WWW.JAMK.FI

LOGIIKAN JA TIEDONKERUUOHJELMAN VÄLISEN KOMMUNIKOINTIOHJELMAN SUUNNITTELU JA TOTEUTUS

Aleksi Hänninen

Opinnäytetyö
Toukokuu 2012

Automaatiotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) HÄNNINEN, Aleksi	Julkaisun laji Opinnäytetyö	Päivämäärä 30.05.2012
	Sivumäärä 50	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
LOGIIKAN JA TIEDONKERUUOHJELMAN VÄLISEN KOMMUNIKOINTIOHJELMAN SUUNNITTELU JA TOTEUTUS

Koulutusohjelma
Automaatiotekniikan koulutusohjelma

Työn ohjaaja(t)
SELOSMAA, Seppo, Lehtori

Toimeksiantaja(t)
Stresstech Oy, Jyväskylä
PARTANEN, Janne, Automaatiosuunnittelija

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi laadunvalvontaan erikoistunut Stresstech Oy. Työn tavoitteena oli tehdä automaattisessa mittausjärjestelmässä käytettävän ohjelmoitavan logiikan ja Viewscan-tiedonkeruuohjelman välinen kommunikointiohjelma. Kommunikoinnin toteuttavan logiikkaohjelman tuli olla helposti mittausjärjestelmän liikkeenohjauksen käytettävissä. Ohjelman tuli lisäksi olla helposti uudelleenkäytettävissä muissa ohjelmointiympäristöissä.

Kommunikointiohjelman ohjelmointikieleksi valittiin Simatic S7-SCL. IEC 61131-3 -standardin mukaisella tekstipohjaisella kielellä täytettiin ohjelmalta vaadittava uudelleenkäytettävyys. Kommunikointiohjelma toteutettiin Viewscan-tiedonkeruuohjelman määrittelemän protokollan mukaisesti. Tiedonsiirto logiikan ja Viewscanin välillä toteutettiin ethernetin välityksellä sekä Siemensin kommunikaatioprosessorin kanssa käytettävillä AG_SEND- ja AG_RECV-toiminnoilla. Ohjelma jaoteltiin rajapinnan määrittelevään pääohjelmaan sekä sen kutsumiin selkeän tehtävän suorittaviin aliohjelmiin.

Työn tuloksena oli yksinkertaisesti ohjattava kommunikoinnin toteuttava logiikkaohjelma. Ohjelma toteuttaa Viewscanin määrittelemän protokollan, minkä johdosta virheiden ilmaantuminen saatiin minimoitua. Ohjelma myös ilmoittaa kommunikoinnissa tapahtuvista virheistä, jolloin niihin voidaan reagoida esimerkiksi lopettamalla mittaus. Lisäksi ohjelma tallentaa lähetetyt ja vastaanotetut viestit sekä mahdolliset virheilmoitukset. Tämä helpottaa ohjelman käyttöönottoa sekä mahdollisten virhetilanteiden selvittelyä.

Perusteellisten testausten perusteella ohjelma toimi halutulla tavalla. Ohjelma otetaan käyttöön kahdessa automaattisessa mittausjärjestelmässä tämän vuoden aikana.

Avainsanat (asiasanat)

Tiedonsiirto, ohjelmoitava logiikka, STEP7, SCL

Muut tiedot

Author(s)
HÄNNINEN, Aleksi

Type of publication
Bachelor´s Thesis

Date
30052012

Pages
50

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
DESIGNING AND IMPLEMENTING A COMMUNICATION PROGRAM BETWEEN THE PROGRAMMABLE LOGIC CONTROLLER AND DATA ACQUISITION SOFTWARE

Degree Programme
Degree Programme in Automation Technology

Tutor(s)
SELOSMAA, Seppo, Lecturer

Assigned by
Stresstech Oy, Jyväskylä
PARTANEN, Janne, Automation designer

Abstract

The bachelor's thesis was made for Stresstech Oy which is specialized in quality inspections. The aim of the bachelor's thesis was to make a communication program to be used in an automatic measurement system between a Programmable logic controller (PLC) and the Viewscan data acquisition software. The PLC program was supposed to be easily controlled by the program which handles the motion control. The program also had to be easily reusable in different programming environments.

The communication program was done with the Simatic S7-SCL programming language. The required reusability level was achieved with the IEC 61131-3 compliant textual programming language. The communication program was made according to the protocol specified by the Viewscan data acquisition software. Data transfer between the PLC and the Viewscan was made via Ethernet and the Siemens communication processor using AG_SEND and AG_RECV communication blocks. The program was divided into a main program which defines the interface and into a few subroutines for specific tasks.

The result of the thesis was an easily controllable communication program. Errors in the communication were minimized by fulfilling the protocol specified by Viewscan. The program also informs about the errors with outputs which for example can be used halting the measurement. Additionally the program also saves the sent and received messages and possible fault messages. This is for great help when implementing the program and also when solving errors of the system if necessary.

After a thorough testing, the program worked as it was supposed to. The program will be implemented into two measurement systems during this year.

Keywords

Data transfer, programmable logic controller, PLC, STEP7, SCL

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

TYÖSSÄ KÄYTETYT LYHENTEET

CNC	Computerized Numerical Control, tietokoneistettu numeerinen ohjaus
CP	Communication processor, kommunikaatioprosessori
DB	Data Block, tiedostoyksikkö eli muistitila
FC	Function, toiminta ilman staattista muistia
FB	Function Block, toimintayksikkö staattisella muistilla
FBD	Function Block Diagram, graafinen logiikkaohjelmointikieli
IP	Internet protocol, tiedonsiirron reitittävä protokolla
IL	Instruction list, tekstipohjainen logiikkaohjelmointikieli
LD	Ladder Diagram, graafinen logiikkaohjelmointikieli
OB	Organization Block, logiikkaohjelman organisaatioyksikkö
OSI	Open Systems Interconnection, tietoliikennemalli
SCL	Structured Control Language, STEP7 logiikkaohjelmointikieli
SFC	Sequential Function Chart, logiikkaohjelman organisointityökalu
ST	Structured text, tekstipohjainen logiikkaohjelmointikieli
STL	Statement list, STEP7 logiikkaohjelmointikieli
TCP	Transmission control protocol, tiedonsiirron kuljetuspalvelu
UDT	User Data Type, käyttäjän määrittelemä tiedostotyyppi
VAT	Variable declaration table, taulukko mm. muuttujien seuraamiseksi
VS	Viewscan-tiedonkeruuohjelma

1 JOHDANTO

1.1 Opinnäytetyön lähtökohdat

Sujuva kommunikointi ihmisten välisessä yhteistyössä on tärkeää, sillä viestinnässä tapahtunut virhe tai katkos voi kostautua kalliisti. Kommunikointi eli viestintä tulee toteuttaa siten, että katkoksia tai sekaannuksia ei pääse tapahtumaan tai niiden esiintyvyys on erittäin harvinaista. Kommunikoinnin toimivuus myös yhteistyötä tekevien laitteiden välillä on erittäin tärkeää niiden muodostaman sovelluksen toimivuuden kannalta. Koko tuotanto voi seisahtua sovelluksen kommunikoinnissa tapahtuneen virheen seurauksena.

Materiaalien laadunvalvontaan erikoistuneen Stresstech Oy:n automaattisissa mittausjärjestelmissä käyttämä Viewscan-tiedonkeruuohjelman ja ohjelmoitavan logiikan välinen kommunikointi vaati kehittämistä. Käytössä olevan kommunikoinnin toteuttavan logiikkaohjelman kehittämiselle oli todellinen tarve, sillä toimimaton ohjelma aiheuttaa katkoksia tuotannossa. Kommunikointiohjelma on hyvin kriittinen osa automaattisen mittausjärjestelmän toimintaa, sillä kommunikoinnin pettäessä voi kone tai koko tuotantolinja olla seisokissa kymmeniä minuutteja.

Käytössä olevan ohjelman heikkouksia ovat sen vikaherkkyys, puutteellinen toiminta sekä ohjelman monimutkaisuus. Ohjelmasta puuttuu tiettyjä toimintoja, jotka edellytettäisiin virheettömän kommunikoinnin toteuttamiseksi Viewscanin kanssa. Ohjelman monimutkaisuus johtuu käytetystä STL-ohjelmointikielestä sekä sekavasta ohjelmarakenteesta. Vikaherkkyyden vuoksi olemassa olevaa ohjelmaa ei haluttu käyttää tulevissa järjestelmissä. Ohjelman monimutkaisuudesta johtuen ei vanhaa ohjelmaa haluttu käyttää pohjana, vaan kommunikointiohjelma päätettiin tehdä kokonaan uudelleen.

Siemensin S7-300-sarjan ohjelmoitavissa logiikoissa käytettävä kommunikointiohjelma lähettää halutut viestit sekä vastaanottaa ja arvioi Viewscanin vastauksen. Tiedonsiirto logiikkaohjelman ja Viewscanin välillä on toteutettu ethernet-kommunikaatioprosessorin välityksellä sekä sen kanssa käytettävillä Siemensin valmiilla toiminnoilla.

Opinnäytetyön aihe antoi ammatillisen kehittymisen kannalta erinomaiset edellytykset syventää logiikkaohjelmoinnin taitoja sekä tietoliikenteen ymmärrystä.

1.2 Opinnäytetyön tavoitteet

Opinnäytetyön tehtävänä oli tehdä Siemens SIMATIC STEP7 -ohjelmointiympäristöllä ohjelma tai ohjelmalohkoja, joilla mahdollistetaan ohjelmoitavan logiikan ja Viewscan-tiedonkeruuohjelman välinen kommunikointi. Ohjelman tuli olla rakenteeltaan ja ohjelmointitavaltaan mahdollisimman selkeä eikä se saanut olla vikaantumisaltis. Lisäksi ohjelma tuli rakentaa niin, että sitä voidaan hyödyntää mahdollisimman pienillä muutoksilla eri valmistajien ohjelmoitavissa logiikoissa ja ohjelmointiympäristöissä.

Kommunikointiohjelman tuli osata käsitellä kaikki Viewscanilta saadut vastaukset sekä havaita mahdolliset virhetilanteet ja tuoda ne myös ilmi sitä käyttävälle ohjelmalle. Virhetilanteiden selvittämistä varten ohjelman tuli myös kerätä talteen lähetetyt ja vastaanotetut viestit sekä virheilmoitukset. Sovelluksesta tuli myös tehdä tarvittava dokumentointi, jotta ohjelma olisi helposti ymmärrettävissä ja käytettävissä.

Tehtyä ohjelmaa käytetään osana automaattista mittausjärjestelmää. Ohjelman käytön tuli olla joustava järjestelmäkohtaisen muokattavuuden vuoksi. Tämän takia ohjelmasta ei voitu tehdä täysin itsenäisesti toimivaa, vaan sitä ohjataan sen rinnalle rakennetulla logiikkaohjelmalla. Ohjelman rajapinnan rinnalle toteuttavaan ohjelmaan tuli tästä syystä olla mahdollisimman yksinkertainen.

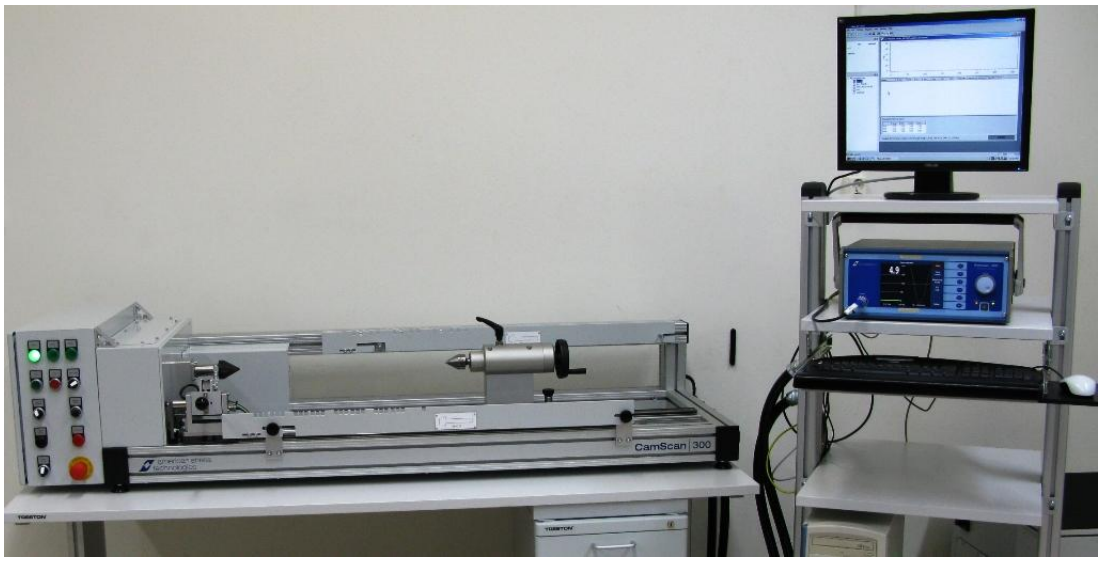
Logiikkaohjelman ei kuitenkaan tarvinnut toteuttaa kaikkia Viewscanin tukemia ulkoisen ohjauksen toimintoja, vaan vain automaattisissa järjestelmissä yleensä käytetyt toiminnot.

1.3 Stresstech Oy

Opinnäytetyön toimeksiantajana toimi Vaajakoskella sijaitseva Stresstech Oy. Yritys on perustettu vuonna 1984 ja se on Stresstech Groupin emoyhtiö, jolla on tytäryhtiöt Yhdysvalloissa, Saksassa ja Intiassa. Lisäksi Stresstech Groupilla on edustajia ympäri

maailmaa noin parikymmentä. (About us n.d.) Stresstech Oy suunnittelee, kehittää ja valmistaa mittalaitteita, antureita ja mittaussovelluksia materiaalien laadunvalvontaan. Mittalaitteita ja mittaussovelluksia käytetään niin tutkimuskäytössä yliopistoissa kuin myös tehtaissa automaattisissa tuotantolinjoissa.

Yrityksen tarjoamista mittausmenetelmistä tärkeimmät ovat Barkhausenin kohina -menetelmään perustuva mittaus sekä röntgendiffraktiomenetelmä. Näillä kahdella menetelmällä voidaan mitata aineen ominaisuuksia sen rakennetta rikkomatta. Menetelmillä voidaan tutkia kappaleessa esiintyviä jännityksiä ja pinnan kovuuden vaihteluita sekä havaita erilaisia vikoja, kuten hionnan aiheuttamia pinnan palamisia. Yleisimmät mittausta vaativat kohteet ovat kampiakselit, nokka-akselit, laakerit ja hammaspyörät. (About us n.d.) Kuviossa 1 on esitetty tyypillinen sovellus nokka-akselin hiottujen pintojen laadun varmistamiseksi Barkhausen-menetelmällä.



KUVIO 1. Tyypillinen Stresstech Oy:n toimittama mittausjärjestelmä (Stresstech Oy 2011)

Stresstechin tärkein asiakaskohderyhmä on autoteollisuus, jonka jälkeen tulevat kone- ja lentokoneteollisuus. Monet autonvalmistajat, Ferrari ja Volvo mukaan lukien, ovat valinneet Stresstechin tuotteet laadunvarmistukseensa.

Vuonna 2011 Stresstech Oy:n palveluksessa työskenteli noin 50 henkilöä, liikevaihdon ollessa noin 5,6 miljoonaa euroa. Kasvua edellisvuoteen tuli liikevaihdon osalta noin 25 % (Tasekirja 2012). Merkittävänä syynä suureen kasvuun oli edellisvuosien lamasta toipuminen. Tänä vuonna yrityksen liikevaihto hyvin todennäköisesti jatkaa kasvuaan. Yrityksen suurin kilpailuetu markkinoilla on kilpailijoiden vähyys. Eritoten Barkhausenin kohina -menetelmään perustuvia teollisia mittalaitteita valmistavia kilpailijoita ei ole.

