusConsnrR: WORD;
0087    drive14PROFIsafeStatusDeviceFault: WORD;
0088    drive14PROFIsafeStatusFVactivated: WORD;
0089    drive14PROFIsafeStatusHostCRC: WORD;
0090    drive14PROFIsafeStatusHostTimeout: WORD;
0091    drive14PROFIsafeStatusiParEN: WORD;
0092    drive14PROFIsafeStatusiParOK: WORD;
0093    drive14PROFIsafeStatusOAC: WORD;
0094    drive14PROFIsafeStatusOAReq: WORD;
0095    drive14PROFIsafeStatusToggleD: WORD;
0096    drive14PROFIsafeStatusResTim: WORD;
0097    drive14PROFIsafeStatusWDTimeout: WORD;
0098
0099    ResponseTimer: TIME := T#0ms;
0100    drive11NewHighValue: WORD;
0101    drive12NewHighValue: WORD;
0102    drive13NewHighValue: WORD;
0103    drive14NewHighValue: WORD;
0104    drive11OldHighValue: WORD;
0105    drive12OldHighValue: WORD;
0106    drive13OldHighValue: WORD;
0107    drive14OldHighValue: WORD;
0108    drive11AlltimeHigh: WORD;
0109    drive12AlltimeHigh: WORD;
0110    drive13AlltimeHigh: WORD;
0111    drive14AlltimeHigh: WORD;
0112    SystemTimer: CurTime;
0113
0114    SysTim: SysTime64;
0115    SysTim1: WORD;
0116    SysTim2: WORD;
0117    SysTim3: WORD;
0118    SysTim4: WORD;
0119
0120    SysTimLog: DWORD;
0121    SysTimTrigger: DWORD := 36000000;
0122    SysTimTriggerHigh: DWORD;
0123    WriteAllOK: BOOL;
0124
0125    OK: WORD := 1;
0126    step11: BYTE := 0;
0127
0128    Write_d11:Write_d11;
0129    Write_d12:Write_d12;
0130    Write_d13:Write_d13;
0131    Write_d14:Write_d14;
0132    Write_OK:Write_OK;
```

```
0133      WriteDone11: BOOL;
0134      WriteDone12: BOOL;
0135      WriteDone13: BOOL;
0136      WriteDone14: BOOL;
0137      WriteDoneOK: BOOL;
0138      WriteError11: BOOL;
0139      WriteError12: BOOL;
0140      WriteError13: BOOL;
0141      WriteError14: BOOL;
0142      WriteErrorOK: BOOL;
0143  END_VAR
0144  VAR_INPUT
0145
0146  END_VAR
0147
0148  VAR_OUTPUT
0149
0150  END_VAR
```

```
0001  drive1Diag;
0002
0003  (* System Time in high and low DWORD (64 bit) *)
0004  SystemTimer(SystemTime := SysTim);
0005
0006  (* Cyclic data exchange *)
0007  StateCheck := I_awCyclicNoneSafe[1];
0008  SLS2req := I_awCyclicNoneSafe[2];
0009  SLS1req := I_awCyclicNoneSafe[3];
0010  SS1req := I_awCyclicNoneSafe[4];
0011  SSEreq := I_awCyclicNoneSafe[5];
0012  POUSreq := I_awCyclicNoneSafe[6];
0013  STOreq := I_awCyclicNoneSafe[7];
0014  SLS4req := I_awCyclicNoneSafe[8];
0015  SLS3req := I_awCyclicNoneSafe[9];
0016  VarSLSreq := I_awCyclicNoneSafe[10];
0017  NegScale := I_awCyclicNoneSafe[11];
0018  PosScale := I_awCyclicNoneSafe[12];
0019  VarSLSLimit := I_awCyclicNoneSafe[13];
0020  ProgTimer := I_awCyclicNoneSafe[14];
0021  drive11STOact := I_awCyclicNoneSafe[15];
0022  LoopNum1 := I_awCyclicNoneSafe[16];
0023  LoopNum2 := I_awCyclicNoneSafe[17];
0024  drive11PROFIsafeStatusActFV := I_awCyclicNoneSafe[18];
0025  drive11PROFIsafeStatusCRC := I_awCyclicNoneSafe[19];
0026  drive11PROFIsafeStatusConsnrR := I_awCyclicNoneSafe[20];
0027  drive11PROFIsafeStatusDeviceFault := I_awCyclicNoneSafe[21];
0028  drive11PROFIsafeStatusFVactivated := I_awCyclicNoneSafe[22];
0029  drive11PROFIsafeStatusHostCRC := I_awCyclicNoneSafe[23];
0030  drive11PROFIsafeStatusHostTimeout := I_awCyclicNoneSafe[24];
0031  drive11PROFIsafeStatusiParEN := I_awCyclicNoneSafe[25];
0032  drive11PROFIsafeStatusiParOK := I_awCyclicNoneSafe[26];
0033  drive11PROFIsafeStatusOAC := I_awCyclicNoneSafe[27];
0034  drive11PROFIsafeStatusOAReq := I_awCyclicNoneSafe[28];
0035  drive11PROFIsafeStatusToggleD := I_awCyclicNoneSafe[29];
0036  drive11PROFIsafeStatusResTim := I_awCyclicNoneSafe[30];
0037  drive11PROFIsafeStatusWDTimeout := I_awCyclicNoneSafe[31];
0038  drive12PROFIsafeStatusActFV := I_awCyclicNoneSafe[32];
0039  drive12PROFIsafeStatusCRC := I_awCyclicNoneSafe[33];
0040  drive12PROFIsafeStatusConsnrR := I_awCyclicNoneSafe[34];
0041  drive12PROFIsafeStatusDeviceFault := I_awCyclicNoneSafe[35];
0042  drive12PROFIsafeStatusFVactivated := I_awCyclicNoneSafe[36];
0043  drive12PROFIsafeStatusHostCRC := I_awCyclicNoneSafe[37];
0044  drive12PROFIsafeStatusHostTimeout := I_awCyclicNoneSafe[38];
0045  drive12PROFIsafeStatusiParEN := I_awCyclicNoneSafe[39];
0046  drive12PROFIsafeStatusiParOK := I_awCyclicNoneSafe[40];
0047  drive12PROFIsafeStatusOAC := I_awCyclicNoneSafe[41];
0048  drive12PROFIsafeStatusOAReq := I_awCyclicNoneSafe[42];
```

```
0049 drive12PROFIsafeStatusToggleD := I_awCyclicNoneSafe[43];
0050 drive12PROFIsafeStatusResTim := I_awCyclicNoneSafe[44];
0051 drive12PROFIsafeStatusWDTimeout := I_awCyclicNoneSafe[45];
0052 drive13PROFIsafeStatusActFV := I_awCyclicNoneSafe[46];
0053 drive13PROFIsafeStatusCRC := I_awCyclicNoneSafe[47];
0054 drive13PROFIsafeStatusConsnrR := I_awCyclicNoneSafe[48];
0055 drive13PROFIsafeStatusDeviceFault := I_awCyclicNoneSafe[49];
0056 drive13PROFIsafeStatusFVactivated := I_awCyclicNoneSafe[50];
0057 drive13PROFIsafeStatusHostCRC := I_awCyclicNoneSafe[51];
0058 drive13PROFIsafeStatusHostTimeout := I_awCyclicNoneSafe[52];
0059 drive13PROFIsafeStatusiParEN := I_awCyclicNoneSafe[53];
0060 drive13PROFIsafeStatusiParOK := I_awCyclicNoneSafe[54];
0061 drive13PROFIsafeStatusOAC := I_awCyclicNoneSafe[55];
0062 drive13PROFIsafeStatusOAReq := I_awCyclicNoneSafe[56];
0063 drive13PROFIsafeStatusToggleD := I_awCyclicNoneSafe[57];
0064 drive13PROFIsafeStatusResTim := I_awCyclicNoneSafe[58];
0065 drive13PROFIsafeStatusWDTimeout := I_awCyclicNoneSafe[59];
0066 drive14PROFIsafeStatusActFV := I_awCyclicNoneSafe[60];
0067 drive14PROFIsafeStatusCRC := I_awCyclicNoneSafe[61];
0068 drive14PROFIsafeStatusConsnrR := I_awCyclicNoneSafe[62];
0069 drive14PROFIsafeStatusDeviceFault := I_awCyclicNoneSafe[63];
0070 drive14PROFIsafeStatusFVactivated := I_awCyclicNoneSafe[64];
0071 drive14PROFIsafeStatusHostCRC := I_awCyclicNoneSafe[65];
0072 drive14PROFIsafeStatusHostTimeout := I_awCyclicNoneSafe[66];
0073 drive14PROFIsafeStatusiParEN := I_awCyclicNoneSafe[67];
0074 drive14PROFIsafeStatusiParOK := I_awCyclicNoneSafe[68];
0075 drive14PROFIsafeStatusOAC := I_awCyclicNoneSafe[69];
0076 drive14PROFIsafeStatusOAReq := I_awCyclicNoneSafe[70];
0077 drive14PROFIsafeStatusToggleD := I_awCyclicNoneSafe[71];
0078 drive14PROFIsafeStatusResTim := I_awCyclicNoneSafe[72];
0079 drive14PROFIsafeStatusWDTimeout := I_awCyclicNoneSafe[73];
0080
0081 LoopNum := WORD_TO_DWORD(LoopNum1) + ROL(WORD_TO_DWORD(LoopNum2), 16);
0082
0083 (* Drive start command word *)
0084 IF (drive11STOact = 1 OR SS1req = 1 OR POUSreq = 1) THEN
0085     drive11Command := 1024;
0086     startTimer := T#0s;
0087 ELSIF (drive11STOact = 0 AND startTimer > T#500ms) THEN
0088     drive11Command := 1151;
0089 ELSE
0090     drive11Command := 1150;
0091     startTimer := startTimer + T#3ms;
0092 END_IF;
0093
0094
0095 drive12Command := drive11Command;
0096 drive13Command := drive11Command;
0097 drive14Command := drive11Command;
0098
0099 (* Cycle drive reference values *)
0100 cyclicTempValue := cyclicTempValue+cyclicDelta;
0101
0102 IF cyclicTempValue > 20000 THEN
0103     cyclicTempValue := 20000;
0104     cyclicDelta := -cyclicDelta;
0105 ELSIF cyclicTempValue < cyclicDelta THEN
0106     cyclicDelta := -cyclicDelta;
0107     cyclicTempValue := ABS(cyclicDelta);
0108 END_IF;
0109
0110 (* Copy values to all drives *)
0111 drive11SpeedRef := cyclicTempValue;
0112 drive11RefPZD3 := cyclicTempValue;
0113 drive11RefPZD4 := cyclicTempValue;
0114 drive11RefPZD5 := cyclicTempValue;
```

