**Anton Malviniemi**

# TIA PORTAL TYÖKALUJEN SOVELTAMINEN APEX AUTOMATION OY:N PROJEKTOINNISSA

**Opinnäytetyö**
**CENTRIA AMMATTIKORKEAKOULU**
**Tietotekniikan koulutusohjelma**
**Syyskuu 2014**

## TIIVISTELMÄ OPINNÄYTETYÖSTÄ

| **Yksikkö**<br>Kokkola-Pietarsaari | **Aika**<br>Syyskuu 2014 | **Tekijä/tekijät**<br>Anton Malviniemi |
|---|---|---|
| **Koulutusohjelma**<br>Tietotekniikan koulutusohjelma | | |
| **Työn nimi**<br>TIA Portal työkalujen soveltaminen Apex Automation Oy:n projektoinnissa | | |
| **Työn ohjaaja**<br>Hannu Ala-Pöntiö | | **Sivumäärä**<br>74+5 |
| **Työelämäohjaaja**<br>Tuomo Känsäkangas | | |


Tämän opinnäytetyön aiheena oli tutkia Siemensin kehittämää TIA Portal -ohjelmistoa ja uudempia Siemensin logiikoita. Tavoitteena oli luoda hyödyllisiä esimerkkiohjelmia TIA Portalilla Apex Automation Oy:n käyttöön.

Tietoperustassa on selvitetty Siemensin suosittelemia ohjelmointitapoja 1200- ja 1500-logiikoiden ohjelmointiin. Samalla tutustutaan myös GRAPH-ohjelmointikieleen ja miten HTML-sivuja voidaan luoda logiikalle, joka on määritelty verkkopalvelimeksi. Teoriaosuudessa kerrotaan myös Profinet kommunikointi standardista ja teollisuuden langattomista lähiverkoista. CU250S-2 ohjausyksikön ja taajuusmuuttajan käyttöönotolle on omistettu oma lukunsa, koska yksi esimerkkiohjelmista liittyy ohjausyksikköön.

Esimerkkiohjelmien esittely ja niiden toiminnan kuvaaminen ovat viimeisen luvun aiheita. Opinnäytetyön tuloksena syntyi kahdeksan ohjelmaa, jotka on tehty TIA Portalilla. Kaikissa ohjelmissa on käytetty 1513-1 PN -logiikkaa ja kommunikointitapana Profinettiä. Esimerkkiohjelmat liittyvät logiikoiden väliseen kommunikointiin, HTML-sivuihin, resepteihin, data log -tiedostoihin, IWLAN-verkon määrittelyyn, kellonaikojen synkronointiin, Profinet IO -laitteen deaktivointiin ja paikoittamiseen käyttäen CU250S-2 -ohjausyksikköä. Tutkimustyön ja ohjelmien tekemisen tuloksena paransin omaa ammattitaitoani ja sain lisää tietoa tulevaa uraani varten.



**Asiasanat**
GRAPH, HTML, ohjelmoitava logiikka, IWLAN, TIA Portal, paikoitus, Profinet, taajuusmuuttaja

## ABSTRACT

| **Unit** Kokkola-Pietarsaari | **Date** September 2014 | **Author/s** Anton Malviniemi |
|---|---|---|
| **Degree programme** Information technology | | |
| **Name of thesis** Applying TIA Portal tools in the projects of Apex Automation Oy | | |
| **Instructor** Hannu Ala-Pöntiö | | **Pages** 74+5 |
| **Supervisor** Tuomo Känsäkangas | | |


The subject of this thesis was to research the TIA Portal software and newer logic controllers which were both developed by Siemens. The aim was to create useful example programs with TIA Portal for the use of Apex.

In the theory part of this thesis the programming guidelines for programming the newer 1200 and 1500 logic controllers are explained. The introduction of GRAPH programming language and how to create HTML pages on a logic controller which is configured as a server are in the same chapter. The theory part has also information about the Profinet communication standard and about industrial wireless local area networks. There is a chapter about the commissioning of the CU250S-2 control unit and a frequency converter because one of the example programs is related to the control unit.

The introduction of the example programs and the functions of those programs are subjects of the last chapter. As a product of this thesis eight programs were created with TIA Portal. In all of the programs the 1513-1 PN logic controller was used and Profinet is used for communication. The example programs are about communication between logic controllers, HTML pages, recipes, data logs, the configuration of a IWLAN network, synchronization of clocks, the deactivation of a Profinet IO device and positioning with the CU250S-2 control unit. As a result of the research and programming I have improved my professional skill and gained more knowledge for my future career.