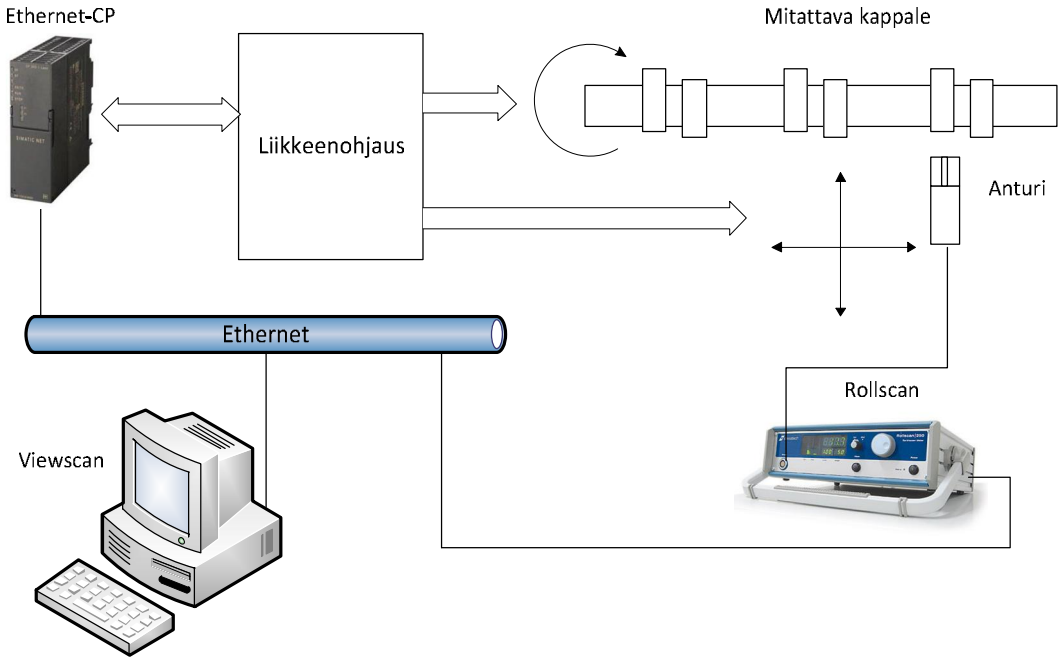
2 AUTOMATISOITU MITTAUSTAPAHTUMA

Useimmiten Barkshausen-mittalaitteella mitataan manuaalisesti. Tyypillisissä sovelluksissa kappale on pyörähdyskappale, kuten nokka- tai kampiakseli. Tällöin kappaleen pyöritys on toteutettu sähköisesti, mutta anturivartta liikutetaan johteita myöten käsivoimin, kuten kuvion 1 laitteessa. Kappaleissa on kuitenkin usein erilaisia mitattavia pintoja, joten niiden mittaaminen edellyttää pinnanmuodoille yksilöityjä antureita. Jos kappaleessa on monia eri anturilla mitattavia pintoja, voi mittaus kokonaisuudessaan kestää pitkän aikaa. Mittauksen helpottamiseksi ja nopeuttamiseksi voidaan mittaus automatisoida. Automatisointi parantaa myös mittauksen luotettavuutta ja toistettavuutta.

Mittaus voidaan suorittaa puoliautomaattisesti, jolloin operaattori vaihtaa kappaleen ja käynnistää mittauksen. Itse mittaustapahtuma on automaattinen. Näin operaattori pystyy käyttämään ja valvomaan samalla muitakin koneita. Mittaus on järkevää toteuttaa täysin automaattisesti, kun kyseessä on tuotantolinja, jonka läpi kulkevat kappaleet halutaan kaikki tarkistaa. Tällöin käytetään robottia tai manipulaattoria kappaleen asettamiseen ja poistamiseen koneesta.

Kommunikointiohjelmaa tarvitaan, kun mittaustapahtuma on automaattinen. Tällöin mittaustapahtuma toteutetaan liikkeenohjauksen, Viewscan-tiedonkeruuohjelman ja mittalaitteen yhteistyöllä (ks. kuvio 2). Liikkeenohjauksen on tiedettävä Viewscanin tila, jotta mittauksen vaatimat liikkeet voidaan toteuttaa. Mittausta ei ole järkevää aloittaa, jos Viewscan ei ole valmis ottamaan vastaan mittausdataa mittalaitteelta.

Mittausta ei myöskään kannata jatkaa loppuun, jos Viewscan menee mittauksen aikana vikatilaan tai menettää yhteyden mittalaitteeseen. Viewscanin ja liikkeenohjauksen kommunikointivirheen jälkeen mittausta ei myöskään kannata jatkaa, sillä ei voida olla varmoja mittaustulosten tallentumisesta. Näiden seikkojen vuoksi kommunikoinnin toimivuudella on suuri merkitys mittauksen onnistumisessa.



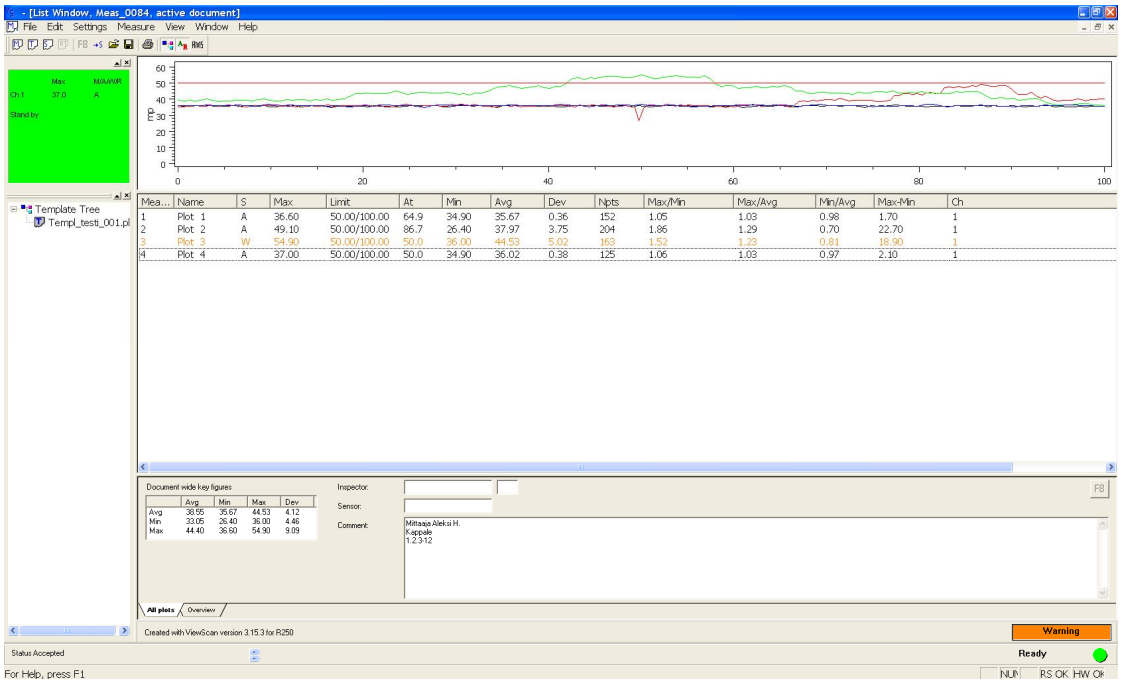
KUVIO 2. Automaattisen mittausjärjestelmän muodostava laitteisto

Automaattisen mittauksen liikkeenohjaus on toteutettu joko numeerisella ohjauksella tai logiikalla. Kommunikointiohjelma pyörii kuitenkin aina logiikassa, sillä Siemensin CNC:hen on integroitu S7-300-sarjan logiikka. Käytettäessä CNC:tä voidaan liikkeenohjauksen toiminnoilla lukea ja kirjoittaa logiikan tiedostoyksikön muistia. Tiedostoyksikön muistilla taas voidaan logiikkaohjelman puolella käskyttää kommunikointiohjelmaa lähettämään halutut viestit Viewscanille.

3 VIEWSCAN-TIEDONKERUUOHJELMA

3.1 Tiedonkeruuohjelman käyttötarkoitus

Viewscan on tietokoneohjelma, jolla voidaan kerätä ja analysoida mittalaitteen mittausdataa (ks. kuvio 3). Ohjelmaa käytetään yrityksen valmistamien Barkhausen-menetelmään perustuvien mittalaitteiden, kuten Rollscanien, kanssa. Ohjelma mahdollistaa edellisten mittaustulosten tarkastelun uuden mittauksen aikana sekä tulosten tallentamisen ja tulostamisen. Ohjelmaan voidaan asetella myös raja hyväksytylle ja hylätylle kappaleelle. Se myös mahdollistaa mittauspohjan eli templaten tekemisen ja käyttämisen mittauksissa usein toistuville kappaleille. (Software ViewScan n.d.)



KUVIO 3. Viewscan-tiedonkeruuohjelman käyttöliittymän perusnäkymä

Mittauspohja mahdollistaa automaattisen kappaleiden mittaamisen. Jokaiselle erilaiselle kappaleelle voi olla oma mittauspohja, johon on määritelty hyväksymisrajat. Mittauksen päätyttyä kysytyn mittaustuloksen perusteella voidaan myös tehdä tiet-

tyjä toimenpiteitä, kuten lajitella hyväksytyt ja hylätyt kappaleet eri pisteisiin esimerkiksi robotilla.

3.2 Kommunikointi ulkoisen laitteen kanssa

Ulkoisella laitteella, kuten ohjelmoitavalla logiikalla, on mahdollista kommunikoida Viewscanin kanssa. Kommunikointi on automaattisen mittaustapahtuman edellytys, sillä logiikan on valmisteltava Viewscan mittausta varten sekä varmistettava mittauksen eteneminen tarkoitetulla tavalla.

Kommunikoinnissa ulkoinen laite toimii isäntälaitteena ja Viewscan orjalaitteena. Viewscan ei siis itsenäisesti lähetä mitään viestejä, vaan ainoastaan kuittaa viestin vastaanotetuksi tai vastaa isäntälaitteen kyselyyn. Kommunikointi on toteutettu siten, että yhtä kyselyä kohti Viewscan lähettää aina yhden vastauksen. Isäntälaitteen tulee odottaa Viewscanin vastausta vähintään kaksi sekuntia viestin lähettämisestä, minkä aikana isäntälaite ei saa lähettää muita viestejä. Jos kahden sekunnin aikana ei kuitenkaan saada vastausta, katsotaan Viewscanin puolen kommunikoinnissa tai yhteydessä olevan jotain vialla. Normaalisti vastaus tulee lähes välittömästi ja yhden sekunnin aikana voidaan tarvittaessa lähettää useita viestejä.

Viewscanin ja ulkoisen ohjaimen kommunikointi ei ole kovin aikakriittinen, joten satunnaisesti esiintyvät pienet viiveet eivät ole vaarallisia. Niillä ei ole vaikutusta mittauksen tallentumiseen, sillä ulkoinen ohjain ei aktiivisesti kysele mittausdataa, vaan lähinnä varmistaa, että Viewscan saa kerättyä mittalaitteelta vastaanotetun mittausdatan talteen.

3.3 Kommunikointiin käytettävät viestit

Kommunikointiin käytetyt viestit ovat muodoltaan merkkijonoja, joiden käsittely on tietokoneelle helppoa, mutta logiikalle ei niin ominaista. Viesti muodostuu koodisanasta, mahdollisista lisätiedoista sekä viestin päättävistä merkeistä. Kommunikointiin käytetyt viestit on esitetty taulukossa 1. Viestien päätemerkit carriage return ja line feed, <CR><LF>, on jätetty taulukossa esittämättä.

TAULUKKO 1. Viewscanin ohjaukseen käytetyt viestit ja niiden vastaukset

	Logiikan lähettämä viesti	Viewscanin vastaus	Vastauksen tarkoitus
Viewscanin alustus	INIT:	OK:<Viewscan versio>	
Viewscanin ohjelmakierron tilan kysyminen	CYCL:	OK:01 OK:02 OK:03 OK:04 OK:05 OK:06 ERR:<error teksti>	Tyhjäkäynnillä Valmisteluvaihe Valmis mittaukseen Mittaus käynnissä Analysoi tuloksia Jälkikäsittely Virhe
Mittauksen valmistelu	PREP:<ohjelman nimi>	OK: ERR:<error teksti>	Viesti vastaanotettu Virhe
Mittauksen aloitus	START:	OK: ERR:<error teksti>	Viesti vastaanotettu Virhe
Kommentin lähetys	COM:<kommentti teksti>	OK: ERR:<error teksti>	Viesti vastaanotettu Virhe
Mittauksen lopetus	END:	OK: ERR:<error teksti>	Viesti vastaanotettu Virhe
Mittaustuloksen kysyminen	MRES:	OK:01 OK:02 OK:03 OK:04 OK:05 OK:06 ERR:<error teksti>	Hyväksytty Hylätty Varoitus Ei tietoa Rajoja ei asetettu Mittauksia ei tehty Virhe

Viewscan tulee alustaa laitteiden käynnistyksen yhteydessä sekä Viewscanin vastatessa virheilmoituksella johonkin viestiin. Alustuksen jälkeen Viewscan siirtyy tyhjäkäyntitilaan ja jää odottamaan mittauksen valmistelukäskyä.

Viewscanin tilan kyselyllä varmistetaan ohjelman oikea tila lähetettäviä viestejä varten. Mittauksen aloittaminen sekä monien viestien lähettäminen on sallittua vain tietyssä ohjelmakierron tilassa. Kyselyllä myös valvotaan mittauksen aikana tiedonkeruun onnistumista. Viewscanin tilaa voidaan kysellä jatkuvasti, jopa viisi kertaa sekunnissa.

Mittauksen valmistelua varten Viewscan tarvitsee ohjelman nimen, jonka mukaan se avaa oikean mittauspohjan. Valmisteluvaiheessa avattavaan pohjaan voitaisiin lähettää erilaisia parametreja ja asetuksia. Näitä ei kuitenkaan ole käytetty, joten ominaisuutta ei tehty myöskään kommunikointiohjelmaan. Aloituskäskyllä Viewscan lopet-

taa valmistelut ja siirtyy valmiustilaan. Tämän jälkeen mittaus voidaan käytännössä aloittaa. Mittaus aloitetaan ja lopetetaan ohjaamalla mittalaitetta logiikan digitaalisella lähdöllä.

Kommenttiviestissä Viewscanille on ilmoitettu esimerkiksi mitattava kappale, jolloin kappaleen tunnistetiedot saadaan näkyville mittausraportteihin. Kappaleen tunnistetiedot voidaan lukea esimerkiksi viivakoodista logiikalle. Kommentissa voidaan haluttaessa lähettää myös esimerkiksi operaattorin nimi tai muuta tarpeellista tietoa.

Lopetuskäsky lähetetään, kun haluttu määrä mittauksia on suoritettu. Tämän jälkeen kysytään mittaustulos, jolla saadaan selville, onko mitattava kappale sallituissa rajoissa. Hyväksymisrajat voidaan asetella Viewscanin käyttämään mittauspohjaan.

Viewscanin vastaama virheviesti voi ilmoittaa esimerkiksi Viewscanin ja Rollscanin välisen yhteyden katoamisesta tai logiikan virheellisestä tai väärään aikaan lähettämästä käskystä. Mittauksen valmistelulle on myös oma virheensä, joka ilmoittaa, jos valmisteltavaa mittauspohjaa ei löydy.

4 TIEDONSIIRTO

4.1 Ethernet-verkko

Tiedonsiirto Viewscan-tiedonkeruuohjelman ja logiikan välillä voidaan toteuttaa sarjaliikenteisesti joko RS232-portin tai ethernetin välityksellä. Yleensä tiedonsiirron fyysisenä kerroksena on käytetty ethernetiä, niin kuin tässäkin työssä. Tämä mahdollistaa standardikaapeleiden käytön sekä muiden laitteiden, kuten kosketuspaneelin, kytkemisen samaan lähiverkkoon.

Ethernet on yleisin lähiverkon toteuttamiseen käytettävä tekniikka. Moni kaupallinen tiedonsiirtosovellus rakentuukin ethernetin päälle. Tämä sallii erilaisten sovellusten olemisen samassa verkossa, mutta ei välttämättä niiden välistä kommunikointia. Ethernetin fyysinen kerros ja siirtoyhteys on määritelty IEEE 802.3 standardissa. (Caro 2009, 108.) Fyysiseen kerrokseen kuuluvat muun muassa tiedonsiirrossa käytettävät

kaapelit ja liittimet. Nykyään yksi yleisimmistä tiedonsiirron fyysisistä toteutuksista on kierretty parikaapeli RJ45-liittimillä.