```
0115 drive11RefPZD6 := cyclicTempValue;
0116 drive11RefPZD7 := cyclicTempValue;
0117 drive11RefPZD8 := cyclicTempValue;
0118 drive11RefPZD9 := cyclicTempValue;
0119 drive11RefPZD10 := cyclicTempValue;
0120 drive11RefPZD11 := cyclicTempValue;
0121 drive11RefPZD12 := cyclicTempValue;
0122
0123 drive12SpeedRef := cyclicTempValue;
0124 drive12RefPZD3 := cyclicTempValue;
0125 drive12RefPZD4 := cyclicTempValue;
0126 drive12RefPZD5 := cyclicTempValue;
0127 drive12RefPZD6 := cyclicTempValue;
0128 drive12RefPZD7 := cyclicTempValue;
0129 drive12RefPZD8 := cyclicTempValue;
0130 drive12RefPZD9 := cyclicTempValue;
0131 drive12RefPZD10 := cyclicTempValue;
0132 drive12RefPZD11 := cyclicTempValue;
0133 drive12RefPZD12 := cyclicTempValue;
0134
0135 drive13SpeedRef := cyclicTempValue;
0136 drive13RefPZD3 := cyclicTempValue;
0137 drive13RefPZD4 := cyclicTempValue;
0138 drive13RefPZD5 := cyclicTempValue;
0139 drive13RefPZD6 := cyclicTempValue;
0140 drive13RefPZD7 := cyclicTempValue;
0141 drive13RefPZD8 := cyclicTempValue;
0142 drive13RefPZD9 := cyclicTempValue;
0143 drive13RefPZD10 := cyclicTempValue;
0144 drive13RefPZD11 := cyclicTempValue;
0145 drive13RefPZD12 := cyclicTempValue;
0146
0147 drive14SpeedRef := cyclicTempValue;
0148 drive14RefPZD3 := cyclicTempValue;
0149 drive14RefPZD4 := cyclicTempValue;
0150 drive14RefPZD5 := cyclicTempValue;
0151 drive14RefPZD6 := cyclicTempValue;
0152 drive14RefPZD7 := cyclicTempValue;
0153 drive14RefPZD8 := cyclicTempValue;
0154 drive14RefPZD9 := cyclicTempValue;
0155 drive14RefPZD10 := cyclicTempValue;
0156 drive14RefPZD11 := cyclicTempValue;
0157 drive14RefPZD12 := cyclicTempValue;
0158
0159 (* tResponseTimeMS max value record *)
0160 ResponseTimer := ResponseTimer + T#3ms;
0161
0162 IF drive11PROFIsafeStatusResTim > drive11NewHighValue THEN
0163 drive11NewHighValue := drive11PROFIsafeStatusResTim;
0164 END_IF;
0165
0166 IF drive12PROFIsafeStatusResTim > drive12NewHighValue THEN
0167 drive12NewHighValue := drive12PROFIsafeStatusResTim;
0168 END_IF;
0169
0170 IF drive13PROFIsafeStatusResTim > drive13NewHighValue THEN
0171 drive13NewHighValue := drive13PROFIsafeStatusResTim;
0172 END_IF;
0173
0174 IF drive14PROFIsafeStatusResTim > drive14NewHighValue THEN
0175 drive14NewHighValue := drive14PROFIsafeStatusResTim;
0176 END_IF;
0177
0178 IF ResponseTimer > T#5000ms THEN
0179 drive11OldHighValue := drive11NewHighValue;
0180 drive12OldHighValue := drive12NewHighValue;
```

```
0181 drive13OldHighValue := drive13NewHighValue;
0182 drive14OldHighValue := drive14NewHighValue;
0183 drive11NewHighValue := 0;
0184 drive12NewHighValue := 0;
0185 drive13NewHighValue := 0;
0186 drive14NewHighValue := 0;
0187 ResponseTimer := T#0ms;
0188 END_IF;
0189
0190 IF drive11OldHighValue > drive11AlltimeHigh THEN
0191 drive11AlltimeHigh := drive11OldHighValue;
0192 END_IF;
0193
0194 IF drive12OldHighValue > drive12AlltimeHigh THEN
0195 drive12AlltimeHigh := drive12OldHighValue;
0196 END_IF;
0197
0198 IF drive13OldHighValue > drive13AlltimeHigh THEN
0199 drive13AlltimeHigh := drive13OldHighValue;
0200 END_IF;
0201
0202 IF drive14OldHighValue > drive14AlltimeHigh THEN
0203 drive14AlltimeHigh := drive14OldHighValue;
0204 END_IF;
0205
0206 (* In case of fault, write data to table *)
0207 IF drive11Status.3 AND oldError11  = FALSE THEN
0208     Drive11_write_table[d11] := drive11FaultNum;
0209     Drive11_write_table[d11 + 1] := VersionNumber;
0210     Drive11_write_table[d11 + 2] := LoopNum1;
0211     Drive11_write_table[d11 + 3] := LoopNum2;
0212     Drive11_write_table[d11 + 4] := drive11Status;
0213     Drive11_write_table[d11 + 5] := drive11PSStatus;
0214     Drive11_write_table[d11 + 6] := drive11PSControl;
0215     Drive11_write_table[d11 + 7] := drive11PROFIsafeStatusActFV;
0216     Drive11_write_table[d11 + 8] := drive11PROFIsafeStatusCRC;
0217     Drive11_write_table[d11 + 9] := drive11PROFIsafeStatusConsnrR;
0218     Drive11_write_table[d11 + 10] := drive11PROFIsafeStatusDeviceFault;
0219     Drive11_write_table[d11 + 11] := drive11PROFIsafeStatusFVactivated;
0220     Drive11_write_table[d11 + 12] := drive11PROFIsafeStatusHostCRC;
0221     Drive11_write_table[d11 + 13] := drive11PROFIsafeStatusHostTimeout;
0222     Drive11_write_table[d11 + 14] := drive11PROFIsafeStatusiParEN;
0223     Drive11_write_table[d11 + 15] := drive11PROFIsafeStatusiParOK;
0224     Drive11_write_table[d11 + 16] := drive11PROFIsafeStatusOAC;
0225     Drive11_write_table[d11 + 17] := drive11PROFIsafeStatusOAReq;
0226     Drive11_write_table[d11 + 18] := drive11PROFIsafeStatusToggleD;
0227     Drive11_write_table[d11 + 19] := drive11PROFIsafeStatusResTim;
0228     Drive11_write_table[d11 + 20] := drive11PROFIsafeStatusWDTimeout;
0229     Drive11_write_table[d11 + 21] := drive11NewHighValue;
0230     Drive11_write_table[d11 + 22] := drive11OldHighValue;
0231     Drive11_write_table[d11 + 23] := drive11AlltimeHigh;
0232
0233     SysTim1 := DWORD_TO_WORD(Systim.ulLow);
0234     SysTim2 := DWORD_TO_WORD(ROL(Systim.ulLow, 16));
0235     SysTim3 := DWORD_TO_WORD(Systim.ulHigh);
0236     SysTim4 := DWORD_TO_WORD(ROL(Systim.ulHigh, 16));
0237
0238     Drive11_write_table[d11 + 24] := SysTim1;
0239     Drive11_write_table[d11 + 25] := SysTim2;
0240     Drive11_write_table[d11 + 26] := SysTim3;
0241     Drive11_write_table[d11 + 27] := SysTim4;
0242
0243     drive11FaultNum := drive11FaultNum +1;
0244     d11 := d11 + 28;
0245 END_IF;
0246
```

```
0247 IF drive12Status.3 AND oldError12 = FALSE THEN
0248     Drive12_write_table[d12] := drive12FaultNum;
0249     Drive12_write_table[d12 + 1] := VersionNumber;
0250     Drive12_write_table[d12 + 2] := LoopNum1;
0251     Drive12_write_table[d12 + 3] := LoopNum2;
0252     Drive12_write_table[d12 + 4] := drive12Status;
0253     Drive12_write_table[d12 + 5] := drive12PSStatus;
0254     Drive12_write_table[d12 + 6] := drive12PSControl;
0255     Drive12_write_table[d12 + 7] := drive12PROFIsafeStatusActFV;
0256     Drive12_write_table[d12 + 8] := drive12PROFIsafeStatusCRC;
0257     Drive12_write_table[d12 + 9] := drive12PROFIsafeStatusConsnrR;
0258     Drive12_write_table[d12 + 10] := drive12PROFIsafeStatusDeviceFault;
0259     Drive12_write_table[d12 + 11] := drive12PROFIsafeStatusFVactivated;
0260     Drive12_write_table[d12 + 12] := drive12PROFIsafeStatusHostCRC;
0261     Drive12_write_table[d12 + 13] := drive12PROFIsafeStatusHostTimeout;
0262     Drive12_write_table[d12 + 14] := drive12PROFIsafeStatusiParEN;
0263     Drive12_write_table[d12 + 15] := drive12PROFIsafeStatusiParOK;
0264     Drive12_write_table[d12 + 16] := drive12PROFIsafeStatusOAC;
0265     Drive12_write_table[d12 + 17] := drive12PROFIsafeStatusOAReq;
0266     Drive12_write_table[d12 + 18] := drive12PROFIsafeStatusToggleD;
0267     Drive12_write_table[d12 + 19] := drive12PROFIsafeStatusResTim;
0268     Drive12_write_table[d12 + 20] := drive12PROFIsafeStatusWDTimeout;
0269     Drive12_write_table[d12 + 21] := drive12NewHighValue;
0270     Drive12_write_table[d12 + 22] := drive12OldHighValue;
0271     Drive12_write_table[d12 + 23] := drive12AlltimeHigh;
0272
0273     SysTim1 := DWORD_TO_WORD(Systim.ulLow);
0274     SysTim2 := DWORD_TO_WORD(ROL(Systim.ulLow, 16));
0275     SysTim3 := DWORD_TO_WORD(Systim.ulHigh);
0276     SysTim4 := DWORD_TO_WORD(ROL(Systim.ulHigh, 16));
0277
0278     Drive12_write_table[d12 + 24] := SysTim1;
0279     Drive12_write_table[d12 + 25] := SysTim2;
0280     Drive12_write_table[d12 + 26] := SysTim3;
0281     Drive12_write_table[d12 + 27] := SysTim4;
0282
0283     drive12FaultNum := drive12FaultNum +1;
0284     d12 := d12 + 28;
0285 END_IF;
0286
0287 IF drive13Status.3 AND oldError13 = FALSE THEN
0288     Drive13_write_table[d13] := drive13FaultNum;
0289     Drive13_write_table[d13 + 1] := VersionNumber;
0290     Drive13_write_table[d13 + 2] := LoopNum1;
0291     Drive13_write_table[d13 + 3] := LoopNum2;
0292     Drive13_write_table[d13 + 4] := drive13Status;
0293     Drive13_write_table[d13 + 5] := drive13PSStatus;
0294     Drive13_write_table[d13 + 6] := drive13PSControl;
0295     Drive13_write_table[d13 + 7] := drive13PROFIsafeStatusActFV;
0296     Drive13_write_table[d13 + 8] := drive13PROFIsafeStatusCRC;
0297     Drive13_write_table[d13 + 9] := drive13PROFIsafeStatusConsnrR;
0298     Drive13_write_table[d13 + 10] := drive13PROFIsafeStatusDeviceFault;
0299     Drive13_write_table[d13 + 11] := drive13PROFIsafeStatusFVactivated;
0300     Drive13_write_table[d13 + 12] := drive13PROFIsafeStatusHostCRC;
0301     Drive13_write_table[d13 + 13] := drive13PROFIsafeStatusHostTimeout;
0302     Drive13_write_table[d13 + 14] := drive13PROFIsafeStatusiParEN;
0303     Drive13_write_table[d13 + 15] := drive13PROFIsafeStatusiParOK;
0304     Drive13_write_table[d13 + 16] := drive13PROFIsafeStatusOAC;
0305     Drive13_write_table[d13 + 17] := drive13PROFIsafeStatusOAReq;
0306     Drive13_write_table[d13 + 18] := drive13PROFIsafeStatusToggleD;
0307     Drive13_write_table[d13 + 19] := drive13PROFIsafeStatusResTim;
0308     Drive13_write_table[d13 + 20] := drive13PROFIsafeStatusWDTimeout;
0309     Drive13_write_table[d13 + 21] := drive13NewHighValue;
0310     Drive13_write_table[d13 + 22] := drive13OldHighValue;
0311     Drive13_write_table[d13 + 23] := drive13AlltimeHigh;
0312
```