**Key words**
GRAPH, HTML, programmable logic controller, IWLAN, Tia Portal, positioning, Profinet, frequency converter

**KÄSITTEIDEN MÄÄRITTELY**

| | |
|---|---|
| Array | Taulukko |
| AWP | Automation Web programming |
| Bitti | Binary digit; binäärinumero |
| CSMA/CD | Carrier Sense Multiple Access with Collission Detection; kanavanvarausmenetelmä, joka sisältää törmäysten havaitsemisen. |
| CSMA/CS | Carrier Sense Multiple Access with Collission Avoidance; kanavanvarausmenetelmä, joka pyrkii estämään törmäyksiä. |
| DB | Data block |
| Enum | Enumerated type; lueteltu tyyppi. |
| FB | Function block |
| FBD | Function Block Diagram; logiikkakaavio. |
| FC | Function |
| GSDML | General Station Description Markup Language |
| GRAPH | Vuokaavio |
| HTML | Hypertext Markup Language; kieli, jolla nettisivut on koodattu. |
| I/O | Input/Output |
| IP | Internet Protocol |
| Int | Integer; kokonaisluku. |
| IWLAN | Industrial Wireless Local Area Network; teollisuuteen suunniteltu langaton lähiverkkotekniikka. |
| LAD | Ladder logic; tikapuukaavio. |
| MAC | Media Access Control |
| maskaaminen | Yksittäisen bitin tilan selvittäminen bittijonosta. |
| Muuttuja | Ohjelmoinnissa käytetty nimitys tietovarastolle. |
| NTP | Network Time Protocol; protokolla kellojen tahdistamiseen. |
| OB | Organization block |
| OSI-malli | Open Systems Interconnection Reference Model; malli, joka kuvaa tiedonsiirtoprotokollien yhdistelmää seitsemässä kerroksessa. |
| PLC | Programmable  Logic Controller; logiikka. |
| Sana | 16 bittiä |

| | |
|---|---|
| SCL | Structured Control Langauge; lausekielinen ohjelmointikieli. |
| SSID | Service Set Indentifier; verkkotunnus langattomassa lähiverkossa. |
| STL | Statement List; käskylista. |
| Struct | Struktuuri |
| Tavu | 8 bittiä |
| TCP | Transmission Control Protocol |
| Telegrammi | Siemens käyttää nimitystä telegram taajuusmuuttajan ohjaamiseen käytetyistä ohjaussanoista. |
| UDP | User Datagram Protocol |
| WDS | Wireless Distribution System; järjestelmä, joka mahdollistaa tukiasemien yhdistämisen langattomasti. |

**LIITTEET**



**KUVIOT**

# 1 JOHDANTO

Tämän opinnäytetyön aiheena oli selvittää, miten TIA Portal -ohjelmistoa voidaan paremmin hyödyntää Apex Automation Oy:n projektoinnissa. Tavoitteena oli tutkia asioita, joita voidaan hyödyntää tulevaisuudessa ja luoda esimerkkiohjelmia ja ohjeita Apexin käyttöön. Lisäksi uusien 1500 -logiikoiden ominaisuuksien selvittäminen ja niiden hyödyntäminen oli yksi aiheista. TIA Portalin ja logiikoiden lisäksi aiheisiin kuuluivat Profinet, IWLAN ja CU250S-2 PN -ohjausyksikön käyttö paikoitukseen.

Opinnäytetyön alussa kerrotaan 1500 -logiikan muistialueista ja miten se suorittaa käyttäjän luoman ohjelman. Ohjelmakierto ja kiertoaika käsitellään kyseisessä luvussa. Ohjelman luomiseen, paneelisovelluksien tekemiseen ja taajuusmuuttajien määrittelyyn käytettävä TIA Portal -ohjelmisto ja sen eri osat esitellään logiikan toiminnan jälkeen. TIA Portal luvussa selvitetään Siemensin omia suosituksia ohjelmien luomiseen 1200- ja 1500 -logiikoille. GRAPH-ohjelmointikielestä ja sen toiminnasta kerrotaan myös TIA Portal luvussa. GRAPH-kieleen tutustuminen ja siitä ohjeiden luominen oli yksi opinnäytetyön tavoitteista. 1500 -logiikan verkkopalvelin ominaisuuden hyödyntämiseksi haluttiin myös tutkia, miten omia HTML-sivuja voidaan luoda logiikalle. Kaikissa esimerkkiohjelmissa, jotka tein Apexille, käytettiin kommunikointitapana Profinetiä. Profinetin toiminnan selvittämiseen pääpiirteittäin perehdytään Profinet luvussa. Samassa luvussa on perustietoa IWLAN-teknologiasta. CU250S-2-ohjausyksikköä käsittelevässä luvussa on selvitetty, miten ohjausyksikkö otetaan käyttöön ja minkälaisia paikoitustapoja se käyttää.

Kaikki esimerkkiohjelmat käsitellään viimeisessä luvussa. Ohjelmat syntyivät yleensä siitä, että haluttiin selvittää miten, jos ollenkaan, jokin asia voidaan tehdä. Kaikki ohjelmat on luotu käyttäen TIA Portalia. Pyrin tekemään ohjelmat käyttäen neuvoja, jotka opin selvitystyön tuloksena.