Ethernetiin perustuvia kaupallisia teollisuusratkaisuja on olemassa monia. Useimmiten ne eivät ole toistensa kanssa yhteensopivia. Näitä ovat esimerkiksi EtherCat, PROFINET ja Modbus/TCP. Ethernet-verkon topologia eli sen fyysinen rakentuminen voi olla monenlainen. Yleisimmät topologiat ovat rengas, tähti, puu, väylä ja verkko. Ethernet toimii kuviossa 4 esitetyn tietoliikenteen referenssimallin, ISO OSI-mallin, sekä TCP/IP-mallin 1. ja 2. kerroksella (Seppälä 2008, 64, 70.)

KUVIO 4. Periaatemallien vertailu tiedonsiirron toteuttamiseksi sovellusten välillä (Seppälä 2008, 23, muokattu)

4.2 TCP/IP-protokolla

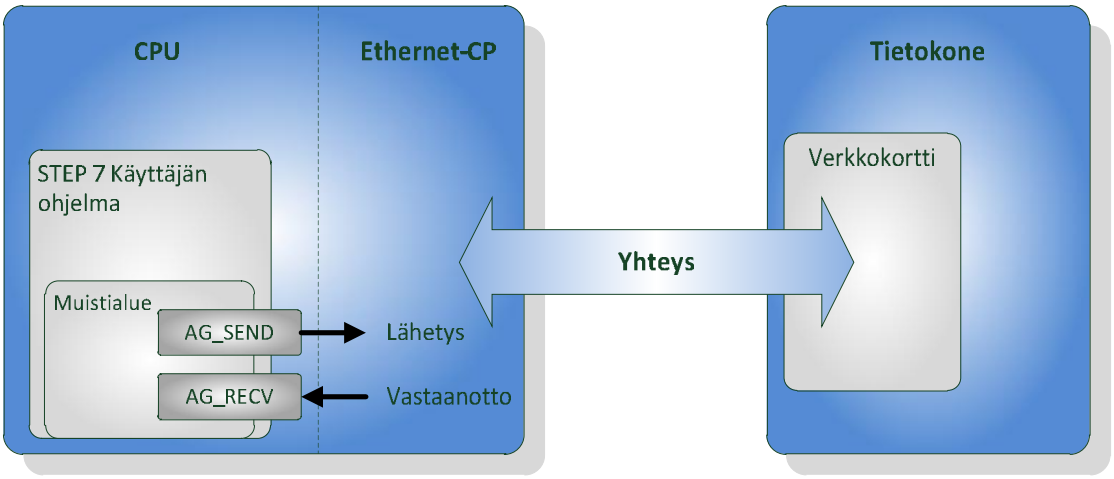
Tietokoneet tarvitsevat tarkat säännöt eli protokollat, joiden mukaan datan siirto sovellusten välillä toteutetaan. Tiedonsiirron onnistumiseksi on lähettäjän ja vastaanottajan toimittava aina samojen käyttäytymissääntöjen mukaisesti. TCP/IP on valmistaja- ja laiteriippumaton protokollaperhe, jonka tehtävänä on tarjota verkoille tarpeelliset liikenne- ja viestintäsäännöt. Verkkokerros eli IP-protokolla sijoittuu OSI-mallin siirtoyhteyskerroksen yläpuolelle, ja se yhdistää erilaiset siirtoverkot muodostamalla päästä-päähän-yhteyden laitteiden välille. TCP eli kuljetuskerros sijoittuu OSI-mallin neljännelle kerrokselle mahdollistaen taas luotettavan päästä-päähän-yhteyden. (Ahola & Sundell 2006, 19, 103–104.) TCP/IP-protokollat siis huolehtivat

yhteyden muodostamisesta ja ylläpitämisestä sekä lähetetyn datan siirtymisestä luotettavasti ja virheettömänä haluttujen laitteiden välillä.

4.3 Kommunikaatioprosessori

Siemensin kommunikaatioprosessoria eli CP:tä käytetään yhdistämään logiikka väylään tai verkkoon, johon siinä itsessään ei ole liitäntää. CP:n välityksellä voidaan myös muodostaa yhteys kolmannen osapuolen laitteisiin, kuten tietokoneeseen tai viivakoodinlukijaan. (SIMATIC S7-300 Communication n.d.) Kommunikaatioprosessori siis mahdollistaa erilaisten yhteyksien muodostamisen muihin laitteisiin. Se myös vähentää logiikan prosessorin kuormitusta sen oman prosessorin ansiosta. (Pigan & Metter 2006, 295.) Kommunikointimoduulin käyttö mahdollistaa avoimen kommunikoinnin SEND- ja RECV-lohkojen avulla PROFINETin tai teollisuus-ethernetin välityksellä (CPU-CPU Communication with SIMATIC Controllers 2010, 267). Kommunikaatioprosessori vastaa yhdessä rakennettavan logiikkaohjelman kanssa kuviossa 4 esitetyn TCP/IP-mallin sovelluskerroksen toiminnoista.

Teollisuus-ethernet kommunikointiin S7-300-sarjan logiikoilla käytetään AG_SEND- ja AG_RECV-toimintoja, joilla voidaan CP:n kautta siirtää jopa 8192 tavua dataa logiikan muistista. AG_SEND-toiminto siirtää dataa logiikan muistista ethernet-CP:lle, joka lähettää sen määritellyn yhteyden kautta eteenpäin. AG_RECV taas tallentaa määritellyn yhteyden kautta CP:lle lähetetyn datan logiikan muistiin. (Program blocks for SIMATIC NET S7 CPs 2011, 23–25, 27, 34.) Kuvio 5 havainnollistaa kommunikaatioprosessorin välityksellä toteutettua tiedonsiirtoa logiikan ja tietokoneen välillä.



KUVIO 5. Logiikan ja tietokoneen välisen tiedonsiirron toteuttaminen (Program blocks for SIMATIC NET S7 CPs 2011, 24, muokattu.)

5 OHJELMOITAVA LOGIIKKA

Ohjelmoitava logiikka on mikroprosessoriohjattu monipuolinen ohjausjärjestelmä. Sitä käytetään teollisuudessa laajasti koneiden, tuotantolinjojen ja erilaisten prosessien ohjaukseen. (Crispin 1997, xi.) Logiikalle tuodaan sisään antureiden antamat tilatiedot, minkä jälkeen lähtöjen toimilaitteiden ohjaukset määräytyvät prosessorin käsittelemän ohjelman mukaisesti. Logiikalla voidaan helposti korvata jopa satoja ennen käytettyjä releitä ja ajastimia. (Keinänen, Kärkkäinen, Lähetkangas & Sumujärvi 2007, 212.)

5.1 Logiikkaohjelmointi

Ohjelmoitavien logiikoiden ohjelmistotuotanto ei ole yhtä kehittynyttä kuin tietokoneohjelmointi, minkä takia logiikkasovellukset voivat olla hyvinkin erilaisia keskenään ja usein tekijästä riippuvaisia. Käytettävät ratkaisumallit ovat myös monesti yrityskohtaisia ja riippuvaisia käytettävistä automaatiotuotteista. Näiden lisäksi sovelluksen ylläpidettävyys kärsii usein huonon dokumentoinnin ja sovelluksen rakenteen sekavuuden vuoksi. Näiden ongelmien ratkaisemiseksi logiikkaohjelmointia on pyritty

standardoimaan. Keskeisenä standardina on IEC 61131-3. (Asmala, Koskinen, Koskela, Mätäsniemi, Soini, Strömman, Tommila & Valkonen 2005, 7.)

5.2 Ohjelmointikielet

Ohjelmoitavia logiikoita käsittelevän IEC 61131 -standardisarjan kolmas osa määrittelee ohjelmoitavissa logiikoissa käytettävät kielet sekä niiden syntaksin ja semantiikan. Ohjelmointikielet koostuvat kahdesta tekstipohjaisesta kielestä sekä kahdesta graafisesta kielestä. Lisäksi standardi määrittelee työkalun ohjelman sisäisen rakenteen organisointiin. (IEC 61131-3, 2003, 14.) Standardin määrittelemät ohjelmointitavat ovat seuraavat:

- Käskylista (IL) vastaa Siemensin STL-ohjelmointikieltä, joka on assemblerkieltä muistuttava tekstipohjainen ohjelmointikieli.
- Rakenteinen teksti (ST) vastaa Siemensin S7-SCL-ohjelmointikieltä, joka on Pascal-kieltä muistuttava korkeamman tason tekstipohjainen kieli.
- Tikapuukaavio (LD) vastaa Siemensin LAD-ohjelmointikieltä, joka on Yhdysvalloissa käytettyjä relekaavioita muistuttava graafinen kieli.
- Toimilohkokaavio (FBD) on graafinen ohjelmointikieli, joka perustuu funktioihin ja toimilohkoihin.
- Sekvenssikaavio (SFC) on kuin Siemensin S7-GRAPH, joka kuvaa ohjelman sekvenssityyppistä rakennetta.

IEC 61131-3 -standardi on valmistajien keskuudessa yleisesti hyväksytty ja sitä noudattavia laitteita on ollut markkinoilla jo pitkään. Vaikka suurimmat valmistajat tukevatkin standardia, ne suhtautuvat avoimuuteen epäluuloisesti. (Asmala ym. 2005, 14–15.)

6 SIMATIC S7-SCL-OHJELMOINTIKIELI

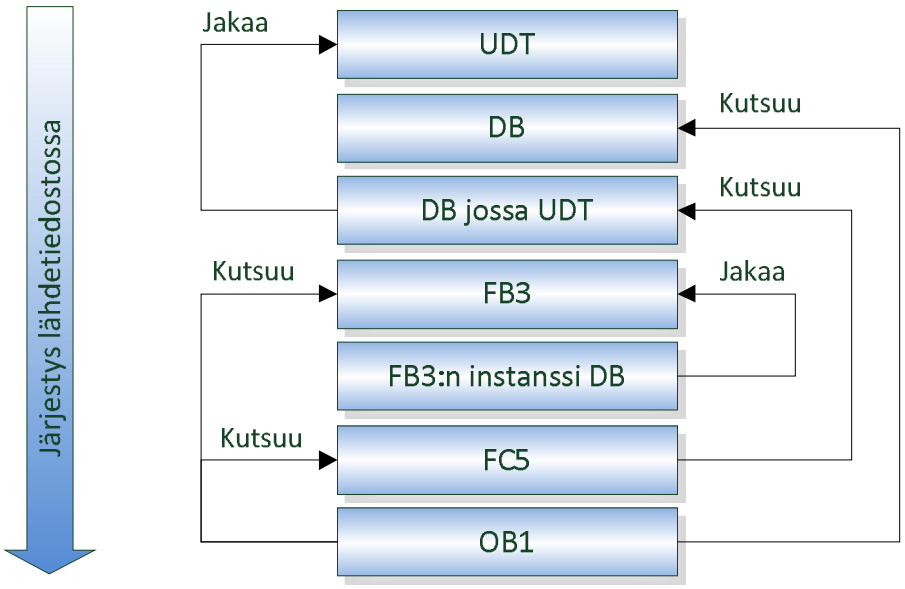
6.1 Ohjelmointikielen käyttö

Seuraavassa kerrotaan tarkemmin kommunikointiohjelman tekemiseen käytetystä ohjelmointikielestä, joka ei monille varmasti ole niin tuttu kuin muut ohjelmointikielet. Siemensin Simatic S7-SCL-ohjelmointikieli vastaa IEC 61131-3 -standardin mukaista rakenteista tekstiä. S7-SCL kuuluu Simatic STEP 7 Professional -ohjelmistopakettiin tai on saatavana optiona myös erikseen (SIMATIC S7-SCL n.d.).

S7-SCL on optimoitu ohjelmoitavien logiikoiden ohjelmointiin sisältäen sekä Pascalin ohjelmointielementit että logiikoissa käytettävät ohjelmointilohkot, kuten ajastimet ja laskurit. S7-SCL sopii erityisesti monimutkaisten algoritmien ja matemaattisten funktioiden ohjelmoimiseen sekä datan käsittelyyn. (S7-SCL V5.3 for S7-300/400 2005, 15.) Muihin ohjelmointikieliin verrattavia etuja ovat tehokkaat ehto- ja valintalauseet IF ja CASE sekä toistorakenteet FOR, WHILE ja REPEAT. Ohjelma on myös helposti luettava ja sen rakenne on selkeä. Ohjelman testaus ja ohjelmointivirheiden etsiminen on myös yksinkertaista debuggerin avulla. (SIMATIC S7-SCL n.d.)

6.2 Ohjelman kirjoittaminen

S7-SCL-ohjelma kirjoitetaan erillisiin lähdetiedostoihin. Koko ohjelma sisältäen useita ohjelmalohkoja voidaan kirjoittaa yhteen lähdetiedostoon. Lähdetiedostoja voidaan kuitenkin käyttää myös useampia, jolloin selkeästi yhteen nivoutuvat ohjelmaosat voidaan esittää samassa lähdetiedostossa. Tällöin ohjelmalohkojen on oltava oikeassa järjestyksessä lähdetiedostossa (ks. kuvio 6). Pääperiaate on, että kutsuttava lohko pitää esitellä ennen sitä kutsuvaa lohkoa (S7-SCL V5.3 for S7-300/400 2005, 118).



KUVIO 6. Ohjelmalohkojen järjestys S7-SCL:n lähdetiedostossa (S7-SCL V5.3 for S7-300/400 2005, 118, muokattu)

S7-SCL:llä voidaan muodostaa kaikki kuvion 6 lohkot. Myös STEP7-kirjastosta löytyvät valmiit ohjelmalohkot ovat käytettävissä normaalisti (S7-SCL V5.3 for S7-300/400 2005, 117). STEP7-ohjelmassa voidaan vapaasti yhdistellä SCL:llä sekä muilla ohjelmointikielillä muodostettuja toimilohkoja. SCL:n lähdetiedostossa voidaan myös kutsua esimerkiksi FBD:llä tai STL:llä tehtyjä lohkoja ja päinvastoin. (S7-SCL V5.3 for S7-300/400 2005, 16.)

6.3 Ohjelmalohkon rakenne

Ohjelmalohkon rakenne noudattaa kaikilla lohkoilla samankaltaista perusrakennetta. Lohkolla on aloitus, jossa määritellään lohkon tyyppi, lohkon numero tai symbolinen nimi sekä FC:n tapauksessa palautusarvon datatyyppi. Tämän jälkeen tulevat vapaaehtoisina otsikko- ja kommenttikenttä sekä lohkon yksityiskohtaisemmat tunnistetiedot. (S7-SCL V5.3 for S7-300/400 2005, 119.)

Edellisten jälkeen tulee lohkon tyypistä riippuen ilmoitusosa, jossa esitellään muun muassa lohkossa käytettävät paikalliset muuttujat sekä tulo- ja lähtöparametrit. Il-

moitusosa jakaantuu aliosioihin, joissa erityyppiset parametrit esitellään omien avainsanojensa sisäpuolella. (S7-SCL V5.3 for S7-300/400 2005, 126.)

Ohjelmalohkoa kutsuttaessa suoritettava ohjelmakoodi on FC:n ja FB:n tapauksessa sijoitettu parametrien esittämisen ja ohjelmalohkon lopetuksen avainsanan väliin. Tällä alueella voidaan sijoittaa arvoja muuttujiin, tehdä laskutoimituksia, kutsua muita ohjelmalohkoja ja käyttää käytettävissä olevia lauserakenteita. (S7-SCL V5.3 for S7-300/400 2005, 128–129.)