```
0313      SysTim1 := DWORD_TO_WORD(Systim.ulLow);
0314      SysTim2 := DWORD_TO_WORD(ROL(Systim.ulLow, 16));
0315      SysTim3 := DWORD_TO_WORD(Systim.ulHigh);
0316      SysTim4 := DWORD_TO_WORD(ROL(Systim.ulHigh, 16));
0317
0318      Drive13_write_table[d13 + 24] := SysTim1;
0319      Drive13_write_table[d13 + 25] := SysTim2;
0320      Drive13_write_table[d13 + 26] := SysTim3;
0321      Drive13_write_table[d13 + 27] := SysTim4;
0322
0323      drive13FaultNum := drive13FaultNum +1;
0324      d13 := d13 + 28;
0325  END_IF;
0326
0327  IF drive14Status.3 AND oldError14 = FALSE THEN
0328      Drive14_write_table[d14] := drive14FaultNum;
0329      Drive14_write_table[d14 + 1] := VersionNumber;
0330      Drive14_write_table[d14 + 2] := LoopNum1;
0331      Drive14_write_table[d14 + 3] := LoopNum2;
0332      Drive14_write_table[d14 + 4] := drive14Status;
0333      Drive14_write_table[d14 + 5] := drive14PSStatus;
0334      Drive14_write_table[d14 + 6] := drive14PSControl;
0335      Drive14_write_table[d14 + 7] := drive14PROFIsafeStatusActFV;
0336      Drive14_write_table[d14 + 8] := drive14PROFIsafeStatusCRC;
0337      Drive14_write_table[d14 + 9] := drive14PROFIsafeStatusConsnrR;
0338      Drive14_write_table[d14 + 10] := drive14PROFIsafeStatusDeviceFault;
0339      Drive14_write_table[d14 + 11] := drive14PROFIsafeStatusFVactivated;
0340      Drive14_write_table[d14 + 12] := drive14PROFIsafeStatusHostCRC;
0341      Drive14_write_table[d14 + 13] := drive14PROFIsafeStatusHostTimeout;
0342      Drive14_write_table[d14 + 14] := drive14PROFIsafeStatusiParEN;
0343      Drive14_write_table[d14 + 15] := drive14PROFIsafeStatusiParOK;
0344      Drive14_write_table[d14 + 16] := drive14PROFIsafeStatusOAC;
0345      Drive14_write_table[d14 + 17] := drive14PROFIsafeStatusOAReq;
0346      Drive14_write_table[d14 + 18] := drive14PROFIsafeStatusToggleD;
0347      Drive14_write_table[d14 + 19] := drive14PROFIsafeStatusResTim;
0348      Drive14_write_table[d14 + 20] := drive14PROFIsafeStatusWDTimeout;
0349      Drive14_write_table[d14 + 21] := drive14NewHighValue;
0350      Drive14_write_table[d14 + 22] := drive14OldHighValue;
0351      Drive14_write_table[d14 + 23] := drive14AlltimeHigh;
0352
0353      SysTim1 := DWORD_TO_WORD(Systim.ulLow);
0354      SysTim2 := DWORD_TO_WORD(ROL(Systim.ulLow, 16));
0355      SysTim3 := DWORD_TO_WORD(Systim.ulHigh);
0356      SysTim4 := DWORD_TO_WORD(ROL(Systim.ulHigh, 16));
0357
0358      Drive14_write_table[d14 + 24] := SysTim1;
0359      Drive14_write_table[d14 + 25] := SysTim2;
0360      Drive14_write_table[d14 + 26] := SysTim3;
0361      Drive14_write_table[d14 + 27] := SysTim4;
0362
0363      drive14FaultNum := drive14FaultNum +1;
0364      d14 := d14 + 28;
0365  END_IF;
0366
0367  oldError11 := drive11Status.3;
0368  oldError12 := drive12Status.3;
0369  oldError13 := drive13Status.3;
0370  oldError14 := drive14Status.3;
0371
0372  (* Reset fault*)
0373  IF drive11Status.3 AND oldError11 THEN
0374  drive11Command.7 := TRUE;
0375  END_IF;
0376
0377  IF drive12Status.3 AND oldError12 THEN
0378  drive12Command.7 := TRUE;
```

```
0379 END_IF;
0380
0381 IF drive13Status.3 AND oldError13 THEN
0382 drive13Command.7 := TRUE;
0383 END_IF;
0384
0385 IF drive14Status.3 AND oldError14 THEN
0386 drive14Command.7 := TRUE;
0387 END_IF;
0388
0389 (* Trigger for "All OK" table *)
0390 SysTimLog := systim.ulLow;
0391
0392 IF SysTimLog > SysTimTrigger THEN
0393     IF SysTimTrigger + 3600000000 > 4294967295 THEN
0394     SysTimTrigger := 3600000000;
0395     SysTimTriggerHigh := SysTimTriggerHigh +1;
0396     ELSE
0397     SysTimTrigger := SysTimTrigger + 3600000000;
0398     END_IF;
0399 WriteAllOK := TRUE;
0400 END_IF;
0401
0402 (* "All OK" table *)
0403 IF WriteAllOK = TRUE THEN
0404     AllOK_write_table[OK] := VersionNumber;
0405     AllOK_write_table[OK + 1] := LoopNum1;
0406     AllOK_write_table[OK + 2] := LoopNum2;
0407     AllOK_write_table[OK + 3] := 1111;
0408     AllOK_write_table[OK + 4] := drive11Status;
0409     AllOK_write_table[OK + 5] :=drive11PSStatus;
0410     AllOK_write_table[OK + 6] :=drive11PSControl;
0411     AllOK_write_table[OK + 7] :=drive11PROFIsafeStatusActFV;
0412     AllOK_write_table[OK + 8] :=drive11PROFIsafeStatusCRC;
0413     AllOK_write_table[OK + 9] :=drive11PROFIsafeStatusConsnrR;
0414     AllOK_write_table[OK + 10] :=drive11PROFIsafeStatusDeviceFault;
0415     AllOK_write_table[OK + 11] :=drive11PROFIsafeStatusFVactivated;
0416     AllOK_write_table[OK + 12] :=drive11PROFIsafeStatusHostCRC;
0417     AllOK_write_table[OK + 13] :=drive11PROFIsafeStatusHostTimeout;
0418     AllOK_write_table[OK + 14] :=drive11PROFIsafeStatusiParEN;
0419     AllOK_write_table[OK + 15] :=drive11PROFIsafeStatusiParOK;
0420     AllOK_write_table[OK + 16] :=drive11PROFIsafeStatusOAC;
0421     AllOK_write_table[OK + 17] :=drive11PROFIsafeStatusOAReq;
0422     AllOK_write_table[OK + 18] :=drive11PROFIsafeStatusToggleD;
0423     AllOK_write_table[OK + 19] :=drive11PROFIsafeStatusResTim;
0424     AllOK_write_table[OK + 20] :=drive11PROFIsafeStatusWDTimeout;
0425     AllOK_write_table[OK + 21] :=drive11NewHighValue;
0426     AllOK_write_table[OK + 22] :=drive11OldHighValue;
0427     AllOK_write_table[OK + 23] :=drive11AlltimeHigh;
0428     AllOK_write_table[OK + 24] :=1212;
0429     AllOK_write_table[OK + 25] :=drive12Status;
0430     AllOK_write_table[OK + 26] :=drive12PSStatus;;
0431     AllOK_write_table[OK + 27] :=drive12PSControl;
0432     AllOK_write_table[OK + 28] :=drive12PROFIsafeStatusActFV;
0433     AllOK_write_table[OK + 29] :=drive12PROFIsafeStatusCRC;
0434     AllOK_write_table[OK + 30] :=drive12PROFIsafeStatusConsnrR;
0435     AllOK_write_table[OK + 31] :=drive12PROFIsafeStatusDeviceFault;
0436     AllOK_write_table[OK + 32] :=drive12PROFIsafeStatusFVactivated;
0437     AllOK_write_table[OK + 33] :=drive12PROFIsafeStatusHostCRC;
0438     AllOK_write_table[OK + 34] :=drive12PROFIsafeStatusHostTimeout;
0439     AllOK_write_table[OK + 35] :=drive12PROFIsafeStatusiParEN;
0440     AllOK_write_table[OK + 36] :=drive12PROFIsafeStatusiParOK;
0441     AllOK_write_table[OK + 37] :=drive12PROFIsafeStatusOAC;
0442     AllOK_write_table[OK + 38] :=drive12PROFIsafeStatusOAReq;
0443     AllOK_write_table[OK + 39] :=drive12PROFIsafeStatusToggleD;
0444     AllOK_write_table[OK + 40] :=drive12PROFIsafeStatusResTim;
```

```
0445      AllOK_write_table[OK + 41] :=drive12PROFIsafeStatusWDTimeout;
0446      AllOK_write_table[OK + 42] :=drive12NewHighValue;
0447      AllOK_write_table[OK + 43] :=drive12OldHighValue;
0448      AllOK_write_table[OK + 44] :=drive12AlltimeHigh;
0449      AllOK_write_table[OK + 45] :=1313;
0450      AllOK_write_table[OK + 46] :=drive13Status;
0451      AllOK_write_table[OK + 47] :=drive13PSStatus;
0452      AllOK_write_table[OK + 48] :=drive13PSControl;
0453      AllOK_write_table[OK + 49] :=drive13PROFIsafeStatusActFV;
0454      AllOK_write_table[OK + 50] :=drive13PROFIsafeStatusCRC;
0455      AllOK_write_table[OK + 51] :=drive13PROFIsafeStatusConsnrR;
0456      AllOK_write_table[OK + 52] :=drive13PROFIsafeStatusDeviceFault;
0457      AllOK_write_table[OK + 53] :=drive13PROFIsafeStatusFVactivated;
0458      AllOK_write_table[OK + 54] :=drive13PROFIsafeStatusHostCRC;
0459      AllOK_write_table[OK + 55] :=drive13PROFIsafeStatusHostTimeout;
0460      AllOK_write_table[OK + 56] :=drive13PROFIsafeStatusiParEN;
0461      AllOK_write_table[OK + 57] :=drive13PROFIsafeStatusiParOK;
0462      AllOK_write_table[OK + 58] :=drive13PROFIsafeStatusOAC;
0463      AllOK_write_table[OK + 59] :=drive13PROFIsafeStatusOAReq;
0464      AllOK_write_table[OK + 60] :=drive13PROFIsafeStatusToggleD;
0465      AllOK_write_table[OK + 61] :=drive13PROFIsafeStatusResTim;
0466      AllOK_write_table[OK + 62] :=drive13PROFIsafeStatusWDTimeout;
0467      AllOK_write_table[OK + 63] :=drive13NewHighValue;
0468      AllOK_write_table[OK + 64] :=drive13OldHighValue;
0469      AllOK_write_table[OK + 65] :=drive13AlltimeHigh;
0470      AllOK_write_table[OK + 66] :=1414;
0471      AllOK_write_table[OK + 67] :=drive14Status;
0472      AllOK_write_table[OK + 68] :=drive14PSStatus;
0473      AllOK_write_table[OK + 69] :=drive14PSControl;
0474      AllOK_write_table[OK + 70] :=drive14PROFIsafeStatusActFV;
0475      AllOK_write_table[OK + 71] :=drive14PROFIsafeStatusCRC;
0476      AllOK_write_table[OK + 72] :=drive14PROFIsafeStatusConsnrR;
0477      AllOK_write_table[OK + 73] :=drive14PROFIsafeStatusDeviceFault;
0478      AllOK_write_table[OK + 74] :=drive14PROFIsafeStatusFVactivated;
0479      AllOK_write_table[OK + 75] :=drive14PROFIsafeStatusHostCRC;
0480      AllOK_write_table[OK + 76] :=drive14PROFIsafeStatusHostTimeout;
0481      AllOK_write_table[OK + 77] :=drive14PROFIsafeStatusiParEN;
0482      AllOK_write_table[OK + 78] :=drive14PROFIsafeStatusiParOK;
0483      AllOK_write_table[OK + 79] :=drive14PROFIsafeStatusOAC;
0484      AllOK_write_table[OK + 80] :=drive14PROFIsafeStatusOAReq;
0485      AllOK_write_table[OK + 81] :=drive14PROFIsafeStatusToggleD;
0486      AllOK_write_table[OK + 82] :=drive14PROFIsafeStatusResTim;
0487      AllOK_write_table[OK + 83] :=drive14PROFIsafeStatusWDTimeout;
0488      AllOK_write_table[OK + 84] :=drive14NewHighValue;
0489      AllOK_write_table[OK + 85] :=drive14OldHighValue;
0490      AllOK_write_table[OK + 86] :=drive14AlltimeHigh;
0491
0492      WriteAllOK := FALSE;
0493      OK := OK + 87;
0494  END_IF;
0495
0496  Write_d11(     WriteDone=> WriteDone11);
0497  Write_d12(WriteDone=> WriteDone12);
0498  Write_d13(WriteDone=> WriteDone13);
0499  Write_d14(WriteDone=> WriteDone14);
0500  Write_OK(WriteDone=> WriteDoneOK);
0501  Write_d11(     WriteError=> WriteError11);
0502  Write_d12(WriteError=> WriteError12);
0503  Write_d13(WriteError=> WriteError13);
0504  Write_d14(WriteError=> WriteError14);
0505  Write_OK(WriteError=> WriteErrorOK);
0506
0507  (* WRITE TABLES TO SD CARD *)
0508  IF d11 = 85 THEN
0509  Write_d11(
0510      oldFailStatus:= ,
```

```
0511      startCommand:= ,
0512      sDevName:= ,
0513      SD_Write11:= TRUE,
0514      WriteDone=> );
0515      IF WriteDone11 OR  WriteError11 THEN
0516      d11 := 1;
0517      END_IF;
0518  END_IF;
0519
0520  IF d12 = 85 THEN
0521  Write_d12(
0522      oldFailStatus:= ,
0523      startCommand:= ,
0524      sDevName:= ,
0525      SD_Write12:= TRUE,
0526      WriteDone=> );
0527      IF WriteDone12 OR WriteError12 THEN
0528      d12 := 1;
0529      END_IF;
0530  END_IF;
0531
0532  IF d13 = 85 THEN
0533  Write_d13(
0534      oldFailStatus:= ,
0535      startCommand:= ,
0536      sDevName:= ,
0537      SD_Write13:= TRUE,
0538      WriteDone=> );
0539      IF WriteDone13 OR WriteError13 THEN
0540      d13 := 1;
0541      END_IF;
0542  END_IF;
0543
0544  IF d14 = 85 THEN
0545  Write_d14(
0546      oldFailStatus:= ,
0547      startCommand:= ,
0548      sDevName:= ,
0549      SD_Write14:= TRUE,
0550      WriteDone=> );
0551      IF WriteDone14 OR WriteError14 THEN
0552      d14 := 1;
0553      END_IF;
0554  END_IF;
0555
0556  IF OK = 349 THEN
0557  Write_OK(
0558      oldFailStatus:= ,
0559      startCommand:= ,
0560      sDevName:= ,
0561      SD_WriteOK:= TRUE,
0562      WriteDone=> );
0563      IF WriteDoneOK OR WriteErrorOK THEN
0564      OK := 1;
0565      END_IF;
0566  END_IF;
```

Turvaohjelman pääohjelma

```
0001    PROGRAM PLC_PRG
0002 VAR
0003    fbCyclicNoneSafeRec             : SF_CYCLIC_PM5XX_S_REC;
0004    fbCyclicNoneSafeSend            : SF_CYCLIC_PM5XX_S_SEND;
0005
0006    SF_WDOG:SF_WDOG_TIME_SET;
0007    ProgTimer: TIME := T#0ms;
0008    StateCheck: INT := 1;
0009    StateChangeTime: TIME := T#5000ms;
0010    drive11VarSLSLimit: INT := 1000;
0011
0012    drive11SLS2reqSend: WORD;
0013    drive11SLS1reqSend: WORD;
0014    drive11SS1reqSend: WORD;
0015    drive11SSEreqSend: WORD;
0016    drive11POUSreqSend: WORD;
0017    drive11STOreqSend: WORD;
0018    drive11SLS4reqSend: WORD;
0019    drive11SLS3reqSend: WORD;
0020    drive11VarSLSreqSend: WORD;
0021    drive11NegScaleSend: WORD;
0022    drive11PosScaleSend: WORD;
0023    ProgTimerSend: WORD;
0024    drive11STOactSend: WORD;
0025    LoopNum: DWORD := 0;
0026    LoopNumSend: WORD;
0027    LoopNumSend2: WORD;
0028    drive11PROFIsafeStatusResTimSend: WORD;
0029    drive11PROFIsafeStatusCRCSend: WORD;
0030    drive11PROFIsafeStatusConsnrRSend: WORD;
0031    drive11PROFIsafeStatusDeviceFaultSend: WORD;
0032    drive11PROFIsafeStatusFVactivatedSend: WORD;
0033    drive11PROFIsafeStatusHostCRCSend: WORD;
0034    drive11PROFIsafeStatusHostTimeoutSend: WORD;
0035    drive11PROFIsafeStatusiParENSend: WORD;
0036    drive11PROFIsafeStatusiParOKSend: WORD;
0037    drive11PROFIsafeStatusOACSend: WORD;
0038    drive11PROFIsafeStatusOAReqSend: WORD;
0039    drive11PROFIsafeStatusToggleDSend: WORD;
0040    drive11PROFIsafeStatusWDTimeoutSend: WORD;
0041    drive11PROFIsafeStatusActFVSend: WORD;
0042    drive12PROFIsafeStatusResTimSend: WORD;
0043    drive12PROFIsafeStatusCRCSend: WORD;
0044    drive12PROFIsafeStatusConsnrRSend: WORD;
0045    drive12PROFIsafeStatusDeviceFaultSend: WORD;
0046    drive12PROFIsafeStatusFVactivatedSend: WORD;
0047    drive12PROFIsafeStatusHostCRCSend: WORD;
0048    drive12PROFIsafeStatusHostTimeoutSend: WORD;
0049    drive12PROFIsafeStatusiParENSend: WORD;
0050    drive12PROFIsafeStatusiParOKSend: WORD;
0051    drive12PROFIsafeStatusOACSend: WORD;
0052    drive12PROFIsafeStatusOAReqSend: WORD;
0053    drive12PROFIsafeStatusToggleDSend: WORD;
0054    drive12PROFIsafeStatusWDTimeoutSend: WORD;
0055    drive12PROFIsafeStatusActFVSend: WORD;
0056    drive13PROFIsafeStatusResTimSend: WORD;
0057    drive13PROFIsafeStatusCRCSend: WORD;
0058    drive13PROFIsafeStatusConsnrRSend: WORD;
0059    drive13PROFIsafeStatusDeviceFaultSend: WORD;
0060    drive13PROFIsafeStatusFVactivatedSend: WORD;
0061    drive13PROFIsafeStatusHostCRCSend: WORD;
0062    drive13PROFIsafeStatusHostTimeoutSend: WORD;
0063    drive13PROFIsafeStatusiParENSend: WORD;
0064    drive13PROFIsafeStatusiParOKSend: WORD;
0065    drive13PROFIsafeStatusOACSend: WORD;
0066    drive13PROFIsafeStatusOAReqSend: WORD;
```

```
0067      drive13PROFIsafeStatusToggleDSend: WORD;
0068      drive13PROFIsafeStatusWDTimeoutSend: WORD;
0069      drive13PROFIsafeStatusActFVSend: WORD;
0070      drive14PROFIsafeStatusResTimSend: WORD;
0071      drive14PROFIsafeStatusCRCSend: WORD;
0072      drive14PROFIsafeStatusConsnrRSend: WORD;
0073      drive14PROFIsafeStatusDeviceFaultSend: WORD;
0074      drive14PROFIsafeStatusFVactivatedSend: WORD;
0075      drive14PROFIsafeStatusHostCRCSend: WORD;
0076      drive14PROFIsafeStatusHostTimeoutSend: WORD;
0077      drive14PROFIsafeStatusiParENSend: WORD;
0078      drive14PROFIsafeStatusiParOKSend: WORD;
0079      drive14PROFIsafeStatusOACSend: WORD;
0080      drive14PROFIsafeStatusOAReqSend: WORD;
0081      drive14PROFIsafeStatusToggleDSend: WORD;
0082      drive14PROFIsafeStatusWDTimeoutSend: WORD;
0083      drive14PROFIsafeStatusActFVSend: WORD;
0084
0085  END_VAR
```

```
0001  (* Toggle Watchdog *)
0002  SF_WDOG(
0003    EN:=TRUE,
0004    WDOG:=500,
0005    RESET:=FALSE,
0006    DONE=>,
0007    ACT_TIME=>,
0008    MAX_TIME=>);
0009
0010  (* Cyclic data exchange receive *)
0011  fbCyclicNoneSafeRec(
0012      EN:=TRUE ,
0013      DATA:= ADR(G_awCyclicNoneSafe_Rec),
0014      DATA_LEN:= SIZEOF(G_awCyclicNoneSafe_Rec),
0015      DONE=> ,
0016      ERR=> ,
0017      ERNO=> );
0018
0019  (* Automatic  operator acknowledge for PROFIsafe *)
0020  IF drive11_ABB_PS1.OA_Req_S = TRUE THEN
0021      drive11_ABB_PS1.OA_C := TRUE;
0022  ELSE
0023      drive11_ABB_PS1.OA_C := FALSE;
0024  END_IF
0025  IF drive12_ABB_PS1.OA_Req_S = TRUE THEN
0026      drive12_ABB_PS1.OA_C := TRUE;
0027  ELSE
0028      drive12_ABB_PS1.OA_C := FALSE;
0029  END_IF
0030  IF drive13_ABB_PS1.OA_Req_S = TRUE THEN
0031      drive13_ABB_PS1.OA_C := TRUE;
0032  ELSE
0033      drive13_ABB_PS1.OA_C := FALSE;
0034  END_IF
0035  IF drive14_ABB_PS1.OA_Req_S = TRUE THEN
0036      drive14_ABB_PS1.OA_C := TRUE;
0037  ELSE
0038      drive14_ABB_PS1.OA_C := FALSE;
0039  END_IF
0040
0041
0042  (* Automatic acknowledge for FSO-11 *)
0043  IF drive11_SF_end_ack_req = TRUE THEN
0044      drive11_SF_end_ack := TRUE;
0045  ELSE
0046      drive11_SF_end_ack := FALSE;
0047  END_IF
```