Kuviossa 7 on esitetty kommunikointiohjelmassa käytetyn AG_SEND_ERROR-toiminnan rakentuminen sen keskiosa pois leikattuna. Toiminnalle tuodaan sisään tieto virheen tapahtumisesta sekä virheen tilatieto. Virheteksti arvioidaan sisään tuodusta virhetilasta CASE-valintarakenteella, jos virhe on aktiivinen. Toiminto palauttaa virhetilan mukaisen virhetekstin 90 merkkiä pitkänä merkkijonona.

```
 7 FUNCTION AG_SEND_ERROR : STRING[90]
 8
 9 TITLE = 'Get AG send error text'
10 //This block is called when AG_SEND block goes to error state.
11 //This block checks what the error status of the AG_SEND block means
12 //
13 VERSION: '1.0'
14 AUTHOR: SOY
15 FAMILY: VS_COM
16
17 // Block Parameters
18 VAR_INPUT
19     // Input Parameters
20     SEND_ERROR : BOOL; // Is there an error
21     SEND_STATUS : INT; //Status of the AG_SEND in INT format
22
23 END_VAR
24
25     // IF error is occured during sending then error text concerning the error is writted to t
   he LOGI
26     IF SEND_ERROR THEN
27
28         (*Errors are evaluated in INT format. Numbers are negative because of the overflow in
   HEX -> INT conversion
29         *)
30         CASE SEND_STATUS OF
31             //****************Most common errors start*********************
32             // Status  = 0x8183 = -32381 dec -> Connection has not been set up, or connection
   has been disconnected
33             -32381 :
34                 AG_SEND_ERROR := 'Send error: 8183 Connection has not been set up, or has been
    disconnected';
   ______ _______ ______ ______ ______ ______ ______ _____ _____ _____ _____ _____ ____ ____

121             // Status = 0x80D2 = -32558 dec -> Module start address incorrect
122             -32558 :
123                 AG_SEND_ERROR :=  'Send error: 80D2 Module start address incorrect';
124
125         ELSE:
126             // Statements_ELSE
127             AG_SEND_ERROR :=  'Send error: Unknown error status of AG_SEND block';
128
129         END_CASE;
130
131     END_IF;
132
133 END FUNCTION
```

KUVIO 7. S7-SCL-ohjelmointikielellä toteutettu toiminta

6.3 Virheiden paikallistaminen

Ohjelma tarkistaa ohjelmakoodin syntaksin oikeellisuuden käännettäessä lähdetiedosto Compile-toiminnolla. Samalla se luo automaattisesti yksiköt virheettömästä koodista projektin Blocks-kansioon. Koodin käännöllä ei kuitenkaan voida havaita mahdollisia ohjelman sisällä pitäviä loogisia virheitä.

S7-SCL debuggaus toiminnoilla voidaan tarkistaa ohjelman toiminta logiikassa ja paikallistaa mahdolliset loogiset virheet. Ohjelmakoodia voidaan tarkastella monitoroimalla muuttujien arvoja logiikan päivittäessä arvoja syklisesti tai pysäyttämällä ohjelman kulku halutulle ohjelman riville, minkä jälkeen voidaan tarkastella muuttujien arvojen muutoksia esimerkiksi rivi kerrallaan edeten. (S7-SCL V5.3 for S7-300/400 2005, 82.)

Merkkijonojen sisältöä ei kuitenkaan voi tarkastella suoraan debuggauksen yhteydessä. Näitä varten pitääkin tehdä VAT-tiedosto, jossa merkkijonot luetaan auki merkki kerrallaan.

7 OPINNÄYTETYÖN TOTEUTUS

7.1 Noudatetut periaatteet

Opinnäytetyön lähtökohtana oli kehittää Viewscanin ja ohjelmoitavan logiikan välistä kommunikointia. Kommunikoinnin toteuttavan nykyisen logiikkaohjelman monimutkaisuus oli tiedossa jo työn alkaessa, jolloin ohjelma päätettiin tehdä kokonaan uudelleen. Uuden ohjelman tekemisessä ei käytetty apuna juuri ollenkaan käytössä ollutta ohjelmaa sen monimutkaisuuden ja puutteellisuuden vuoksi.

Työn alkaessa tärkein kysymys oli, kuinka logiikka ja Viewscan voivat kommunikoida keskenään. Siemensin laitteiden kommunikoinnista kolmannen osapuolen laitteiden kanssa ei ollut aikaisempaa kokemusta. Mietittävää riitti myös siinä, kuinka ohjelmasta tehtäisiin mahdollisimman yksinkertainen käyttää. Uudelleenkäytettävyyttä ajatellen oli myös selvitettävä muiden logiikoiden mahdollisuuksia.

Luettavuuden takia ohjelma päätettiin tehdä käyttäen vain ja ainoastaan symbolisia nimiä muuttujissa ja toimilohkojen nimissä. Ohjelmassa ei käytetty absoluuttisia osoitteita, jolloin päällekkäisyyksiä ei pääse syntymään rinnalle rakennettavan ohjelman kanssa. Toimilohkojen numeroiden vaihtaminen käy tarvittaessa helpommin, koska niitä ei vaihdon yhteydessä tarvitse itse koodiin vaihtaa.

Jotta ohjelmasta saataisiin yksinkertaisempi, päätettiin merkkijonojen käsittelyt ja vertailut toteuttaa valmiilla IEC-standardin mukaisilla toiminnoilla. Merkkijonotoiminnoilla on mahdollista muun muassa yhdistää merkkijonoja ja erottaa merkkijonosta jokin osio vertailua tai tallennusta varten.

7.2 Ohjelmointikielen valinta

Kommunikointiohjelmaa on käytettävä aina, kun halutaan keskustella logiikalla Viewscanin kanssa. Käytettävä logiikka ei kuitenkaan tule aina olemaan Siemensin valmistama vaan sovelluksen vaatimat toiminnot tai asiakkaan määritykset saattavat edellyttää eri valmistajan laitteiden käyttöä. On siis hyvin mahdollista, että seuraava projekti tehdäänkin esimerkiksi CoDeSys-ohjelmointiympäristöllä. Tämän vuoksi työn yhtenä painopisteenä oli ohjelman uudelleenkäytettävyys. Ohjelmaa tulisi voida käyttää mahdollisimman pienillä muutoksilla muiden valmistajien logiikoissa. Tämän takia tutkittiin ohjelmoitaviin logiikoihin liittyviä standardeja sekä eri ohjelmointiympäristöillä tehtyjen ohjelmien yhteensopivuutta.

IEC 61131-3 on ainut maailmanlaajuinen logiikoiden ohjelmointia käsittelevä standardi (PLCopen n.d.). Hyväksynnän standardinmukaisuudesta eri valmistajien ohjelmoitavissa logiikoissa käytettäville ohjelmointikielille myöntää PLCopen-organisaatio. Sen tuotekohtainen sertifikaattitaulukko osoittaa, että moni valmistaja Siemensin ohella on saanut ST-kielelle vähintään perustason hyväksynnän. Tämä tarkoittaa, että kieli noudattaa standardisyntaksia. (TC3-Certification of IEC 61131-3 Environments n.d.)

Monen valmistajan käskylista-ohjelmointikieli toteuttaa myös standardin vaatimukset, joita Siemensin STL ei kuitenkaan kokonaisuudessaan täytä. Ohjelmointikieltä ei myöskään haluttu käyttää sen vaikeaselkoisuuden vuoksi. FBD eli toimilohkokaavio-

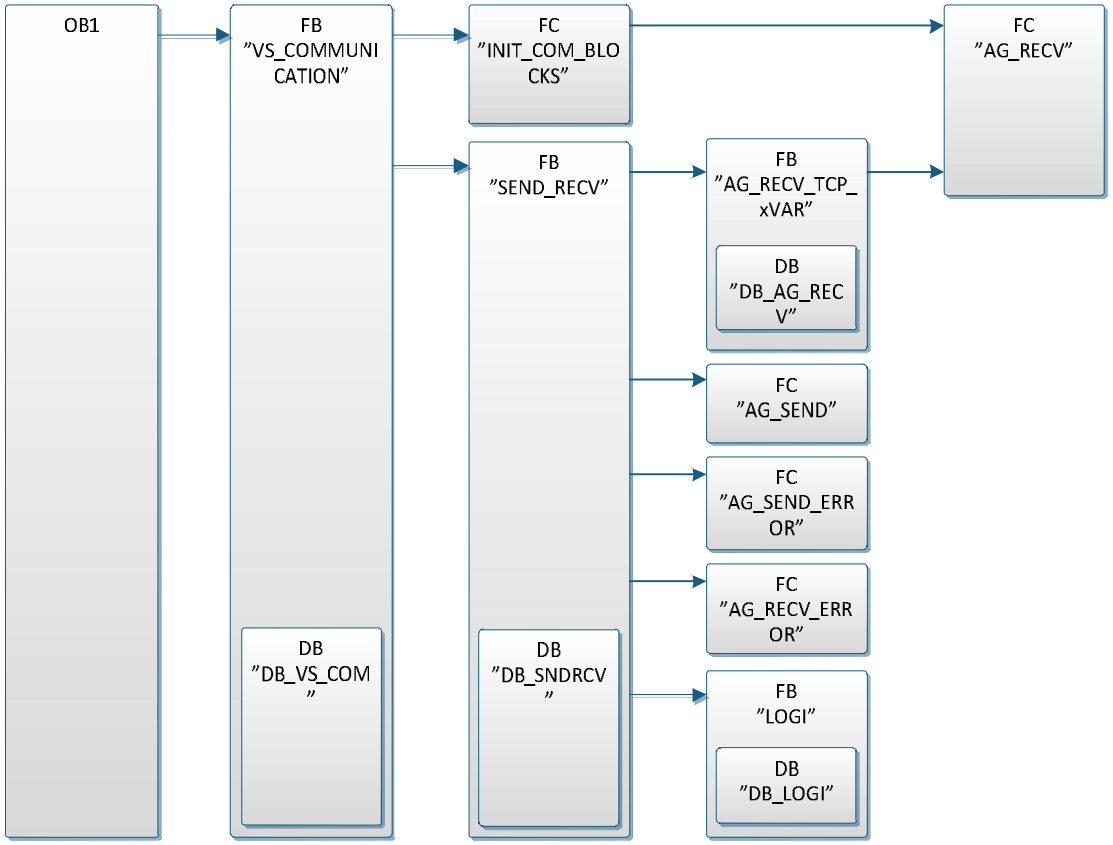
pohjainen toteutus olisi ollut tuttu ja selkeä esitystapa ohjelmalle. Sen uudelleenkäytettävyys ei kuitenkaan ole vaaditulla tasolla. Käytännössä ohjelma pitäisi rakentaa lohko lohkolta kokonaan uudestaan toiseen ohjelmointiympäristöön siirryttäessä.

Structured text -ohjelmointikieli on sen sijaan tekstimuotoista ja monella valmistajalla standardin mukaista, joten se on helposti siirrettävissä järjestelmästä toiseen kopioimalla koodi. Koodiin saadaan myös paljon toimintoja tiiviiseen tilaan ja se on melko helposti ymmärrettävissä myös tietokoneohjelmointia tehneille henkilöille. Näiden seikkojen vuoksi ohjelmaa lähdettiin toteuttamaan ST-kieltä vastaavalla Siemensin S7-SCL-ohjelmointikielellä.

7.3 Ohjelman rakenne

Kommunikointiohjelman tuli olla helposti CNC-ohjaimen tai logiikan käsiteltävissä. Tämän vuoksi liikkeenohjauksen toteuttavalle ohjelmalle kommunikointiohjelman tuli näkyä yksittäisenä toimilohkona, joka määrittää niiden välisen rajapinnan. Tietyn viestin lähettämiseksi vaadittiin liipaisu kyseisen viestin tuloon. Kommunikointiohjelma hoitaa siitä eteenpäin viestin lähettämisen sekä vastauksen vastaanoton ja arvioinnin. Toimilohko ilmoittaa myös viestin onnistuneesta lähettämisestä kyseisen viestin lähdöllä, jotta voitaisiin varmistaa viestin perille meno.

Kommunikointiohjelmaan muodostettiin päätason toimintayksikköä, joka määrittää rajapinnan sitä käyttävälle logiikkaohjelmalle. Pääohjelmassa myös määritetään lähetettävä viesti sekä arvioidaan Viewscanin vastaus. Lisäksi muodostettiin muutama selkeästi rajattu aliohjelma, minkä johdosta ohjelman rakenteesta saatiin selkeä. Kuviossa 8 on esitelty tehdyn ohjelman rakenne pääosiltaan sekä toimintojen kutsumisjärjestys. Kuviosta on jätetty esittämättä kutsutut valmiit toiminnat, kuten merkkijonojen käsittelyihin käytetyt toiminnot.



KUVIO 8. Kommunikointiohjelman rakenne ja toimintojen kutsumisjärjestys

Viestin lähetystä ja vastaanottoa varten tehtiin oma toimintayksikkö. Lähetys- ja vastaanottotoimintoa kutsutaan jokaisen viestin lähetyksen yhteydessä. Näin voitiin käyttää samaa aliohjelmaa kaikkien viestien käsittelyn yhteydessä. Tätä kutsuttaessa käytettiin aina samaa oheis-tiedostoyksikköä, sillä yhden viestin käsittely suoritetaan aina loppuun ennen toisen viestin lähetystä. Viestin lähetys ja vastauksen vastaanotto vaatii AG-lohkojen käsittelystä sekä vastauksen odottamisesta johtuen kuitenkin aina useampia ohjelmakierroksia, minkä vuoksi tarvittiin staattista muistia. Viestin käsittelyn loputtua ei yksikön tilaa tarvitse enää muistaa, vaan sen tila tulee palautua alkuperäiseen tilaan seuraavan viestin lähettämistä varten.

Omat aliohjelmat tehtiin lisäksi AG-lohkojen virheilmoitustekstien määrittelyitä varten, viestien ja virheilmoitusten tallennusta varten sekä ohjelman alustusta varten. Kuviossa 8 esitetyt AG_RECV-, AG_RECV_TCP_xVAR- ja AG_SEND-toimilohkot ovat tiedonsiirtoon käytettyjä Siemensin valmiita toimintoja.

7.4 Viestin lähetys ja vastaanotto

7.4.1 Tiedonsiirron toteutus

Tietokonepohjaisen tiedonkeruuohjelman kommunikointirajapinta viesteineen ja protokollineen oli määritelty, eikä tähän voitu vaikuttaa. Protokollan lisäksi tiedettiin, että tiedonsiirtoon tulee käyttää ethernetiä. Tämän pohjalta alettiin selvittää kuinka ohjelmoitavan logiikan ja Viewscanin välillä voidaan siirtää tietoa. Logiikan ja Viewscanin väliseen tiedonsiirtoon on käytettävä Siemensin CP-moduulia. Kommunikaatioprosessorille on olemassa valmiita toimintoja, joilla voidaan siirtää tietoa logiikan ja ulkoisen laitteen välillä. Näiden tietojen perusteella lähdettiin selvittämään kuinka näitä AG_SEND- ja AG_RECV-lohkoja käytetään. Tässä avuksi tulivat Siemensin manuaalit ja tukisivustojen aihetta käsittelevät artikkelit sekä sieltä löytyneet esimerkkiohjelmat. Siemensin tukisivustojen kautta löytyikin melko paljon tietoa lohkojen käytöstä sekä joitakin esimerkkiohjelmia. Täydellistä esimerkkiohjelmaa, jossa olisi määritelty tarvittavat toiminnot, ei kuitenkaan löytynyt, vaan tieto piti koota pienistä tiedonrippeistä.

Koska AG-lohkojen käytöstä ei ollut kokemusta eikä tietoa mitä ne tarvitsevat ympärilleen muodostaakseen toimivan ohjelman, aloitettiin ohjelman rakentaminen lähetyksen ja vastaanoton käsittelemisestä. Lähetys vaatii useita ohjelmakiertoja sen onnistumiseksi. Myös kyselyn vastausta saattaa joutua odottamaan hetken aikaa. Staattiselle muistille oli siis tarvetta, jotta lohkon tila olisi oikea seuraavalla ohjelmakierroksella. Tämän vuoksi lähetys ja vastaanotto rakennettiin samaan FB:hen, jota voidaan kutsua viestiä lähetettäessä.

7.4.2 Käsiteltävän viestin määrittäminen

Ohjelman päätason toimintayksikkö määrittelee lähetettävän viestin ja arvioi lähetys ja vastaanotto -aliohjelmalta saadun vastauksen. Viesti lähetetään päätason kommunikointilohkon saatuaan käskyn johonkin viestin lähetystuloista. Jokaiselle viestille on oma tulonsa, jonka nouseva reuna tulkitaan viestin lähettämiskäskyksi.

Useimpia viestejä saa kuitenkin lähettää ainoastaan Viewscanin tietyn ohjelmakierron tilan aikana. Näiden viestien ehtona käytetään siis CYCL viestiin saatua vastausta.

Aluksi Viewscanin tilaa aiottiin kysyä vain ennen ehtoja sisältävien viestien lähetystä sekä mittauksen aikana. Tällä haluttiin varmistaa, että kyseisen viestin lähettäminen on sallittua Viewscanin puolesta. Viesti ei kuitenkaan saanut jäädä lähettämättä, jos lähettäminen ei ollutkaan sallittua. Ongelman välttämiseksi oli parempi kysellä Viewscanin tilaa jatkuvasti tietyin väliajoin, jolloin tilan ollessa väärä, voidaan viesti lähettää Viewscanin tilan muuttuessa oikeaksi.

Viestien lähetyskäskyt voivat tulla myös samanaikaisesti tai kun toisen viestin vastaanottoa odotetaan. Viewscanilla saa kuitenkin olla vain yksi viesti käsiteltävänä kerrallaan. Kommunikointiohjelman tuli täten muistaa lähetettäväksi halutut viestit ja lähettää viesti vasta kun edellisen viestin lähetys ja vastaanotto on suoritettu loppuun sekä Viewscanin tila on oikea. Tämän takia käytettiin toimintayksikköä, joka mahdollistaa myös seuraavilla ohjelmakierroksilla voimassa olevan muistin käytön. Viestien hallitsemiseksi ohjelmaan muodostettiin rakenne, joka mahdollistaa viestien käsittelemisen peräkkäin viesti kerrallaan. Yhden viestin lähettäminen ja sen vastauksen arviointi tehdään kokonaisuudessaan ennen seuraavaan siirtymistä.

Viestien lähettämisen järjestystä ei kuitenkaan määrittele tulojen liipaisujärjestys, vaan viestien käsittelyyn käytetty IF-valintalause. IF-valintalauseen haaroista toteutetaan ensimmäinen, jonka ehdot täyttyvät. Näin ollen viestien järjestys tässä valintarakenteessa määrittelee lähetysjärjestyksen. INIT sijoitettiin ensimmäiseksi valintarakenteeseen, sillä sen lähetys haluttiin varmistaa, koska se palauttaa Viewscanin virhetilasta. CYCL asetettiin toiseksi, koska haluttiin varmistaa sen lähetyksen onnistuminen halutun ajan välein. Näin muistissa oleva ohjelmakierron tila, jota käytetään ehtona muiden viestien lähettämisessä, on aina ajan tasalla. Muiden viestien lähettämisjärjestys mukailee oikean mittaustapahtuman viestien lähetysjärjestystä.

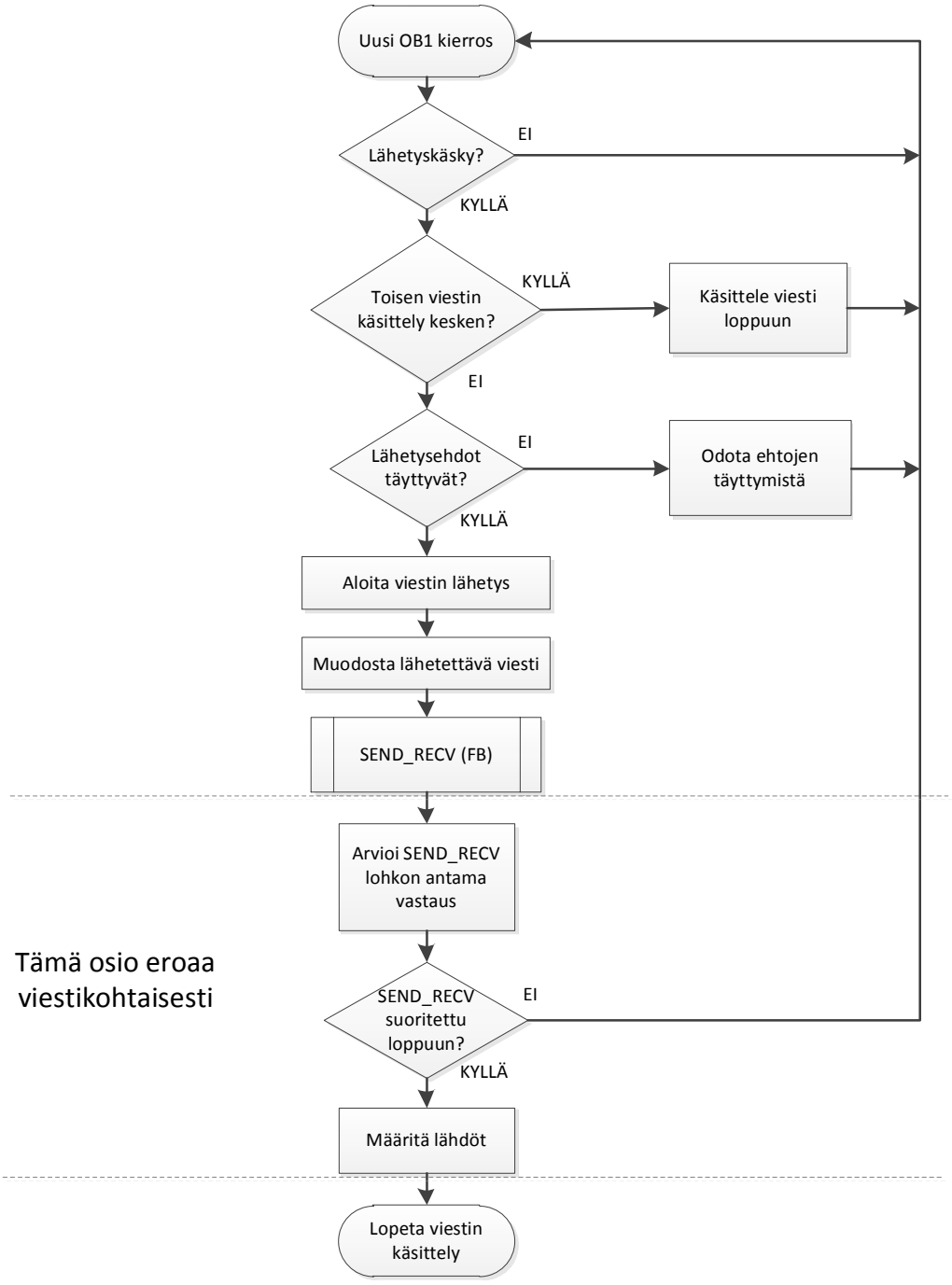
Viestien lähettämisen ehdoksi määriteltiin myös, että INIT on täytynyt olla lähetetty ennen kuin muiden viestien lähetys on mahdollista. Alustuksen lähetyksen muistava muuttuja nollataan Viewscanin ilmoittaessa virheestä, sillä muiden viestien lähettäminen Viewscanin virhetilassa ei ole sallittua. INIT tulee lähettää aina Viewscanin virheilmoituksen jälkeen, joten sen lähettäminen toteutettiin automaattisesti. Näin Viewscanin virhetila saadaan kuitattua ja mahdollistetaan muiden viestien lähetys.

Virhetilanteen jälkeen mittausta ei voida kuitenkaan Viewscanista johtuen jatkaa normaalisti, vaan se tulee aloittaa alusta.

7.4.3 Viestin lähetys

Aloitettaessa jonkun viestin käsittely, estetään samalla muiden viestien lähettäminen. Ohjelman kulku pidetään tulevilla ohjelmakierroksilla aloitetun viestin käsittelemisessä, sillä viestin lähettäminen vaatii aina useita ohjelmakierroksia. Seuraavaan viestiin voidaan siirtyä, kun viesti on lähetetty ja vastaus tähän on saatu tai tiedonsiirrossa tapahtuu virhe. Tiettyjen virheiden ilmaantuessa estetään viestien lähetys, kunnes virhe kuitataan. Käsiteltävän viestin määrittävästä pääohjelmasta kutsutaan viestin lähetyksen ja vastaanoton suorittavaa SEND_RECV-aliohjelmaa. SEND_RECV-toimintayksikölle tuodaan sisään lähetettävä viesti kokonaisuudessaan ja lohko palauttaa vastaanotetun viestin tai ilmoituksen virheestä.

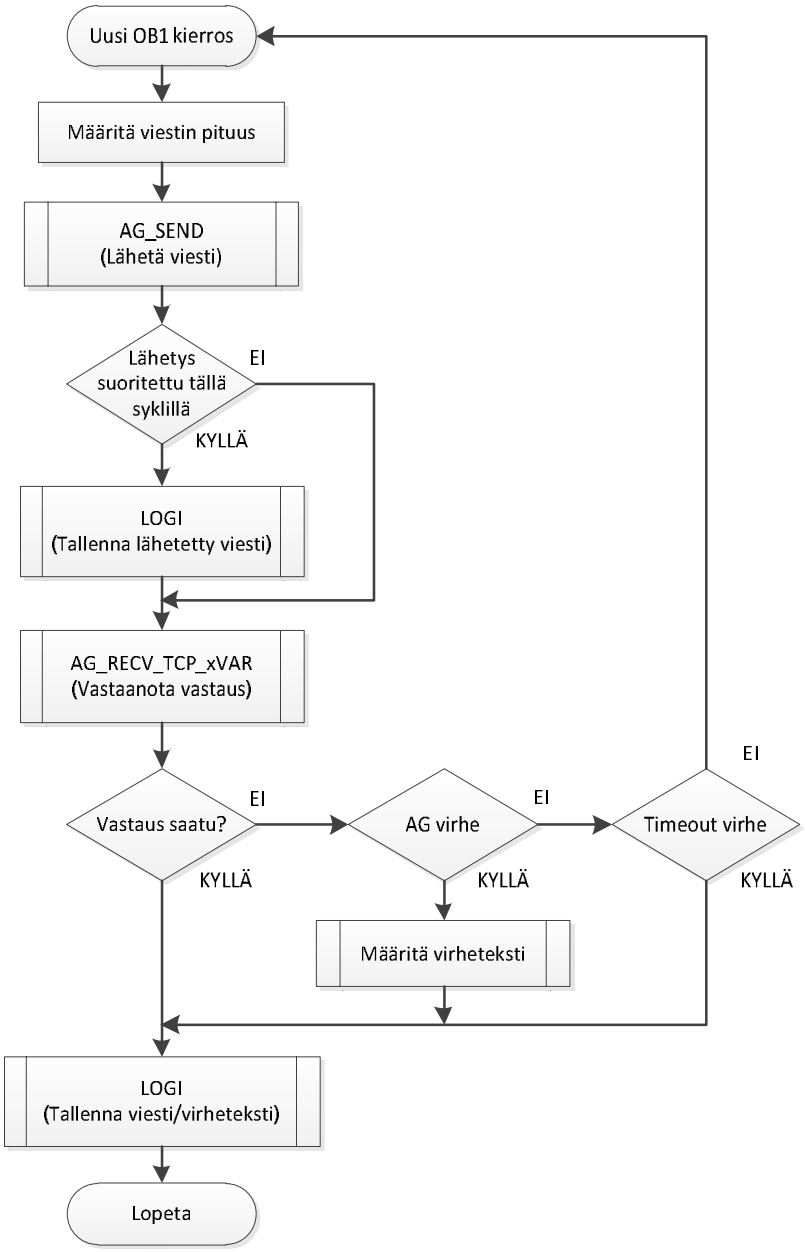
Kuviossa 9 on kuvattu yksittäisen viestin käsittely vuokaaviomuodossa. Samaa perusrakennetta käytetään jokaisen kommunikointiin käytettävän viestin osalta. Viestikohtaisesti eroava osio eroaa lähinnä vastauksen arvioinnin ja mahdollisen vastauksen sisältämän informaation käsittelyn ja tallentamisen sekä virhetilanteista seuraavien toimenpiteiden osalta.



KUVIO 9. Yksittäisen viestin käsittelyn toteutus

SEND_RECV-toimintayksikön puolella tehdään viestin lähettämisen vaatimia alustavia toimintoja ja kutsutaan AG_SEND-toimintoa, joka lähettää sille määritellyn data-alueen välivarastona toimivan tiedostoyksikön muistista. AG_SEND-toimintoa pitää aina kutsua vähintään kahdella ohjelmakierroksella ennen kuin se ilmoittaa viestin onnistuneesta lähetyksestä. Suuremmat datamäärät vaativat useampia syklejä, sillä

data pilkotaan paketeiksi. Viestin lähettämisen sekä vastauksen vastaanoton suorittavan SEND_RECV-aliohjelman toiminta on kuvattu kuviossa 10.



KUVIO 10. Viestin lähetyksen ja vastaanoton toteutus

Lähetyksen onnistuessa tallennetaan viesti LOGI-aliohjelmalla sen oheistiedostoyksikköön ja jatketaan Viewscanin vastauksen kyselemistä kommunikaatioprosessorilta.

AG_SEND-lohkon ilmoittaessa virheestä haetaan AG_SEND_ERROR-aliohjelmalta AG_SEND-lohkon tilatietoa vastaava virheteksti. AG_SEND voi ilmoittaa lähes 30 eri virheestä. Tämän jälkeen virheteksti tallennetaan LOGIin ja palautetaan virheilmoitus pääohjelmalle. Virheen jälkeinen toiminta riippuu lähetettävästä viestistä, jota on käsitelty vastauksen arviointi kappaleessa.

7.4.4 Viestin vastaanotto

Viewscanin vastaus logiikan lähettämään viestiin tulee välittömästi kyselyn jälkeen. Tästä syystä viestin saapumista pitää alkaa kyselemään CP:ltä heti logiikan viestin lähetyksen jälkeen. Tämä oli yksi syy miksi viestin vastaanotto toteutettiin samaan toimintayksikköön viestin lähetyksen kanssa. Sen myös uskottiin helpottavan AG-lohkojen virheiden arviointia ja niistä seuraavien toimintojen toteuttamista.

Data kulkee CP:n ja tietokoneen välillä sarjamuotoisena eikä viestiin sisällytetä sen pituutta, joten on mahdotonta tietää viestin päättymis- ja alkamiskohdat (Transferring data with variable message lengths via the TCP protocol n.d.). AG_RECV-lohko ei myöskään ilmoita uuden datan vastaanoton loppumisesta ennen kuin sille määritelty vastaanotetun viestin tila on täytetty kokonaisuudessaan (CPU-CPU Communication with SIMATIC Controllers 2010, 276). Viewscanilta vastaanotettujen viestien pituudet kuitenkin vaihtelevat, joten pituutta ei voida ennalta tietää. Tämän takia vastaanottamiseen käytettiin Siemensin tukisivustoilta käyttöönotettua toimintayksikköä, johon voidaan määrittää viestin päättymismerkki. Tässä tapauksessa sen käyttö oli kätevää, koska kaikki viestit päättyvät line feed -merkkiin. Logiikan ei tällöin tarvitse tietää Viewscanin lähettämän viestin pituutta, vaan se voi todeta viestin vastaanotetuksi kokonaisuudessaan kohdatessaan viestin päättymismerkin.

Viestin siirrettiin CP:ltä logiikalle AG_RECV_TCP_xVAR-lohkolla, joka käyttää AG_RECV-toimintoa datan siirtoon. Lohkon avulla siirretään CP:ltä dataa logiikan välivarastona toimivaan tiedostoyksikköön, kunnes se saadaan vastaanotettua kokonaisuudessaan. Toimintayksikön ilmoittaessa vastauksen vastaanotetuksi tallennetaan se merkkijono muotoisena LOGIin ja palautetaan pääohjelmalle arvioitavaksi.

Vastauksen kysely lopetetaan, jos määritelty odotusaika Viewscanin vastauksen saamiseksi umpeutuu tai AG_RECV ilmoittaa virhetilanteesta. Odotusajan päättymisestä

ilmoitetaan pääohjelmalle, joka taas ilmoittaa virhetilanteesta lähdöllään. AG_ RECV-lohkon ilmoittaessa virheestä haetaan AG_RECV_ERROR-aliohjelmalta AG_ RECV-lohkon tilatietoa vastaava virheteksti, joka tallennetaan LOGIin ja palautetaan virheilmoitus pääohjelmalle.