```
0048 IF drive12_SF_end_ack_req = TRUE THEN
0049     drive12_SF_end_ack := TRUE;
0050 ELSE
0051     drive12_SF_end_ack := FALSE;
0052 END_IF
0053 IF drive13_SF_end_ack_req = TRUE THEN
0054     drive13_SF_end_ack := TRUE;
0055 ELSE
0056     drive13_SF_end_ack := FALSE;
0057 END_IF
0058 IF drive14_SF_end_ack_req = TRUE THEN
0059     drive14_SF_end_ack := TRUE;
0060 ELSE
0061     drive14_SF_end_ack := FALSE;
0062 END_IF
0063
0064 (* State program where safety functions are activated *)
0065 ProgTimer := ProgTimer + T#1ms;
0066
0067 IF StateCheck > 0 AND ProgTimer > StateChangeTime THEN
0068     ProgTimer := T#0s;
0069     IF StateCheck = 1 THEN (* all on*)
0070     LoopNum := LoopNum + 1;
0071     drive11SLS2req := TRUE;
0072     drive11SLS1req := TRUE;
0073     drive11SS1req := TRUE;
0074     drive11SSEreq := TRUE;
0075     drive11POUSreq := FALSE;
0076     drive11STOreq := TRUE;
0077     drive11SLS4req := TRUE;
0078     drive11SLS3req := TRUE;
0079     drive11VarSLSreq := TRUE;
0080     drive11NegScale := TRUE;
0081     drive11PosScale := TRUE;
0082     StateChangeTime := T#5000ms;
0083     ELSIF StateCheck = 2 THEN (* POUS*)
0084     drive11SLS2req := FALSE;
0085     drive11SLS1req := FALSE;
0086     drive11SS1req := FALSE;
0087     drive11SSEreq := FALSE;
0088     drive11POUSreq := TRUE;
0089     drive11STOreq := FALSE;
0090     drive11SLS4req := FALSE;
0091     drive11SLS3req := FALSE;
0092     drive11VarSLSreq := FALSE;
0093     drive11NegScale := FALSE;
0094     drive11PosScale := FALSE;
0095     StateChangeTime := T#5000ms;
0096     ELSIF StateCheck = 3 THEN (* all off*)
0097     drive11SLS2req := FALSE ;
0098     drive11SLS1req := FALSE;
0099     drive11SS1req := FALSE;
0100     drive11SSEreq := FALSE;
0101     drive11POUSreq := FALSE;
0102     drive11STOreq := FALSE;
0103     drive11SLS4req := FALSE;
0104     drive11SLS3req := FALSE;
0105     drive11VarSLSreq := FALSE;
0106     drive11NegScale := FALSE;
0107     drive11PosScale := FALSE;
0108     StateChangeTime := T#5000ms;
0109     ELSIF StateCheck = 4 THEN (* SLS2*)
0110     drive11SLS2req := TRUE ;
0111     drive11SLS1req := FALSE;
0112     drive11SS1req := FALSE;
0113     drive11SSEreq := FALSE;
```

```
0114    drive11POUSreq := FALSE;
0115    drive11STOreq := FALSE;
0116    drive11SLS4req := FALSE;
0117    drive11SLS3req := FALSE;
0118    drive11VarSLSreq := FALSE;
0119    drive11NegScale := FALSE;
0120    drive11PosScale := FALSE;
0121    StateChangeTime := T#10000ms;
0122    ELSIF StateCheck = 5 THEN (* SS1*)
0123    drive11SLS2req := FALSE ;
0124    drive11SLS1req := FALSE;
0125    drive11SS1req := TRUE;
0126    drive11SSEreq := FALSE;
0127    drive11POUSreq := FALSE;
0128    drive11STOreq := FALSE;
0129    drive11SLS4req := FALSE;
0130    drive11SLS3req := FALSE;
0131    drive11VarSLSreq := FALSE;
0132    drive11NegScale := FALSE;
0133    drive11PosScale := FALSE;
0134    StateChangeTime := T#10000ms;
0135    ELSIF StateCheck = 6 THEN (* SLS1*)
0136    drive11SLS2req := FALSE ;
0137    drive11SLS1req := TRUE;
0138    drive11SS1req := FALSE;
0139    drive11SSEreq := FALSE;
0140    drive11POUSreq := FALSE;
0141    drive11STOreq := FALSE;
0142    drive11SLS4req := FALSE;
0143    drive11SLS3req := FALSE;
0144    drive11VarSLSreq := FALSE;
0145    drive11NegScale := FALSE;
0146    drive11PosScale := FALSE;
0147    StateChangeTime := T#10000ms;
0148    ELSIF StateCheck = 7 THEN (* SLS3*)
0149    drive11SLS2req := FALSE ;
0150    drive11SLS1req := FALSE;
0151    drive11SS1req := FALSE;
0152    drive11SSEreq := FALSE;
0153    drive11POUSreq := FALSE;
0154    drive11STOreq := FALSE;
0155    drive11SLS4req := FALSE;
0156    drive11SLS3req := TRUE;
0157    drive11VarSLSreq := FALSE;
0158    drive11NegScale := FALSE;
0159    drive11PosScale := FALSE;
0160    StateChangeTime := T#10000ms;
0161    ELSIF StateCheck = 8 THEN (* SLS4*)
0162    drive11SLS2req := FALSE ;
0163    drive11SLS1req := FALSE;
0164    drive11SS1req := FALSE;
0165    drive11SSEreq := FALSE;
0166    drive11POUSreq := FALSE;
0167    drive11STOreq := FALSE;
0168    drive11SLS4req := TRUE;
0169    drive11SLS3req := FALSE;
0170    drive11VarSLSreq := FALSE;
0171    drive11NegScale := FALSE;
0172    drive11PosScale := FALSE;
0173    StateChangeTime := T#10000ms;
0174    ELSIF StateCheck = 9 THEN (* all SLS simultaneously*)
0175    drive11SLS2req := TRUE ;
0176    drive11SLS1req := TRUE;
0177    drive11SS1req := FALSE;
0178    drive11SSEreq := FALSE;
0179    drive11POUSreq := FALSE;
```

```
0180      drive11STOreq := FALSE;
0181      drive11SLS4req := TRUE;
0182      drive11SLS3req := TRUE;
0183      drive11VarSLSreq := FALSE;
0184      drive11NegScale := FALSE;
0185      drive11PosScale := FALSE;
0186      StateChangeTime := T#10000ms;
0187      ELSIF StateCheck = 10 THEN (* Variable SLS*)
0188      StateCheck := 0; (* return back to the start *)
0189      drive11SLS2req := FALSE ;
0190      drive11SLS1req := FALSE;
0191      drive11SS1req := FALSE;
0192      drive11SSEreq := FALSE;
0193      drive11POUSreq := FALSE;
0194      drive11STOreq := FALSE;
0195      drive11SLS4req := FALSE;
0196      drive11SLS3req := FALSE;
0197      drive11VarSLSreq := TRUE;
0198      drive11NegScale := FALSE;
0199      drive11PosScale := FALSE;
0200      StateChangeTime := T#10000ms;
0201      drive11VarSLSLimit := drive11VarSLSLimit+1000; (* add 1000 to reference*)
0202      END_IF
0203
0204      StateCheck := StateCheck+1; (* jump to next state*)
0205  END_IF
0206
0207  IF drive11VarSLSLimit = 15000 THEN (* when 15000 reached, back to 1000*)
0208      drive11VarSLSLimit := 1000;
0209  END_IF
0210
0211  (* Copy to all drives *)
0212  drive12SLS2req          := drive11SLS2req;
0213  drive12SLS1req          := drive11SLS1req;
0214  drive12SS1req           := drive11SS1req;
0215  drive12SSEreq           := drive11SSEreq;
0216  drive12POUSreq          := drive11POUSreq;
0217  drive12STOreq           := drive11STOreq;
0218  drive12SLS4req          := drive11SLS4req;
0219  drive12SLS3req          := drive11SLS3req;
0220  drive12VarSLSreq         := drive11VarSLSreq;
0221  drive12NegScale           := drive11NegScale;
0222  drive12PosScale          := drive11PosScale;
0223  drive12VarSLSLimit       := drive11VarSLSLimit;
0224
0225  drive13SLS2req          := drive11SLS2req;
0226  drive13SLS1req          := drive11SLS1req;
0227  drive13SS1req           := drive11SS1req;
0228  drive13SSEreq           := drive11SSEreq;
0229  drive13POUSreq          := drive11POUSreq;
0230  drive13STOreq           := drive11STOreq;
0231  drive13SLS4req          := drive11SLS4req;
0232  drive13SLS3req          := drive11SLS3req;
0233  drive13VarSLSreq          := drive11VarSLSreq;
0234  drive13NegScale           := drive11NegScale;
0235  drive13PosScale         := drive11PosScale;
0236  drive13VarSLSLimit        := drive11VarSLSLimit;
0237
0238  drive14SLS2req          := drive11SLS2req;
0239  drive14SLS1req          := drive11SLS1req;
0240  drive14SS1req           := drive11SS1req;
0241  drive14SSEreq           := drive11SSEreq;
0242  drive14POUSreq          := drive11POUSreq;
0243  drive14STOreq           := drive11STOreq;
0244  drive14SLS4req          := drive11SLS4req;
0245  drive14SLS3req          := drive11SLS3req;
```

```
0246 drive14VarSLSreq                := drive11VarSLSreq;
0247 drive14NegScale                 := drive11NegScale;
0248 drive14PosScale                 := drive11PosScale;
0249 drive14VarSLSLimit              := drive11VarSLSLimit;
0250
0251
0252 (* Change to WORD for writing to memory *)
0253 drive11SLS2reqSend := BOOL_TO_WORD(drive11SLS2req);
0254 drive11SLS1reqSend := BOOL_TO_WORD(drive11SLS1req);
0255 drive11SS1reqSend := BOOL_TO_WORD(drive11SS1req);
0256 drive11SSEreqSend := BOOL_TO_WORD(drive11SSEreq);
0257 drive11POUSreqSend := BOOL_TO_WORD(drive11POUSreq);
0258 drive11STOreqSend := BOOL_TO_WORD(drive11STOreq);
0259 drive11SLS4reqSend := BOOL_TO_WORD(drive11SLS4req);
0260 drive11SLS3reqSend := BOOL_TO_WORD(drive11SLS3req);
0261 drive11VarSLSreqSend := BOOL_TO_WORD(drive11VarSLSreq);
0262 drive11NegScaleSend := BOOL_TO_WORD(drive11NegScale);
0263 drive11PosScaleSend := BOOL_TO_WORD(drive11PosScale);
0264 ProgTimerSend := TIME_TO_WORD(ProgTimer);
0265 drive11STOactSend := BOOL_TO_WORD(drive11STOact);
0266 LoopNumSend := DWORD_TO_WORD(LoopNum);
0267 LoopNumSend2 := DWORD_TO_WORD(ROL(LoopNum, 16));
0268
0269 drive11PROFIsafeStatusActFVSend := BOOL_TO_WORD(drive11_ABB_PS1.activate_FV_C);
0270 drive11PROFIsafeStatusCRCSend := BOOL_TO_WORD(drive11_ABB_PS1.CE_CRC);
0271 drive11PROFIsafeStatusConsnrRSend := BOOL_TO_WORD(drive11_ABB_PS1.cons_nr_R);
0272 drive11PROFIsafeStatusDeviceFaultSend := BOOL_TO_WORD(drive11_ABB_PS1.Device_Fault);
0273 drive11PROFIsafeStatusFVactivatedSend := BOOL_TO_WORD(drive11_ABB_PS1.FV_activated_S);
0274 drive11PROFIsafeStatusHostCRCSend := BOOL_TO_WORD(drive11_ABB_PS1.Host_CE_CRC);
0275 drive11PROFIsafeStatusHostTimeoutSend := BOOL_TO_WORD(drive11_ABB_PS1.HostTimeout);
0276 drive11PROFIsafeStatusiParENSend := BOOL_TO_WORD(drive11_ABB_PS1.iPar_EN_C);
0277 drive11PROFIsafeStatusiParOKSend := BOOL_TO_WORD(drive11_ABB_PS1.iPar_OK_S);
0278 drive11PROFIsafeStatusOACSend := BOOL_TO_WORD(drive11_ABB_PS1.OA_C);
0279 drive11PROFIsafeStatusOAReqSend := BOOL_TO_WORD(drive11_ABB_PS1.OA_Req_S);
0280 drive11PROFIsafeStatusToggleDSend := BOOL_TO_WORD(drive11_ABB_PS1.Toggle_d);
0281 drive11PROFIsafeStatusResTimSend := TIME_TO_WORD(drive11_ABB_PS1.tResponseTimeMS);
0282 drive11PROFIsafeStatusWDTimeoutSend := BOOL_TO_WORD(drive11_ABB_PS1.WD_timeout);
0283 drive12PROFIsafeStatusActFVSend := BOOL_TO_WORD(drive12_ABB_PS1.activate_FV_C);
0284 drive12PROFIsafeStatusCRCSend := BOOL_TO_WORD(drive12_ABB_PS1.CE_CRC);
0285 drive12PROFIsafeStatusConsnrRSend := BOOL_TO_WORD(drive12_ABB_PS1.cons_nr_R);
0286 drive12PROFIsafeStatusDeviceFaultSend := BOOL_TO_WORD(drive12_ABB_PS1.Device_Fault);
0287 drive12PROFIsafeStatusFVactivatedSend := BOOL_TO_WORD(drive12_ABB_PS1.FV_activated_S);
0288 drive12PROFIsafeStatusHostCRCSend := BOOL_TO_WORD(drive12_ABB_PS1.Host_CE_CRC);
0289 drive12PROFIsafeStatusHostTimeoutSend := BOOL_TO_WORD(drive12_ABB_PS1.HostTimeout);
0290 drive12PROFIsafeStatusiParENSend := BOOL_TO_WORD(drive12_ABB_PS1.iPar_EN_C);
0291 drive12PROFIsafeStatusiParOKSend := BOOL_TO_WORD(drive12_ABB_PS1.iPar_OK_S);
0292 drive12PROFIsafeStatusOACSend := BOOL_TO_WORD(drive12_ABB_PS1.OA_C);
0293 drive12PROFIsafeStatusOAReqSend := BOOL_TO_WORD(drive12_ABB_PS1.OA_Req_S);
0294 drive12PROFIsafeStatusToggleDSend := BOOL_TO_WORD(drive12_ABB_PS1.Toggle_d);
0295 drive12PROFIsafeStatusResTimSend := TIME_TO_WORD(drive12_ABB_PS1.tResponseTimeMS);
0296 drive12PROFIsafeStatusWDTimeoutSend := BOOL_TO_WORD(drive12_ABB_PS1.WD_timeout);
0297 drive13PROFIsafeStatusActFVSend := BOOL_TO_WORD(drive13_ABB_PS1.activate_FV_C);
0298 drive13PROFIsafeStatusCRCSend := BOOL_TO_WORD(drive13_ABB_PS1.CE_CRC);
0299 drive13PROFIsafeStatusConsnrRSend := BOOL_TO_WORD(drive13_ABB_PS1.cons_nr_R);
0300 drive13PROFIsafeStatusDeviceFaultSend := BOOL_TO_WORD(drive13_ABB_PS1.Device_Fault);
0301 drive13PROFIsafeStatusFVactivatedSend := BOOL_TO_WORD(drive13_ABB_PS1.FV_activated_S);
0302 drive13PROFIsafeStatusHostCRCSend := BOOL_TO_WORD(drive13_ABB_PS1.Host_CE_CRC);
0303 drive13PROFIsafeStatusHostTimeoutSend := BOOL_TO_WORD(drive13_ABB_PS1.HostTimeout);
0304 drive13PROFIsafeStatusiParENSend := BOOL_TO_WORD(drive13_ABB_PS1.iPar_EN_C);
0305 drive13PROFIsafeStatusiParOKSend := BOOL_TO_WORD(drive13_ABB_PS1.iPar_OK_S);
0306 drive13PROFIsafeStatusOACSend := BOOL_TO_WORD(drive13_ABB_PS1.OA_C);
0307 drive13PROFIsafeStatusOAReqSend := BOOL_TO_WORD(drive13_ABB_PS1.OA_Req_S);
0308 drive13PROFIsafeStatusToggleDSend := BOOL_TO_WORD(drive13_ABB_PS1.Toggle_d);
0309 drive13PROFIsafeStatusResTimSend := TIME_TO_WORD(drive13_ABB_PS1.tResponseTimeMS);
0310 drive13PROFIsafeStatusWDTimeoutSend := BOOL_TO_WORD(drive13_ABB_PS1.WD_timeout);
0311 drive14PROFIsafeStatusActFVSend := BOOL_TO_WORD(drive14_ABB_PS1.activate_FV_C);
```

```
0312 drive14PROFIsafeStatusCRCSend := BOOL_TO_WORD(drive14_ABB_PS1.CE_CRC);
0313 drive14PROFIsafeStatusConsnrRSend := BOOL_TO_WORD(drive14_ABB_PS1.cons_nr_R);
0314 drive14PROFIsafeStatusDeviceFaultSend := BOOL_TO_WORD(drive14_ABB_PS1.Device_Fault);
0315 drive14PROFIsafeStatusFVactivatedSend := BOOL_TO_WORD(drive14_ABB_PS1.FV_activated_S);
0316 drive14PROFIsafeStatusHostCRCSend := BOOL_TO_WORD(drive14_ABB_PS1.Host_CE_CRC);
0317 drive14PROFIsafeStatusHostTimeoutSend := BOOL_TO_WORD(drive14_ABB_PS1.HostTimeout);
0318 drive14PROFIsafeStatusiParENSend := BOOL_TO_WORD(drive14_ABB_PS1.iPar_EN_C);
0319 drive14PROFIsafeStatusiParOKSend := BOOL_TO_WORD(drive14_ABB_PS1.iPar_OK_S);
0320 drive14PROFIsafeStatusOACSend := BOOL_TO_WORD(drive14_ABB_PS1.OA_C);
0321 drive14PROFIsafeStatusOAReqSend := BOOL_TO_WORD(drive14_ABB_PS1.OA_Req_S);
0322 drive14PROFIsafeStatusToggleDSend := BOOL_TO_WORD(drive14_ABB_PS1.Toggle_d);
0323 drive14PROFIsafeStatusResTimSend := TIME_TO_WORD(drive14_ABB_PS1.tResponseTimeMS);
0324 drive14PROFIsafeStatusWDTimeoutSend := BOOL_TO_WORD(drive14_ABB_PS1.WD_timeout);
0325
0326 (* Write to memory for data exchange send *)
0327 G_awCyclicNoneSafe_Send[1] := StateCheck;
0328 G_awCyclicNoneSafe_Send[2] := drive11SLS2reqSend;
0329 G_awCyclicNoneSafe_Send[3] := drive11SLS1reqSend;
0330 G_awCyclicNoneSafe_Send[4] := drive11SS1reqSend;
0331 G_awCyclicNoneSafe_Send[5] := drive11SSEreqSend;
0332 G_awCyclicNoneSafe_Send[6] := drive11POUSreqSend;
0333 G_awCyclicNoneSafe_Send[7] := drive11STOreqSend;
0334 G_awCyclicNoneSafe_Send[8] := drive11SLS4reqSend;
0335 G_awCyclicNoneSafe_Send[9] := drive11SLS3reqSend;
0336 G_awCyclicNoneSafe_Send[10] := drive11VarSLSreqSend;
0337 G_awCyclicNoneSafe_Send[11] := drive11NegScaleSend;
0338 G_awCyclicNoneSafe_Send[12] := drive11PosScaleSend;
0339 G_awCyclicNoneSafe_Send[13] := drive11VarSLSLimit;
0340 G_awCyclicNoneSafe_Send[14] := ProgTimerSend;
0341 G_awCyclicNoneSafe_Send[15] := drive11STOactSend;
0342 G_awCyclicNoneSafe_Send[16] := LoopNumSend;
0343 G_awCyclicNoneSafe_Send[17] := LoopNumSend2;
0344 G_awCyclicNoneSafe_Send[18] := drive11PROFIsafeStatusActFVSend;
0345 G_awCyclicNoneSafe_Send[19] := drive11PROFIsafeStatusCRCSend;
0346 G_awCyclicNoneSafe_Send[20] := drive11PROFIsafeStatusConsnrRSend;
0347 G_awCyclicNoneSafe_Send[21] := drive11PROFIsafeStatusDeviceFaultSend;
0348 G_awCyclicNoneSafe_Send[22] := drive11PROFIsafeStatusFVactivatedSend;
0349 G_awCyclicNoneSafe_Send[23] := drive11PROFIsafeStatusHostCRCSend;
0350 G_awCyclicNoneSafe_Send[24] := drive11PROFIsafeStatusHostTimeoutSend;
0351 G_awCyclicNoneSafe_Send[25] := drive11PROFIsafeStatusiParENSend;
0352 G_awCyclicNoneSafe_Send[26] := drive11PROFIsafeStatusiParOKSend;
0353 G_awCyclicNoneSafe_Send[27] := drive11PROFIsafeStatusOACSend;
0354 G_awCyclicNoneSafe_Send[28] := drive11PROFIsafeStatusOAReqSend;
0355 G_awCyclicNoneSafe_Send[29] := drive11PROFIsafeStatusToggleDSend;
0356 G_awCyclicNoneSafe_Send[30] := drive11PROFIsafeStatusResTimSend;
0357 G_awCyclicNoneSafe_Send[31] := drive11PROFIsafeStatusWDTimeoutSend;
0358 G_awCyclicNoneSafe_Send[32] := drive12PROFIsafeStatusActFVSend;
0359 G_awCyclicNoneSafe_Send[33] := drive12PROFIsafeStatusCRCSend;
0360 G_awCyclicNoneSafe_Send[34] := drive12PROFIsafeStatusConsnrRSend;
0361 G_awCyclicNoneSafe_Send[35] := drive12PROFIsafeStatusDeviceFaultSend;
0362 G_awCyclicNoneSafe_Send[36] := drive12PROFIsafeStatusFVactivatedSend;
0363 G_awCyclicNoneSafe_Send[37] := drive12PROFIsafeStatusHostCRCSend;
0364 G_awCyclicNoneSafe_Send[38] := drive12PROFIsafeStatusHostTimeoutSend;
0365 G_awCyclicNoneSafe_Send[39] := drive12PROFIsafeStatusiParENSend;
0366 G_awCyclicNoneSafe_Send[40] := drive12PROFIsafeStatusiParOKSend;
0367 G_awCyclicNoneSafe_Send[41] := drive12PROFIsafeStatusOACSend;
0368 G_awCyclicNoneSafe_Send[42] := drive12PROFIsafeStatusOAReqSend;
0369 G_awCyclicNoneSafe_Send[43] := drive12PROFIsafeStatusToggleDSend;
0370 G_awCyclicNoneSafe_Send[44] := drive12PROFIsafeStatusResTimSend;
0371 G_awCyclicNoneSafe_Send[45] := drive12PROFIsafeStatusWDTimeoutSend;
0372 G_awCyclicNoneSafe_Send[46] := drive13PROFIsafeStatusActFVSend;
0373 G_awCyclicNoneSafe_Send[47] := drive13PROFIsafeStatusCRCSend;
0374 G_awCyclicNoneSafe_Send[48] := drive13PROFIsafeStatusConsnrRSend;
0375 G_awCyclicNoneSafe_Send[49] := drive13PROFIsafeStatusDeviceFaultSend;
0376 G_awCyclicNoneSafe_Send[50] := drive13PROFIsafeStatusFVactivatedSend;
0377 G_awCyclicNoneSafe_Send[51] := drive13PROFIsafeStatusHostCRCSend;
```