7.4.5 Vastauksen arviointi

Viewscanin vastaus kyselyyn tai mahdollinen virheilmoitus palautetaan SEND_RECV-toimintayksiköltä pääohjelmalle. Vastaus arvioidaan pääohjelmassa, koska arviointi on viestikohtaisesti yksilöllinen. Yleisesti voidaan sanoa, että viestistä tarkistetaan alkaako se ”OK:” vai ”ERR” merkeillä. Jos vastaukseen sisältyy jotakin informaatiota, kuten ohjelmakierron tilan kyselyssä, erotetaan informaatio-osio viestistä ja arvioidaan se merkkijono-toiminnoilla (ks. kuvio 11). Jos vastaus ei ole taulukossa 1 esitettyjen vastausten mukainen, tulkitaan se virheelliseksi Viewscanin toiminnaksi.

```
896                     //Normally the reply to CYCL looks like: OK:01<cr><lf> OR ERR:xxx<cr><lf>
897                     // First 3 characters of the message are saved for comparison
898                     RECV_MSG_TEMP := LEFT(IN := RECV_MSG // IN: STRING
899                             ,L := 3 // IN: INT //If L is greater than the current length of t
    he STRING variable, the input value is returned
900                             ); // STRING //With L = 0 and with a blank string as the input va
    lue, a blank string is returned
901
902                     //First 3 characters of received message are compared
903                     IF RECV_MSG_TEMP = 'OK:' THEN // Comparison operator = calls function EQ_
    STRNG, which compares two STRINGS. Returns 1 if equals
904
905                         //Separate the cycle status part from the message. Mid saves the mid
    part of the received message OK:*cut*01*cut*<cr><lf>
906                         RECV_MSG_TEMP:=MID(IN := RECV_MSG // IN: STRING
907                                            ,L :=  2// IN: INT // length from P character. 4.
     and 5. char saved
908                                            ,P :=  4// IN: INT // Start character of the stri
    ng wanted
909                                            ); // STRING
910                         //Evaluate the cycle status in string format
911                         IF RECV_MSG_TEMP = '01' THEN
912                             CYCLE_STATUS_INT := 1;
913                         ELSIF RECV_MSG_TEMP = '02' THEN
914                             CYCLE_STATUS_INT := 2;
915                         ELSIF RECV_MSG_TEMP = '03' THEN
916                             CYCLE_STATUS_INT := 3;
917                         ELSIF RECV_MSG_TEMP = '04' THEN
918                             CYCLE_STATUS_INT := 4;
919                         ELSIF RECV_MSG_TEMP = '05' THEN
920                             CYCLE_STATUS_INT := 5;
921                         ELSIF RECV_MSG_TEMP = '06' THEN
922                             CYCLE_STATUS_INT := 6;
923                         ELSE
924                             FATAL_VS_ERROR := TRUE; //Cycle status of the VS is not between 0
    1 and 06'
925                             CYCLE_STATUS_INT := 0;// Status of the VS is 0
926                         END_IF;
927
928
929                     ELSIF RECV_MSG_TEMP = 'ERR' THEN // It doesn't matter which error is rece
    ived
930                         // Statement Section_ELSIF
931                         VS_COMMUNICATION_ERR := TRUE; //VS replied error to the message
932                         INIT_OK := false; //After error VS has to be initialized to recover V
    S from error state
933                         ;
934                     ELSE;
935                         //If RECV_MSG is not TIMEOUT, OK: or ERR
936                         FATAL_VS_ERROR := TRUE; //VS reply is unknown
937                     END_IF;
```

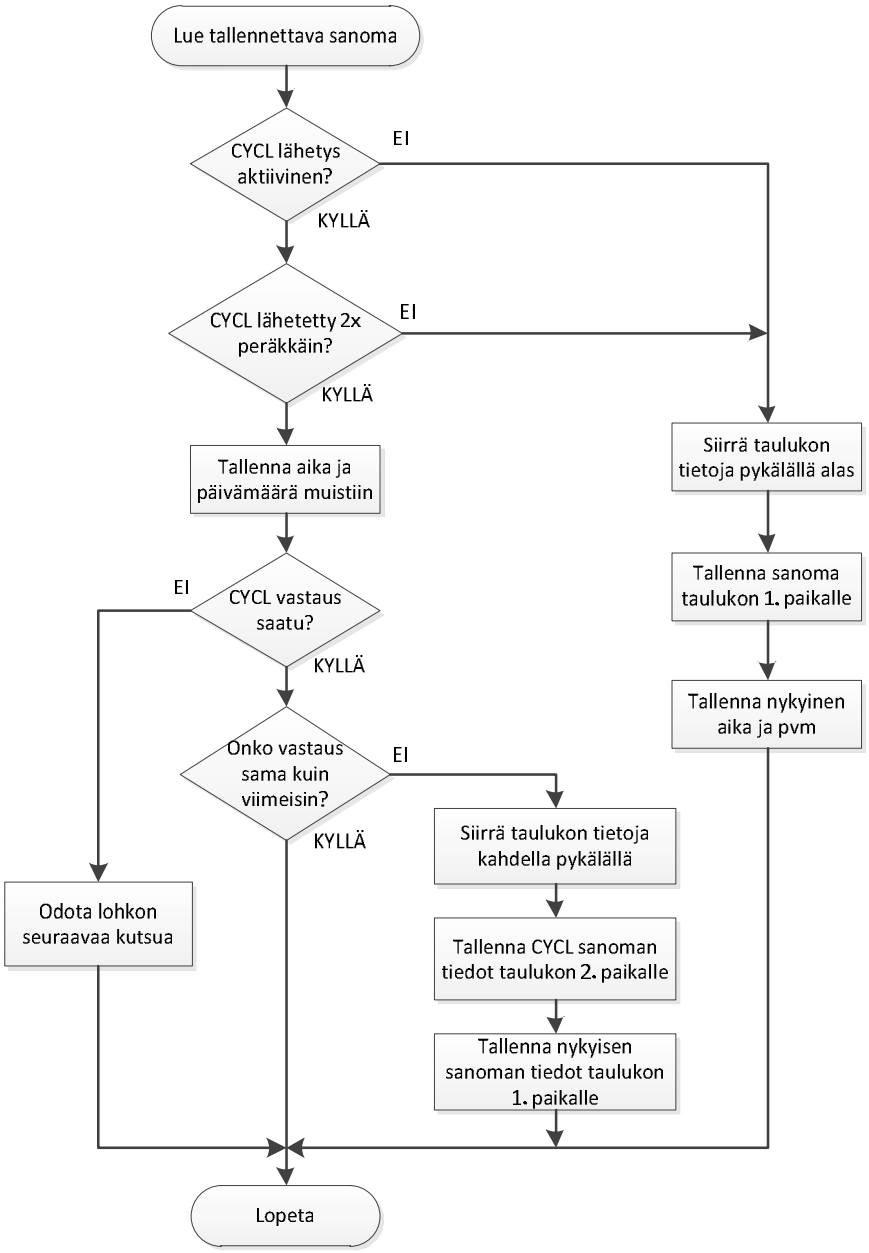
KUVIO 11. Viewscanin tilan kyselyn vastauksen arviointi

Eri viesteillä on myös erilaiset toimintatavat AG-lohkon ilmoittaessa virheestä. Koska INIT on tarkoitettu lähetettäväksi heti käynnistyksen yhteydessä, yritetään sen lähetystä toistaa muutaman kerran kymmenen sekunnin väliajoin ennen kuin ilmoitetaan virheestä. Tämä siksi, että Viewscanin ja kommunikaatioprosessorin välinen yhteys ei ole välttämättä ehtinyt muodostua esimerkiksi laitteiden äskettäisen käynnistyksen vuoksi. Muilla viesteillä ei viestin käsittelyä lopeteta yksittäisistä AG-lohkojen virheilmoituksista, vaan niitä pitää tulla useampi peräkkäisillä ohjelmakierroksilla. Viestin lähettämistä toistetaan siis useamman kerran. Tämä johtuu AG-lohkojen tilapäisistä häiriöistä, joihin ratkaisuna saattaa olla viestin lähettämisen toistaminen. Virheestä ilmoitetaan kuitenkin välittömästi kommunikointiohjelman lähdöllä, jos virhe ilmaantuu muutamalla peräkkäisellä syklillä.

7.5 Viestien ja virheilmoitusten tallennus

Mahdollisten virhetilanteiden selvittämistä varten ohjelmaan tehtiin rekisteri, joka tallentaa lähetetyt ja vastaanotetut viestit sekä normaalista poikkeavien tapahtumien ilmoitukset. Rekisteri voidaan lukea käyttöliittymälle helpottaen huomattavasti virhetilanteiden selvittelyä. Tämän ansiosta virhetilanteissa ei ole tarvetta tarkastella tietokoneen verkkoliikennettä esimerkiksi Wiresharkin tapaisella analysaattorilla. Rekisteriä varten muodostettiin LOGI-toimintayksikkö, joka tallentaa sille annetun merkkijonon oheis-tiedostoyksikköönsä. Myös tiedon tallentamishetken aika- ja päivämäärätieto tallennetaan samalla, jotta voidaan tarkastella milloin ja missä järjestyksessä tapahtumat on tallennettu. Tämä tiedostoyksikkö määriteltiin muista yksiköistä poiketen retentiiviseksi. Näin tallennetut sanomat säilyvät tallessa myös virtojen katkaisun yhteydessä.

LOGIin muodostettiin siis kaksiosainen taulukko, joka varaa koko tiedostoyksikön muistin tallennettavalle datalle. Tallennettaville viesteille määriteltiin vakiomittainen tila, johon jokainen viesti mahtuu. Tämä pienensi tallennettavien viestien määrää, mutta mahdollisti tallennusten pitämisen järjestyksessä. Käytettävästä logiikasta riippuen tiedostoyksikköön mahtuu vähintään 80 viestin tiedot. Tallennettaessa taulukon tietoja siirretään pykälällä alaspäin ja uusi tieto tallennetaan aina taulukon ensimmäiselle paikalle. Listan viimeinen tapahtumatallennus näin ollen ylikirjoitetaan. Tarkemman käsityksen tietojen tallentamisesta saa kuvion 12 vuokaaviosta.

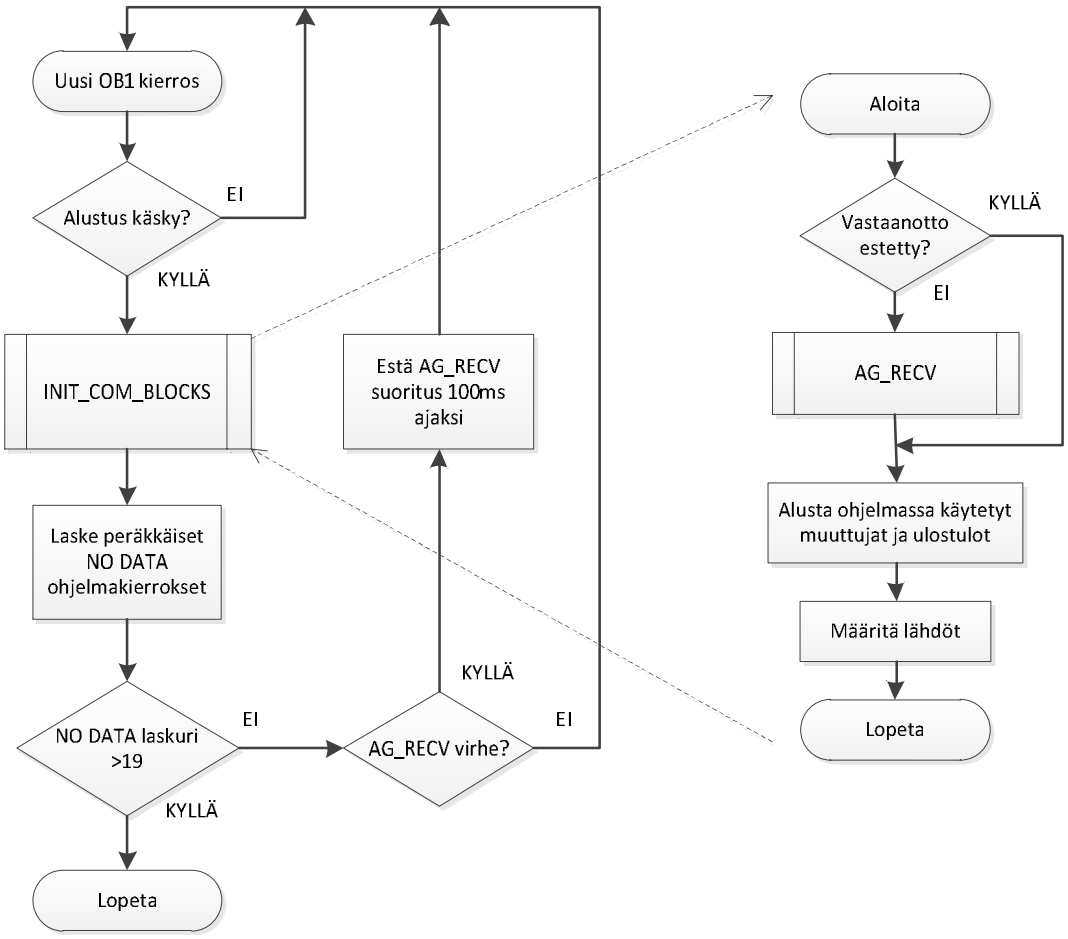


KUVIO 12. Viestien ja virheilmoitusten tallennuksen toteutus

Viewscanin tilaa kysellään jatkuvasti, minkä takia CYCL viestit ja niiden vastaukset olisivat täyttäneet LOGIn melko nopeasti. Tämän takia CYCL viestiä eikä siihen saatua vastausta tallennettu, jos CYCL oli myös viimeiseksi taulukkoon tallennettu viesti sekä Viewscanin tila oli pysynyt samana.

7.6 Ohjelman alustus

Ohjelman alustusta ja virhetilanteiden kuittausta varten ohjelmaan tehtiin sisäinen alustustoiminto, joka aktivoidaan kommunikointiohjelmaan muodostetulla alustustulolla. Toiminto nollaa yksittäin kaikki kommunikointilohkoissa käytetyt staattiset muuttujat ja lähdöt alkutilaan. Samalla myös tyhjennetään kommunikaatioprosessorin puskuri. Puskuriin voi jäädä tietoja esimerkiksi jos yhteys logiikan ja Viewscanin välillä katkeaa, ennen kuin tieto luetaan kommunikaatioprosessorista ulos. Puskuri pitää virhetilanteissa tyhjentää, sillä seuraavan viestin vastaus voidaan tulkita virheelliseksi puskurissa olleen vanhan datan vuoksi. Kuviossa 13 on selvennetty alustuksen toimintaa pääohjelmasta lähtien.



KUVIO 13. Kommunikointiohjelman alustuksen toteutus

Puskurin tyhjennykseen käytettiin AG_RECV-lohkoa, jolla data luettiin yhden tavun mittaisina pätkinä eli merkki kerrallaan tiedostoyksikköön, kunnes voitiin olettaa puskurin olevan tyhjä. Tähän käytettiin pelkkää AG_RECV-lohkoa, koska datassa ei välttämättä ole normaalien vastaanottojen tapaan käytettyä katkaisumerkkiä, johtuen esimerkiksi yhteyden katkeamisesta tietyllä hetkellä. AG_RECV-lohkossa ei ole tilatietoa, joka ilmoittaisi puskurin olevan tyhjä. Sen sijaan lohkolla on tila, joka ilmoittaa jos dataa ei ole vielä saatavilla. Tämä ei kuitenkaan aina tarkoita puskurin olevan tyhjä, vaan voi esiintyä myös kun CP ei ole vielä valmis antamaan dataa logiikalle. Tästä johtuen ohjelma lopettaa tyhjennyksen, kun riittävän monta 'No data available yet' -tilailmoitusta saadaan peräkkäin.

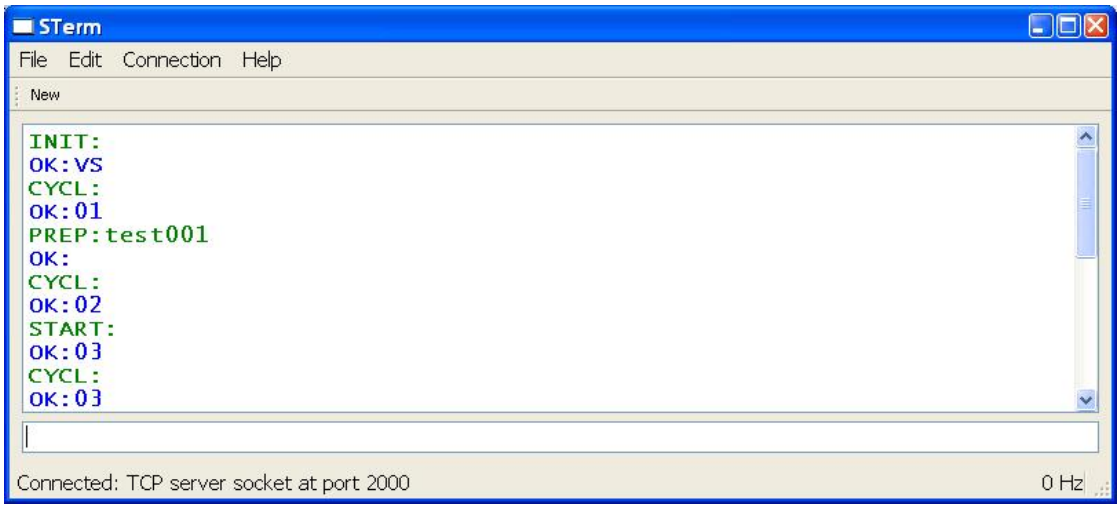
Virheen ollessa päällä, esimerkiksi jos yhteyttä logiikan ja tietokoneen välillä ei ole, estettiin vastaanotto 0,1 sekunnin ajaksi AG_RECV-lohkon käyttöohjeiden mukaisesti. Puskurissa olevaa dataa ei saada luettua pois virheen ollessa päällä, mistä syystä myöskään alustusta ei lopeteta ennen kuin virhe poistuu ja puskuri saadaan tyhjennettyä.

7.7 Ohjelman testaus

Ohjelman toimintoja testattiin jo sen toteutusvaiheen aikana. Kommunikointiohjelmaa testattiin suoraan logiikassa, minkä ansiosta vältyttiin simuloinnilta. Näin pystyttiin varmistamaan, että kaikki toiminnot toimivat varmasti myös oikeassa logiikassa ja järjestelmässä. Testauksessa käytettiin Siemensin CPU 315-2 PN/DP -logiikkaa, CP 343-1 Lean -kommunikaatioprosessoria sekä kytkimillä ja merkkivaloilla varusteltua simulointiyksikköä. Simulointiyksiköstä oli mahdollista valita käyttääkö sitä 16 tulona, 16 lähtönä vai 8 tulona ja 8 lähtönä.

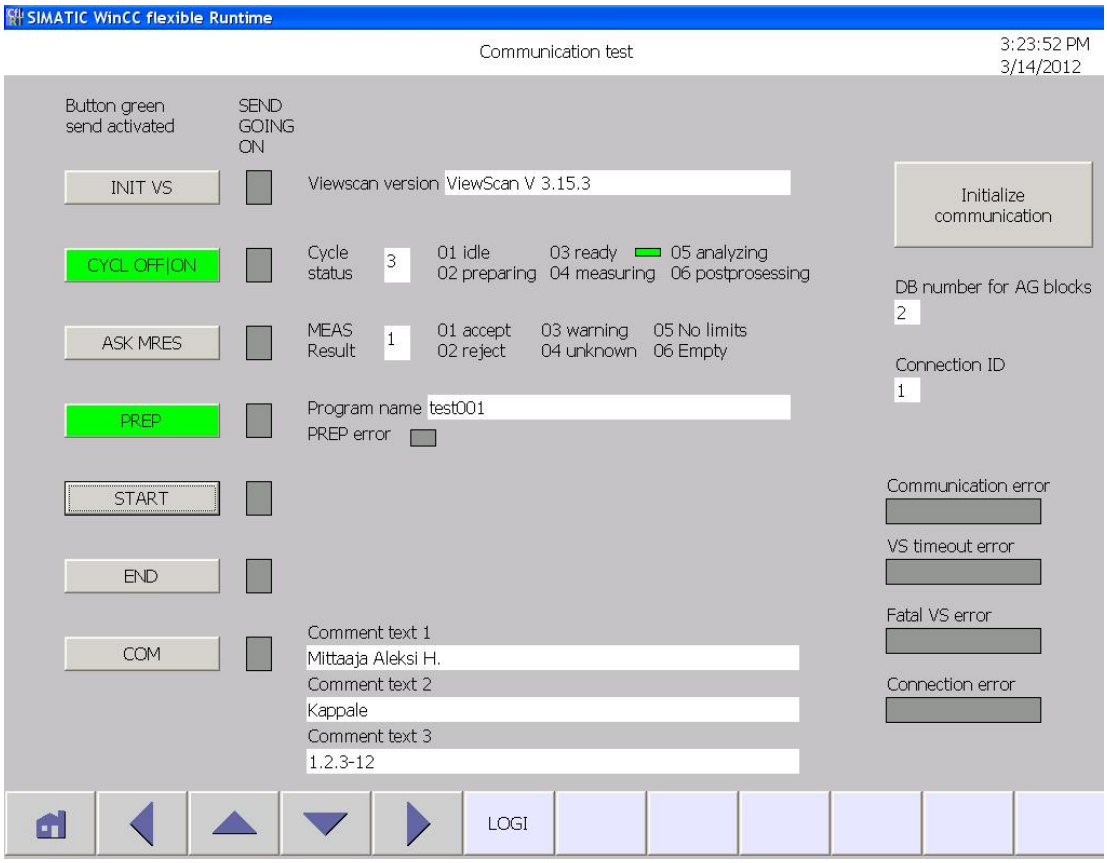
Alkuvaiheessa tiedonsiirron tarkkailuun käytettiin Wireshark-ohjelmaa, jolla pystyttiin tarkistamaan lähetettyjen viestien oikea muoto. Kun viestit oli saatu oikean muotoisiksi lopetusmerkkeineen, voitiin siirtyä käyttämään apuna STerm-ohjelmaa. Ohjelma on Stresstechin oma sovellus, jolla voidaan vastaanottaa ja lähettää dataa erilaisten yhteyksien kautta. Ohjelmaan voidaan määritellä tarkkailtava yhteys ja käytettävät viestien katkaisumerkit. STermillä voitiin testata logiikkaohjelman toimivuutta vastaamalla logiikan kyselyihin erilaisilla sanomilla. Tietokoneen logiikalta vastaanot-

tamat viestit sekä käyttäjän logiikalle lähettämät viestit tallentuvat kätevästi ohjelman tekstikenttään (ks. kuvio 14).



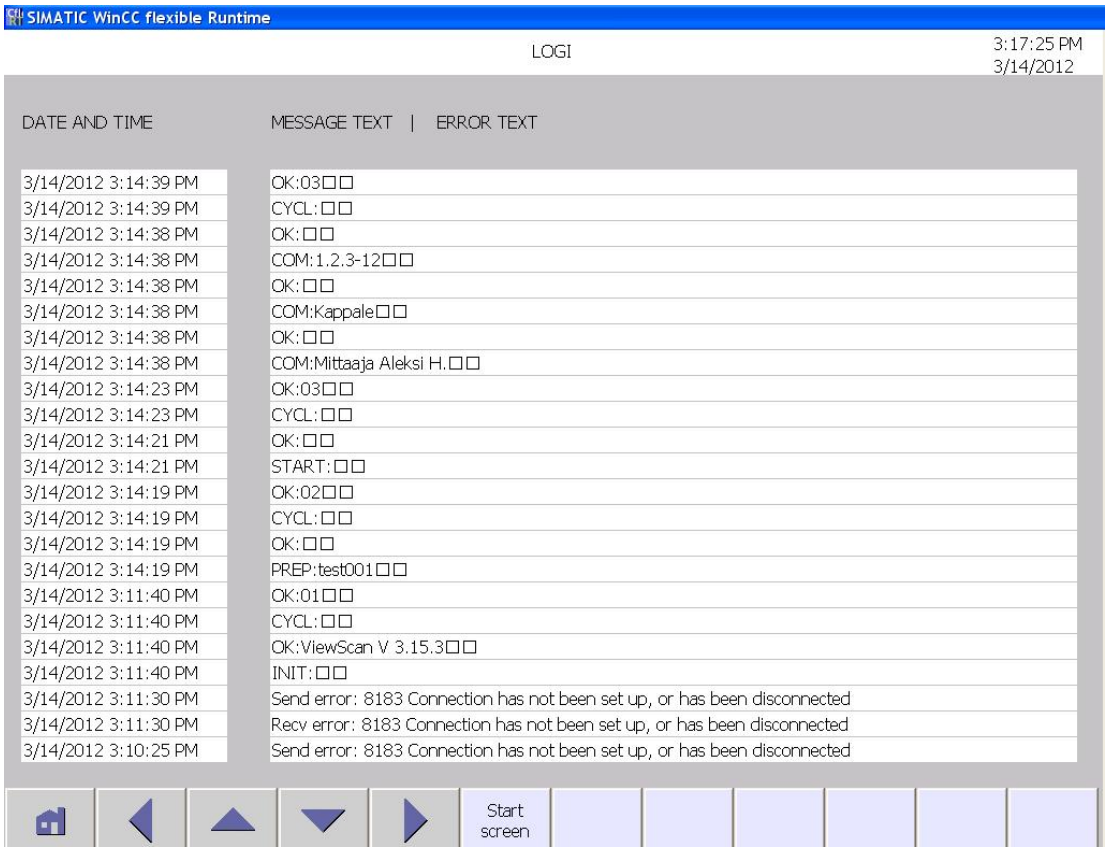
KUVIO 14. STerm-ohjelman käyttöliittymä

Alkuvaiheessa testaukseen käytetyn simulointiyksikön kytkinten ja merkkivalojen merkitykset menivät helposti sekaisin. Merkkivaloja ei myöskään ollut tarpeeksi kaiken informaation näyttämiseksi. Testauksen helpottamiseksi ja lopputestejä varten rakennettiin WinCC:llä testausympäristö. Näin saatiin mahdollisimman paljon toimintoja ja informaatiota yhdistettyä samaan käyttöliittymään. Testauksesta saatiin näin huomattavasti havainnollisempi ja voitiin olla varmoja kullekin viestille kuuluvista toiminnoista. Testausympäristöllä voitiin kätevästi ohjata logiikkaa lähettämään halutut viestit, myös valmistelu- ja kommenttitekstien kanssa, sekä seurata Viewscanin tilaa (ks. kuvio 15).



KUVIO 15. WinCC-testausympäristö ohjelman toimintojen testaamista varten

Merkkijonojen monitorointi STEP 7 ohjelman toiminnoilla oli melko työlästä eikä kovin havainnollistavaa. Merkkijonoja ei voinut tarkastella S7-SCL debuggauksen yhteydessä, vaan ne piti lukea auki VAT-tiedostossa merkki kerrallaan. Viestien seuraamista varten testausympäristöön tehtiin oma sivu, jossa ison LOGI-taulukonkin seuraaminen kävi kätevästi (ks. kuvio 16). Näin voitiin helposti seurata lähetettyjä ja vastaanotettuja viestejä sekä ilmaantuneita virheilmoituksia. Kuviossa 16 näkyvät mittauksen valmistelun vaatimat viestit, joiden jälkeen voidaan suorittaa haluttu määrä mittauksia. Alussa olevat virheilmoitukset johtuvat Viewscanin käynnistämisestä vasta ensimmäisen viestin lähetyksen jälkeen. Viestien perässä näkyvät neliöt ovat viesteissä käytetyt päätemerkit, jotka eivät jostakin syystä näkyneet testausympäristön puolella oikein.



KUVIO 16. WinCC-testausympäristö viestien tarkastelua varten

Testattavista toiminnoista tehtiin testaussuunnitelma ennen lopputestausta. Näin varmistuttiin, että epänormaalit tilanteetkin tuli testattua. Testaus voitiin näin tehdä systemaattisesti, eikä ohjelman testaaminen jäänyt vain satunnaisiin kokeiluihin. Systemaattisen testauksen apuna käytettiin STerm-ohjelmaa, jolloin voitiin muun muassa vastata ennalta arvaamattomilla viesteillä logiikan kyselyihin tai jättää kokonaan vastaamatta. Ohjelman avulla virheiden simulointi oli kätevää ja varmistuttiin ohjelman toimiminen myös harvinaisemmissa tilanteissa. AG-lohkoille myös simuloitiin virheitä muun muassa vaihtamalla DB:n osoite ja yhteysparametrit vääriksi. Ohjelman käyttäytymistä testattiin myös sammutettaessa Viewscan ja irroitettaessa ethernet-kaapeli erilaisissa tilanteissa. Logiikan ja WinCC:n yhteys muodostettiin logiikan MPI-yhteyden (Multipoint interface) kautta, jolloin niiden välinen yhteys pysyi aina toimivana.

Testaamisen yhteydessä löydettiin luonnollisesti vielä puutteita kommunikointiohjelmasta, mutta ne pystyttiin korjaamaan. Kommunikointiohjelmaa käytettiin tietysti

myös paljon Viewscan ja Rollscan kytkettyinä ja samalla simuloitiin mittauksia testianturilla. Näin pystyttiin toteamaan myös hyväksymisrajojen ja kommenttien toimiminen. Viewscanin uusimmasta versiosta löydettiin testauksen yhteydessä myös joitakin puutteita, sekä alkutiedoista poikkeavia toimintoja. Puutteet olivat pieniä, jotka olivat helposti ohjelmistopuolen henkilöiden korjattavissa. Poikkeavat toiminnot olivat myös suhteellisen pieniä, vaatien joitakin muutoksia tehtyyn koodiin.

Ohjelman toiminnallisuuden testaamiseksi rakennettiin myös mittaussekvenssiä muistuttava ohjelma. Tämä tehtiin S7-GRAPH-ohjelmointikielellä, jolla saadaan sekvenssityylisistä ohjelmista rakenteiltaan selkeitä. Kommunikointiohjelman ympärille ei kuitenkaan rakennettu täydellisesti mittaustapahtumaa muistuttavaa ohjelmaa. Mittaustapahtuma on lähes aina hyvin erilainen, minkä takia ei katsottu järkeväksi kuluttaa aikaa tällaisen tekemiseen. Testillä pystyttiin kuitenkin osoittamaan, että yksinkertainen mittaussekvenssi, jonka käynnistäminen onnistuu napin painalluksella, voidaan toteuttaa melko helposti.

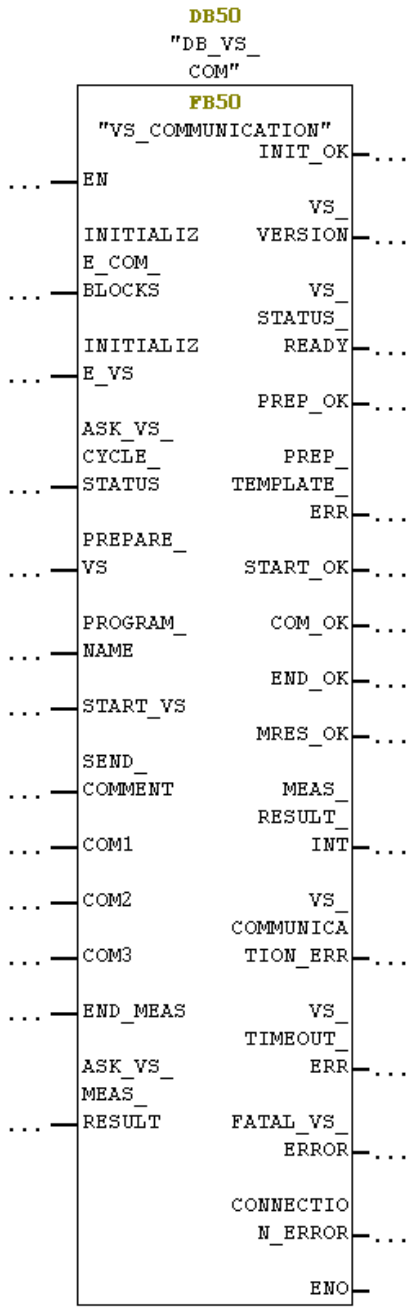
8 OPINNÄYTETYÖN TULOKSET

Opinnäytetyön tuloksena saatiin aikaan Siemens S7-300-sarjan logiikoissa käytettävä ohjelma, jolla mahdollistetaan logiikan ja Viewscan-tiedonkeruuohjelman välinen kommunikointi. Ohjelma ei kuitenkaan toimi itsenäisesti vaan se tarvitsee ympärilleen ohjelman, joka määrittelee, mitä viestejä halutaan milloinkin lähettää. Ohjelmasta saatavien lähdöillä voidaan varmistaa haluttu toiminta ja suorittaa halutut toiminnot virhetilanteen ilmaantuessa, kuten pysäyttää liikkeet ja lopettaa mittaus. Virheestä riippuen tulee virhe kuitata ennen uuden mittauksen yritystä. Ohjelman käyttö jätettiin tarkoituksellisesti melko joustavaksi, koska haluttu toiminta on yleensä sovelluskohtainen. Aina toteutettavat toiminnot rakennettiin tehtyyn ohjelmaan kuitenkin sisäisesti.