```
0378 G_awCyclicNoneSafe_Send[52] := drive13PROFIsafeStatusHostTimeoutSend;
0379 G_awCyclicNoneSafe_Send[53] := drive13PROFIsafeStatusiParENSend;
0380 G_awCyclicNoneSafe_Send[54] := drive13PROFIsafeStatusiParOKSend;
0381 G_awCyclicNoneSafe_Send[55] := drive13PROFIsafeStatusOACSend;
0382 G_awCyclicNoneSafe_Send[56] := drive13PROFIsafeStatusOAReqSend;
0383 G_awCyclicNoneSafe_Send[57] := drive13PROFIsafeStatusToggleDSend;
0384 G_awCyclicNoneSafe_Send[58] := drive13PROFIsafeStatusResTimSend;
0385 G_awCyclicNoneSafe_Send[59] := drive13PROFIsafeStatusWDTimeoutSend;
0386 G_awCyclicNoneSafe_Send[60] := drive14PROFIsafeStatusActFVSend;
0387 G_awCyclicNoneSafe_Send[61] := drive14PROFIsafeStatusCRCSend;
0388 G_awCyclicNoneSafe_Send[62] := drive14PROFIsafeStatusConsnrRSend;
0389 G_awCyclicNoneSafe_Send[63] := drive14PROFIsafeStatusDeviceFaultSend;
0390 G_awCyclicNoneSafe_Send[64] := drive14PROFIsafeStatusFVactivatedSend;
0391 G_awCyclicNoneSafe_Send[65] := drive14PROFIsafeStatusHostCRCSend;
0392 G_awCyclicNoneSafe_Send[66] := drive14PROFIsafeStatusHostTimeoutSend;
0393 G_awCyclicNoneSafe_Send[67] := drive14PROFIsafeStatusiParENSend;
0394 G_awCyclicNoneSafe_Send[68] := drive14PROFIsafeStatusiParOKSend;
0395 G_awCyclicNoneSafe_Send[69] := drive14PROFIsafeStatusOACSend;
0396 G_awCyclicNoneSafe_Send[70] := drive14PROFIsafeStatusOAReqSend;
0397 G_awCyclicNoneSafe_Send[71] := drive14PROFIsafeStatusToggleDSend;
0398 G_awCyclicNoneSafe_Send[72] := drive14PROFIsafeStatusResTimSend;
0399 G_awCyclicNoneSafe_Send[73] := drive14PROFIsafeStatusWDTimeoutSend;
0400
0401
0402 (* Cyclic data exchange send *)
0403 fbCyclicNoneSafeSend(
0404     EN:=TRUE,
0405     DATA:= ADR(G_awCyclicNoneSafe_Send),
0406     DATA_LEN:= SIZEOF(G_awCyclicNoneSafe_Send),
0407     DONE=> ,
0408     ERR=> ,
0409     ERNO=> );
0410
```

Ensimmäisen virhetaulukon kirjoituslohko

```
0001 FUNCTION_BLOCK Write_d11
0002 VAR_INPUT
0003     oldFailStatus: BOOL := FALSE;
0004     startCommand: BOOL := TRUE;
0005     sDevName: BOOL;
0006     SD_Write11: BOOL := TRUE;
0007 END_VAR
0008 VAR_OUTPUT
0009     WriteDone: BOOL;
0010     WriteError: BOOL := FALSE;
0011 END_VAR
0012 VAR
0013
0014     diagAmount: DWORD := 0;
0015
0016     drive1_PN_ALARMOut: ARRAY [0..999] OF BYTE;
0017
0018
0019     uSlotti: BYTE := 2;
0020
0021     done: BOOL;
0022     errorNo: BOOL;
0023     Segment: STRING :='[nicce]';
0024     WriteData: ARRAY [0..100] OF WORD;
0025     ReadData: ARRAY [0..100] OF WORD;
0026     i: INT := 0;
0027
0028
0029     WriteErrNumber: WORD := 0;
0030     SD_WRITE_d11: SD_WRITE;
0031
0032     SD_EndWrite: BOOL := TRUE;
0033     SD_READ: BOOL := TRUE;
0034     SD_EndRead: BOOL := TRUE;
0035
0036 END_VAR
0037 VAR_IN_OUT
0038
0039 END_VAR
```

```
0001
0002     SD_WRITE_d11(
0003         EN:= SD_Write11,
0004         ATTRIB:=2 ,
0005         FILENO:=11 ,
0006         SEG:= ADR(Segment),
0007         FORMAT:= 16,
0008         NVAR:=85 ,
0009         ADRVAR:= ADR(Drive11_write_table[1]),
0010         DONE=> WriteDone,
0011         ERR=> WriteError,
0012         ERNO=> WriteErrNumber);
0013
0014 IF SD_WRITE_d11.DONE THEN
0015     SD_Write11 := FALSE;
0016 ELSIF SD_WRITE_d11.ERR THEN
0017     SD_Write11 := FALSE;
0018 END_IF;
0019
```

Toisen virhetaulukon kirjoituslohko

```
0001 FUNCTION_BLOCK Write_d12
0002 VAR_INPUT
0003     oldFailStatus: BOOL := FALSE;
0004     startCommand: BOOL := TRUE;
0005     sDevName: BOOL;
0006     SD_Write12: BOOL := TRUE;
0007 END_VAR
0008 VAR_OUTPUT
0009     WriteDone: BOOL;
0010     WriteError: BOOL := FALSE;
0011 END_VAR
0012 VAR
0013
0014     diagAmount: DWORD := 0;
0015
0016     drive1_PN_ALARMOut: ARRAY [0..999] OF BYTE;
0017
0018
0019     uSlotti: BYTE := 2;
0020     error: BOOL;
0021     done: BOOL;
0022     errorNo: BOOL;
0023     Segment: STRING :='[nicce]';
0024     WriteData: ARRAY [0..100] OF WORD;
0025     ReadData: ARRAY [0..100] OF WORD;
0026     i: INT := 0;
0027
0028
0029     WriteErrNumber: WORD := 0;
0030     SD_WRITE_d12: SD_WRITE;
0031
0032     SD_EndWrite: BOOL := TRUE;
0033     SD_READ: BOOL := TRUE;
0034     SD_EndRead: BOOL := TRUE;
0035 END_VAR
0036 VAR_IN_OUT
0037
0038 END_VAR
```

```
0001
0002     SD_WRITE_d12(
0003         EN:= SD_Write12,
0004         ATTRIB:=2 ,
0005         FILENO:=12 ,
0006         SEG:= ADR(Segment),
0007         FORMAT:= 16,
0008         NVAR:=85 ,
0009         ADRVAR:= ADR(Drive12_write_table[1]),
0010         DONE=> WriteDone,
0011         ERR=> WriteError,
0012         ERNO=> WriteErrNumber);
0013
0014 IF SD_WRITE_d12.DONE THEN
0015     SD_Write12 := FALSE;
0016 ELSIF SD_WRITE_d12.ERR THEN
0017     SD_Write12 := FALSE;
0018 END_IF;
0019
```

Kolmannen virhetaulukon kirjoituslohko

```
0001 FUNCTION_BLOCK Write_d13
0002 VAR_INPUT
0003     oldFailStatus: BOOL := FALSE;
0004     startCommand: BOOL := TRUE;
0005     sDevName: BOOL;
0006     SD_Write13: BOOL := TRUE;
0007 END_VAR
0008 VAR_OUTPUT
0009     WriteDone: BOOL;
0010     WriteError: BOOL := FALSE;
0011 END_VAR
0012 VAR
0013
0014     diagAmount: DWORD := 0;
0015
0016     drive1_PN_ALARMOut: ARRAY [0..999] OF BYTE;
0017
0018
0019     uSlotti: BYTE := 2;
0020     error: BOOL;
0021     done: BOOL;
0022     errorNo: BOOL;
0023     Segment: STRING :='[nicce]';
0024     WriteData: ARRAY [0..100] OF WORD;
0025     ReadData: ARRAY [0..100] OF WORD;
0026     i: INT := 0;
0027
0028
0029     WriteErrNumber: WORD := 0;
0030     SD_WRITE_d13: SD_WRITE;
0031
0032     SD_EndWrite: BOOL := TRUE;
0033     SD_READ: BOOL := TRUE;
0034     SD_EndRead: BOOL := TRUE;
0035
0036 END_VAR
0037 VAR_IN_OUT
0038
0039 END_VAR
```

```
0001
0002     SD_WRITE_d13(
0003         EN:= SD_Write13,
0004         ATTRIB:=2 ,
0005         FILENO:=13 ,
0006         SEG:= ADR(Segment),
0007         FORMAT:= 16,
0008         NVAR:=85 ,
0009         ADRVAR:= ADR(Drive13_write_table[1]),
0010         DONE=> WriteDone,
0011         ERR=> WriteError,
0012         ERNO=> WriteErrNumber);
0013
0014 IF SD_WRITE_d13.DONE THEN
0015     SD_Write13 := FALSE;
0016 ELSIF SD_WRITE_d13.ERR THEN
0017     SD_Write13 := FALSE;
0018 END_IF;
0019
```

Neljännen virhetaulukon kirjoituslohko