Kommunikointitoimilohkoa tulee käydä syklisesti läpi logiikan ohjelmassa. Toimilohkolla voidaan lähettää halutut viestit, esimerkiksi logiikan merkkereillä ohjattuna. Toimilohkon ohjaamiseksi CNC-ohjaimesta käytetään sen omia käskyjä, joilla se voi

lukea ja kirjoittaa logiikan muistialuetta. Tätä muistialuetta taas voidaan käyttää kommunikointiohjelman ohjaamiseen.

Logiikasta kutsuttavan kommunikaatiosovelluksen lohkokaavio-muotoinen toimilohko on esitetty kuviossa 17. Lohko pitää sisällään aikaan saadun ohjelman kokonaisuudessaan, ja siitä on nähtävissä käytettävät tulot ja lähdöt. Tarkemmat selvennykset kommunikointiohjelman tuloista on esitetty taulukossa 2 ja lähdöistä taulukossa 3.



KUVIO 17. Tehdyn ohjelman syklisesti kutsuttava toimilohko

TAULUKKO 2. Kommunikointiohjelman tuloparametrit selvennyksineen

Tuloparametri	Datatyyppi	Parametrin tarkoitus
INITIALIZE_COM_BLOCKS	BOOL	**Kommunikointiohjelman alustus.** Ohjelmassa on sisäinen toiminto, joka alustaa kommunikoinnissa käytettävät muuttujat ja lähdöt alkutilaan. Virheet nollataan tällä.
INITIALIZE_VS	BOOL	**Viewscanin alustus.** Lähettää viestin, joka alustaa VS:n. Tällä saadaan VS:n versio ohjelman lähtöön. Tällä myös palautetaan VS virhetilanteesta.
ASK_VS_CYCLE_STATUS	BOOL	**VS:n ohjelmakierron kysyminen.** Liipaistaan syklisesti n. 500..2000 ms välein esimerkiksi logiikan kellopulssilla.
PREPARE_VS	BOOL	**VS:n valmistelu mittausta varten.** Ohjelman nimi lähetetään tämän viestin mukana.
PROGRAM_NAME	STRING	**Mittausohjelman nimi.** Yhdistetään PREP viestiin lähetettäessä se.
START_VS	BOOL	**Mittauksen aloitus.** VS avaa valmisteltavana olleen mittauspohjan. Siirtyy odottamaan mittauksen aloitusliipaisua tai kommentteja.
SEND_COMMENT	BOOL	**Kommenttien lähetys.** Tällä lähetetään COM1..COM3 tulojen kommentit peräkkäin VS:n aktiivisen mittauspohjan kommenttikenttään.
COM1	STRING	**Lähetettävä kommentti.** Maksimissaan 80 merkkiä pitkä.
COM2	STRING	**Lähetettävä kommentti.** Maksimissaan 80 merkkiä pitkä.
COM3	STRING	**Lähetettävä kommentti.** Maksimissaan 80 merkkiä pitkä.
END_MEAS	BOOL	**Mittauksen lopetus.** Haluttu määrä mittauksia on suoritettu.
ASK_VS_MEAS_RESULT	BOOL	**Mittaustuloksen kysyminen** mittauksen päätyttyä. Tulos tuodaan lähtöön integer-muotoisena.

TAULUKKO 3. Kommunikointiohjelman lähtöparametrit selvennyksineen

Lähtöparametri	Datatyyppi	Parametrin tarkoitus
INIT_OK	BOOL	**VS alustettu onnistuneesti.** Tämä on päällä kun INIT on lähetetty onnistuneesti ja nollautuu jos VS vastaa virheilmoituksella johonkin viestiin.
VS_VERSION	STRING	**Viewscanin versio.** Voidaan tallentaa ja näyttää esimerkiksi käyttöliittymässä.
VS_STATUS_READY	BOOL	**VS ja mittalaite on valmis mittauksen aloitukseen.** VS:n tila on tosin 'Ready' myös lopetettua mittaus.
PREP_OK	BOOL	**Valmistelukäsky vastaanotettu.** Voimassa vain yhden ohjelmakierroksen ajan.
PREP_TEMPLATE_ERR	BOOL	**Template virhe.** PROGRAM_NAME pohjaa ei löydy. Päällä kunnes PREP lähetetään onnistuneesti tai suoritetaan ohjelman alustus.
START_OK	BOOL	**Aloituskäsky vastaanotettu.** Voimassa vain yhden ohjelmakierroksen ajan.
COM_OK	BOOL	**Kaikki kommentit vastaanotettu.** Voimassa vain yhden ohjelmakierroksen ajan.
END_OK	BOOL	**Lopetuskäsky vastaanotettu.** Voimassa vain yhden ohjelmakierroksen ajan.
MRES_OK	BOOL	**Mittaustuloksen kysely vastaanotettu.** Voimassa vain yhden ohjelmakierroksen ajan.
MEAS_RESULT_INT	INTEGER	**Mittaus tulos integer muotoisena.** Voimassa vain yhden ohjelmakierroksen ajan.
VS_COMMUNICATION_ERR	BOOL	**Kommunikointivirhe.** VS vastasi virheilmoituksella logiikan kyselyyn. Tämän jälkeen ohjelma lähettää automaattisesti INIT viestin, mikä palauttaa VS:n virhetilanteesta ja resetoi virhelähdön.
VS_TIMEOUT_ERR	BOOL	**Ei vastausta.** VS ei vastannut kyselyyn timeout ajan sisällä (10s). Voi johtua yhteyden katkeamisesta tai VS:n virheestä. Virhe resetoidaan ohjelman alustuksella.
FATAL_VS_ERROR	BOOL	**Väärä vastaus.** VS vastasi ennalta määrittämättömällä sanomalla. Virhe resetoidaan ohjelman alustuksella.
CONNECTION_ERROR	BOOL	**Yhteysvirhe.** AG-lohkojen ilmoittama virhe. Voi johtua mm. yhteyden katkeamisesta tai konfiguraation puuttumisesta. Virhe resetoidaan ohjelman alustuksella.

9 POHDINTA

Opinnäytetyön tarkoituksena oli luoda luotettava ja helposti käytettävä kommunikointiohjelma käytettäväksi logiikan ja Viewscanin väliseen kommunikointiin. Rajapinnan sitä käyttävälle ohjelmalle tuli olla mahdollisimman yksinkertainen. Opinnäytetyön tuloksena syntynyt logiikkaohjelma oli testausten perusteella varsin onnistunut. Ohjelman tuloilla haluttujen viestien lähettäminen on helppoa ja lähdöillä voidaan varmistaa kommunikoinnin oikea toiminta. Mahdollisten virhetilanteiden selvittämistä helpottavat viestien ja virheilmoitusten tallennus. Ohjelman rajapinnan uskotaan olevan hyvä, mutta todellisuudessa asia varmistuu, kun ohjelma otetaan käyttöön ensimmäisessä sovelluksessa kesän 2012 aikana.

Kommunikointisovelluksen ohjelmointikieleksi valittu S7-SCL osoittautui varsin päteväksi työkaluksi. Erilaisten ehtojen ja valintarakenteiden toteuttaminen onnistui sillä kätevästi. Jotkin toiminnot olivat kuitenkin normaalia hankalampia toteuttaa valitulla ohjelmointikielellä. Erityisesti hankaluuksia tuottivat tiedostoyksikön muistialueen osoittaminen AG-lohkoille sekä reunojen tunnistukset liipaisuja varten.

Ohjelman siirrettävyyttä toiseen ohjelmointiympäristöön tekstimuotoinen ohjelma helpottaa huomattavasti. Ohjelmoinnissa pystyttiin myös suurimmilta osin käyttämään standardin mukaisia toimintoja, jotka toimivat monen muunkin valmistajan logiikoissa. Viestien lähetys ja vastaanotto pitää kuitenkin rakentaa luultavammin uudestaan, sillä tähän käytettiin Siemensin omia AG_SEND- ja AG_RECV-toimintoja. Lähetys ja vastaanotto on kuitenkin rakennettu omaan aliohjelmaansa, jolloin toiseen ohjelmointiympäristöön siirryttäessä voidaan rakentaa samantapainen aliohjelma käytettävissä olevilla tiedonsiirtotoiminnoilla.

Viestien lähetys ja vastaanotto päätettiin tehdä samaan toimintayksikköön jo alkuvaiheessa. Tämän uskottiin helpottavan AG-lohkojen virheiden arviointia ja niistä seuraavien toimintojen toteuttamista. Tätä ratkaisu helpottikin, mutta ohjelman selkeyden ja ymmärrettävyyden kannalta olisi voinut olla järkevämpää tehdä omat aliohjelmat lähetykselle ja vastaanotolle. Tämä olisi tosin voinut tehdä pääohjelmassa tapahtuvan aliohjelmien käsittelyn sekä virheiden arvioinnin monimutkaisemmaksi.

Virheilmoitusten tallennusten ansiosta voidaan ohjelman käyttöönoton yhteydessä helposti havaita mahdolliset yhteyden konfiguraatiovirheet. Myös esimerkiksi kommunikaatioprosessorin vikaantuminen voidaan todentaa virheilmoitusten perusteella. LOGI osoittautui myös erittäin hyödylliseksi työkaluksi testattaessa ohjelman toimintaa. Sillä havaittiin kätevästi ilmaantuneiden virheiden syyt, jolloin virheiden poistaminen helpottui. Virheiden selvittely olisi ilman rekisteriä ollut hyvin työlästä, ja varmasti jotkin pienemmät virheet olisi jäänyt kokonaan löytämättä.

LOGIn toteutus ei kuitenkaan välttämättä ollut paras mahdollinen. Tosin nyt tiedostoyksikössä olevat tapahtumat ovat tallessa ja tarvittaessa ne voidaan lukea monesta paikasta. LOGI olisi voitu myös tehdä niin, että viestit olisi lähetetty suoraan WinCC:n käyttöliittymälle hälytysten tallennustoiminnolla. Tämä olisi mahdollistanut suurempien viestimäärien tallentamisen. Tallennettuja viestejä ei kuitenkaan tarvitse tarkistella, kuin lähinnä käyttöönoton ja virhetilanteen yhteydessä. Virhetilanteet ovat kuitenkin käyttöönotetussa järjestelmässä harvinaisia ja silloinkin virheen aiheuttaja selviää viimeisimmistä viesteistä, jolloin suuren viestimäärän tallentaminen ei ollut välttämätöntä.

Samalla logiikalla on tarvittaessa mahdollista ohjata kahtakin Viewscan-ohjelmaa molempien kerätessä samanaikaisesti tietoa omalta mittalaitteeltaan. Kommunikointiohjelman päätason toimintayksikköä voidaan kutsua useampaan kertaan organisaatioyksikössä siten, että määritetään toimintayksiköille omat parametrit sekä oheistiedostoyksiköt. Vaikka tämä onkin mahdollista, on tällöin ohjelman asetuksia ja aliohjelmien nimet muokattava jokaiselle yhteydelle yksilöllisiksi, jotta tarvittavat toiminnot löytyvät. Ominaisuuden hyödyntäminen olisi voitu tehdä helpommaksi tekemällä ohjelma hieman eri tavalla. Ominaisuudelle ei kuitenkaan ollut tarvetta työtä tehtäessä, ja idea tuli ilmi työn myöhäisessä vaiheessa, joten sen toteuttamiseen ei ryhdytty.

Kun Viewscanista tehdään tulevaisuudessa laajempi päivitys, tulisi ohjelman toiminta myös osana automaattista mittausjärjestelmää suunnitella uudelleen. Tiedonsiirto ja kommunikointiin käytettävät viestit tulisi miettiä uudelleen, sillä nykyisen protokollan mukaisen kommunikoinnin toteutus logiikalle on varsin työläs. RS232-liikennöinnin sijasta voitaisiin miettiä RS485-tiedonsiirron hyödyntämistä, jolloin

kommunikointi profibuksen välityksellä voisi tulla mahdolliseksi. Yhtenä vaihtoehtona voisi olla myös profinetin tai OPC-serverin hyödyntäminen.

Oma ammatillinen osaaminen kehittyi projektin myötä logiikkaohjelmoinnin osalta huomattavasti. Käytetty ohjelmointikieli ei ollut minulle entuudestaan lainkaan tuttu, joten sen ominaisuudet ja salat piti selvittää työn tekemisen myötä. Opittu kieli on hyvin tehokas, joten sille tulee varmasti työelämässä käyttöä. En ole aiemmin myöskään juuri hyödyntänyt mahdollisuutta jakaa ohjelmaa useaan tietyn toiminnon suorittavaan yksikköön. Työn myötä huomasin tämän selkeyttävän ohjelman rakennetta huomattavasti.

LÄHTEET

About us. n.d. Artikkeli Stresstech Groupin sivustolla. Viitattu 26.4.2012. http://www.stresstechgroup.com/content/en/11501/13/13.html

Ahola, A. & Sundell, L. 2006. Tietokonekommunikaatio. 4. uud. p. Helsinki: Opetushallitus.

Asmala, H., Koskinen, K., Koskela, M., Mätäsniemi, T., Soini, A., Strömman, M., Tommila, T. & Valkonen, J. 2005. Automaatiosovellusten ohjelmistokehitys: suunnittelun työtavat, välineet ja sovellusarkkitehtuurit. Helsinki: Suomen automaatioseura.

Caro, D. 2009. Automation network selection: a reference manual. 2nd ed. Research Triangle Park (NC): ISA - International Society of Automation.

CPU-CPU Communication with SIMATIC Controllers. 2010. Communication manual. Siemens AG. Viitattu 9.3.2012. http://support.automation.siemens.com/WW/view/en/20982954.

Crispin, A. 1997. Programmable logic controllers and their engineering applications. 2nd ed. New York: McGraw-Hill Publishing Company.

IEC 61131-3. 2003. Programmable controllers – Part 3: Programming languages. 2nd edition. Geneva: International Electrotechnical Commission.

Keinänen, T., Kärkkäinen, P., Lähetkangas, M. & Sumujärvi, M. 2007. Automaatiojärjestelmien logiikat ja ohjaustekniikat. Helsinki: WSOY.

Pigan, R. & Metter, M. 2006. Automating with PROFINET. Erlangen: Publicis Corporate Publishing.

PLCopen. n.d. Artikkeli PLCopenin sivustolla. Viitattu 9.3.2012. http://www.plcopen.org/.

Program blocks for SIMATIC NET S7 CPs. 2011. Programming manual. Siemens AG. Viitattu 5.3.2012. http://support.automation.siemens.com/US/view/en/30564821.

S7-SCL V5.3 for S7-300/400. 2005. S7-SCL Programming manual. Siemens AG. Viitattu 9.3.2012.
http://cache.automation.siemens.com/dnl/zMxOTcwMwAA_5581793_HB/SCL_e.pdf

Seppälä, J. 2008. Teollisuus-Ethernet-väylät. Teoksessa Teollisuusautomaation tiedonsiirtoliikenne: turvaväylät. Toim. M. Sundquist. Espoo: Inspecta koulutus.

SIMATIC S7-300 Communication. n.d. Tuoteinformaatio. Siemens AG. Viitattu 10.2.2012. http://www.automation.siemens.com/mcms/programmable-logic-controller/en/simatic-s7-controller/s7-300/communication/Pages/Default.aspx.

SIMATIC S7-SCL. n.d. Ohjelmointikielen kuvaus. Siemens AG. Viitattu 4.3.2012. http://www.automation.siemens.com/mcms/simatic-controller-software/en/step7/simatic-s7-scl/Pages/Default.aspx.

Software ViewScan. n.d. Artikkeli Stresstech Groupin sivustolla. Viitattu 9.3.2012. http://www.stresstech.fi/content/en/11501/116/116.html.

Stresstech Oy. 2011. Kuva Camscan 300 -mittausjärjestelmästä Stresstech Oy:n kuva-arkistosta.

Tasekirja. 2012. Stresstech Oy:n vuoden 2011 tasekirja.

TC3-Certification of IEC 61131-3 Environments. n.d. Artikkeli PLCopenin sivustolla. Viitattu 9.3.2012. http://www.plcopen.org/pages/tc3_certification/.

Transferring data with variable message lengths via the TCP protocol. n.d. Artikkeli Siemensin tukisivustolla. Siemens AG. Viitattu 3.4.2012. http://support.automation.siemens.com/WW/view/en/19033929.