```
0001 FUNCTION_BLOCK Write_d14
0002 VAR_INPUT
0003     oldFailStatus: BOOL := FALSE;
0004     startCommand: BOOL := TRUE;
0005     sDevName: BOOL;
0006     SD_Write14: BOOL := TRUE;
0007 END_VAR
0008 VAR_OUTPUT
0009     WriteDone: BOOL;
0010     WriteError: BOOL := FALSE;
0011 END_VAR
0012 VAR
0013
0014     diagAmount: DWORD := 0;
0015
0016     drive1_PN_ALARMOut: ARRAY [0..999] OF BYTE;
0017
0018
0019     uSlotti: BYTE := 2;
0020     error: BOOL;
0021     done: BOOL;
0022     errorNo: BOOL;
0023     Segment: STRING :='[nicce]';
0024     WriteData: ARRAY [0..100] OF WORD;
0025     ReadData: ARRAY [0..100] OF WORD;
0026     i: INT := 0;
0027
0028
0029     WriteErrNumber: WORD := 0;
0030     SD_WRITE_d14: SD_WRITE;
0031
0032     SD_EndWrite: BOOL := TRUE;
0033     SD_READ: BOOL := TRUE;
0034     SD_EndRead: BOOL := TRUE;
0035
0036 END_VAR
0037 VAR_IN_OUT
0038
0039 END_VAR
```

```
0001
0002     SD_WRITE_d14(
0003         EN:= SD_Write14,
0004         ATTRIB:=2 ,
0005         FILENO:=14 ,
0006         SEG:= ADR(Segment),
0007         FORMAT:= 16,
0008         NVAR:=85 ,
0009         ADRVAR:= ADR(Drive14_write_table[1]),
0010         DONE=> WriteDone,
0011         ERR=> WriteError,
0012         ERNO=> WriteErrNumber);
0013
0014 IF SD_WRITE_d14.DONE THEN
0015     SD_Write14 := FALSE;
0016 ELSIF SD_WRITE_d14.ERR THEN
0017     SD_Write14 := FALSE;
0018 END_IF;
```

Tarkistustaulukon kirjoituslohko

```
0001 FUNCTION_BLOCK Write_OK
0002 VAR_INPUT
0003     oldFailStatus: BOOL := FALSE;
0004     startCommand: BOOL := TRUE;
0005     sDevName: BOOL;
0006     SD_WriteOK: BOOL := TRUE;
0007 END_VAR
0008 VAR_OUTPUT
0009     WriteDone: BOOL;
0010     WriteError: BOOL := FALSE;
0011 END_VAR
0012 VAR
0013
0014     diagAmount: DWORD := 0;
0015
0016     drive1_PN_ALARMOut: ARRAY [0..999] OF BYTE;
0017
0018
0019     uSlotti: BYTE := 2;
0020     error: BOOL;
0021     done: BOOL;
0022     errorNo: BOOL;
0023     Segment: STRING :='[nicce]';
0024     WriteData: ARRAY [0..100] OF WORD;
0025     ReadData: ARRAY [0..100] OF WORD;
0026     i: INT := 0;
0027
0028
0029     WriteErrNumber: WORD := 0;
0030     SD_WRITE_OK: SD_WRITE;
0031
0032     SD_EndWrite: BOOL := TRUE;
0033     SD_READ: BOOL := TRUE;
0034     SD_EndRead: BOOL := TRUE;
0035
0036 END_VAR
0037 VAR_IN_OUT
0038
0039 END_VAR
```

```
0001
0002     SD_WRITE_OK(
0003         EN:= SD_WriteOK,
0004         ATTRIB:=2 ,
0005         FILENO:=0 ,
0006         SEG:= ADR(Segment),
0007         FORMAT:= 16,
0008         NVAR:=349 ,
0009         ADRVAR:= ADR(AllOK_write_table[1]),
0010         DONE=> WriteDone,
0011         ERR=> WriteError,
0012         ERNO=> WriteErrNumber);
0013
0014 IF SD_WRITE_OK.DONE THEN
0015     SD_WriteOK := FALSE;
0016 ELSIF SD_WRITE_OK.ERR THEN
0017     SD_WriteOK := FALSE;
0018 END_IF;
0019
```

Globaalit muuttujat

```
0001 VAR_GLOBAL
0002     O_awCyclicNoneSafe     AT %QW1.1024 : ARRAY[1..80] OF WORD;
0003
0004     I_awCyclicNoneSafe AT %IW1.1024 : ARRAY[1..80] OF WORD;
0005
0006     Drive11_write_table: ARRAY [1..85] OF WORD;
0007     Drive12_write_table: ARRAY [1..85] OF WORD;
0008     Drive13_write_table: ARRAY [1..85] OF WORD;
0009     Drive14_write_table: ARRAY [1..85] OF WORD;
0010     AllOK_write_table: ARRAY [1..349] OF WORD;
0011
0012     SD_CARD_ERROR11 : INT;
0013
0014 END_VAR
0015
0016 VAR_GLOBAL CONSTANT
0017     STRING_SIZE_READ: WORD := 200;
0018     STRING_SIZE_WRITE: WORD := 80;
0019
0020
0021     STRING_ARRAY_ROWS : BYTE := 15;
0022     STRING_ARRAY_COLUMNS : BYTE := 3;
0023 END_VAR
```

Ensimmäisen taajuusmuuttajan muuttujien kartoitus PPO7-viestilohkoon

: drive11PPO7

PNIO Module Configuration

Parameters:

Name:	Type:	Value:	Unit:	Description:
Module parameters				
Module ident number	UDINT	16#00000107		
Slot number	UINT	1		
Number of submodules	UINT	1		
ModuleState	UINT	16#FFFF		
PPO7 Data Object				
Submodule ident number	UDINT	16#00000107		
Subslot number	UINT	1		
API	UDINT	0		
PROFIsafe supported	BOOL	false		
Isochrone mode				
T_DC_Base	UINT			
T_DC_Min	UINT			
T_DC_Max	UINT			
T_IO_Base	UDINT			
T_IO_InputMin	UDINT			
T_IO_OutputMin	UDINT			
Isochrone mode required	BOOL	false		
Number of record data	UINT	1		
Properties of submodule	UINT	3		
IOPS Length	USINT	1		
IOCS Length	USINT	1		
SubmoduleState	UINT	16#7FFF		
DiagIndicator	UINT			
General parameters				
Index	UINT	1		
Transfer sequence	UINT	0		
Record data length	UDINT	26		
Record data	ARRAY[0..25] OF BYTE	[16#01,16#00]		

PNIO Module I/O Mapping

Input Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:
drive11Status	Status	UINT	%IW2.0		
drive11SpeedAct	Speed Actual	UINT	%IW2.1		
drive11ActPZD3	Actual PZD3	UINT	%IW2.2		
drive11ActPZD4	Actual PZD4	UINT	%IW2.3		
drive11ActPZD5	Actual PZD5	UINT	%IW2.4		
drive11ActPZD6	Actual PZD6	UINT	%IW2.5		
drive11ActPZD7	Actual PZD7	UINT	%IW2.6		
drive11ActPZD8	Actual PZD8	UINT	%IW2.7		
drive11ActPZD9	Actual PZD9	UINT	%IW2.8		
drive11ActPZD10	Actual PZD10	UINT	%IW2.9		
drive11ActPZD11	Actual PZD11	UINT	%IW2.10		
drive11ActPZD12	Actual PZD12	UINT	%IW2.11		

Output Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:

: drive11PPO7

drive11Command	Command	UINT	%QW2.0
drive11SpeedRef	Speed Reference	UINT	%QW2.1
drive11RefPZD3	Reference PZD3	UINT	%QW2.2
drive11RefPZD4	Reference PZD4	UINT	%QW2.3
drive11RefPZD5	Reference PZD5	UINT	%QW2.4
drive11RefPZD6	Reference PZD6	UINT	%QW2.5
drive11RefPZD7	Reference PZD7	UINT	%QW2.6
drive11RefPZD8	Reference PZD8	UINT	%QW2.7
drive11RefPZD9	Reference PZD9	UINT	%QW2.8
drive11RefPZD10	Reference PZD10	UINT	%QW2.9
drive11RefPZD11	Reference PZD11	UINT	%QW2.10
drive11RefPZD12	Reference PZD12	UINT	%QW2.11

Information

Name:	PPO Type 7
Vendor:	ABB Oy
Version:	SW=V2.99, HW=V1.0
Order number:	6410038040802
Description:	PPO Type 7

Ensimmäisen taajuusmuuttajan muuttujien kartoitus PS1-viestilohkoon

: drive11_ABB_PS1

PNIO Module Configuration

Parameters:

Name:	Type:	Value:	Unit:	Description:
Module parameters				
Module ident number	UDINT	16#00000221		
Slot number	UINT	2		
Number of submodules	UINT	1		
ModuleState	UINT	16#FFFF		
Safety module	INT	1		
F-Parameters				
Submodule ident number	UDINT	16#00000221		
Subslot number	UINT	1		
API	UDINT	0		
PROFIsafe supported	BOOL	true		
Isochrone mode				
T_DC_Base	UINT			
T_DC_Min	UINT			
T_DC_Max	UINT			
T_IO_Base	UDINT			
T_IO_InputMin	UDINT			
T_IO_OutputMin	UDINT			
Isochrone mode required	BOOL	false		
Number of record data	UINT	0		
Properties of submodule	UINT	3		
IOPS Length	USINT	1		
IOCS Length	USINT	1		
SubmoduleState	UINT	16#7FFF		
DiagIndicator	UINT			
F_ParameterRecordData				
FParamDescCRC	UINT	7378		
Index	UINT	1		
Transfer sequence	UINT	0		
FParameter	ARRAY[0..15] OF BYTE	[8, 64, 0, 100, 0, 11, 0, 200, 192, 70]		

PNIO Module I/O Mapping

Input Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:
drive11STOact	STO active	BOOL	%IX2.24.7		
drive11_SF_end_ack_req	SF end ack req	BOOL	%IX2.29.0		
drive11PSStatus	Byte0	BYTE	%IB2.32		

Output Parameters:

Mapping:	Channel:	Type:	Address:	Unit:	Description:
drive11SLS2req	Reserved out0 0	BOOL	%QX2.24.0		
drive11SLS1req	SLS1 activate	BOOL	%QX2.24.1		
drive11SS1req	SS1 activate	BOOL	%QX2.24.4		
drive11SSEreq	SSE activate	BOOL	%QX2.24.5		
drive11POUSreq	POUS activate	BOOL	%QX2.24.6		
drive11STOreq	STO activate	BOOL	%QX2.24.7		
drive11SLS4req	SLS4 activate	BOOL	%QX2.25.6		
drive11SLS3req	SLS3 activate	BOOL	%QX2.25.7		
drive11VarSLSreq	Variable SLS activate	BOOL	%QX2.26.0		
drive11_SF_end_ack	SF end ack	BOOL	%QX2.26.2		
drive11NegScale	Negative scaling	BOOL	%QX2.27.6		
drive11PosScale	Positive scaling	BOOL	%QX2.27.7		

: drive11_ABB_PS1

drive11VarSLSLimit	ABB_PS1 Out VarSLSLimit	INT	%QW2.14
drive11PSControl	Byte0	BYTE	%QB2.30

Information

Name:	PROFIsafe ABB_PS1
Vendor:	ABB Oy
Version:	SW=V2.99, HW=V1.0
Order number:	6410038040802
Description:	PROFIsafe ABB_PS1