## 2 S7-1500 -OHJELMOITAVAT LOGIIKAT

Ohjelmoitavat logiikat ovat teollisuusympäristöön erikoistuneita tietokoneita. Ohjelmoitava logiikka saa tietoa kytkimiltä ja antureilta sen sisääntulojen kautta ja käsittelee saatua tietoa ohjelman perusteella, jonka jälkeen ohjelma muuttaa logiikan ulostulojen tiloja. Ulostulot ohjaavat konetta tai prosessia. Alun perin ohjelmoitavia logiikoita käytettiin korvaamaan perinteisiä releillä toteutettuja logiikoita. Logiikoiden ominaisuuksien lisääntyessä niitä ryhdyttiin käyttämään myös monimutkaisemmissa sovelluksissa. (PLCTutor 2013.)

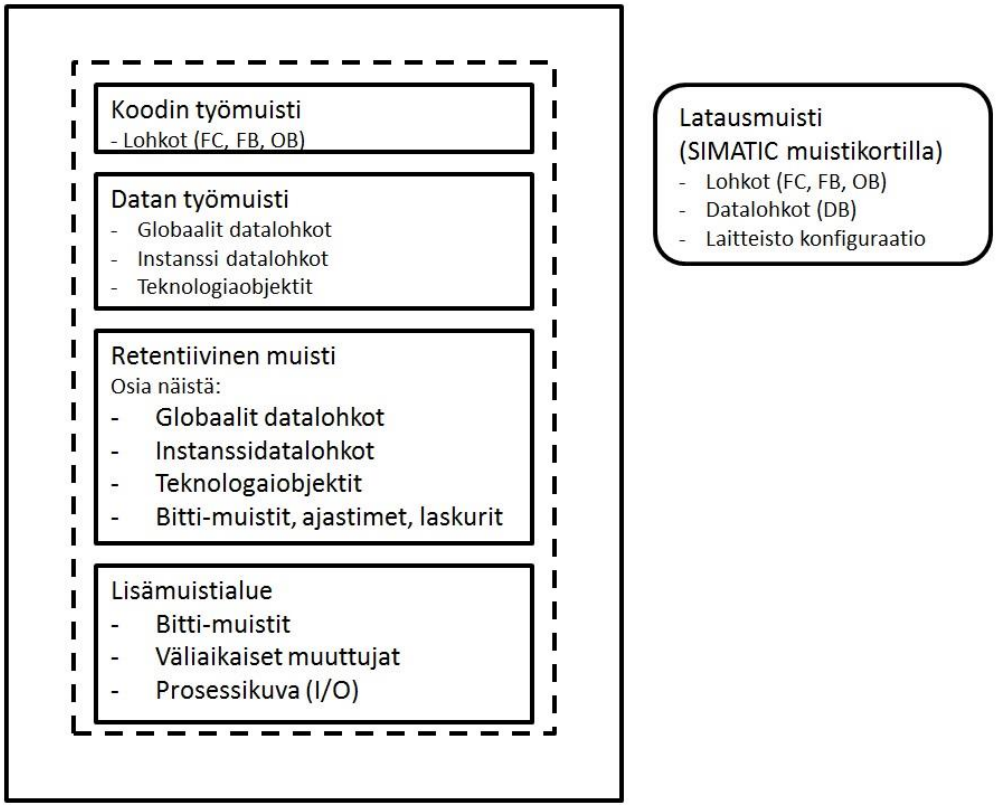
NEMA (National Electrical Manufacturers Association) määrittelee ohjelmoitavan logiikan digitaaliseksi sähköisesti toimivaksi laitteeksi, joka käyttää ohjelmoitavaa muistia sisäisenä tallennustilana käskyille, jotka toteuttavat tiettyjä toimintoja. Toimintoja ovat ajastintoiminnot, laskuritoiminnot, loogiset vertailut ja aritmeettiset laskutoiminnot. Toiminnoilla ohjataan erityyppisiä prosesseja ja koneita. (PLCTutor 2013.)

Siemensin valmistama S7-1500 -logiikkaohjain on suunniteltu korvaamaan vanhemmat S7-300- ja S7-400 -logiikkaohjaimet. Tuoteperheen logiikat ovat suunniteltu pienten koneiden ohjaukseen ja myös suurempien kokonaisuuksien ohjaamiseen. Aiemmin tarvittiin lisäkomponentteja tai ohjelmistoja toteuttamaan tiettyjä asioita. S7-1500 -logiikat sisältävät liikkeenohjaustoimintoja ja PID-säätimiä, jotka mahdollistavat erilaisten asioiden toteuttamisen ilman lisäohjelmistoja. S7-1500 -logiikoiden ohjelmointiin käytetään Step 7 Professional -ohjelmistoa, joka on integroituna TIA Portal -ohjelmointityökaluun. S7-1500 -logiikkaohjainten käyttökohteita ovat mm. isojen koneiden päälogiikkana toimiminen, kuljetinjärjestelmien ohjaaminen ja yksinkertaisten paikoitussovellusten toteuttaminen. (S7-1500 2014.)

S7-1500 -logiikoissa on näyttöpaneeli ja ohjausnappeja paneelissa navigointia varten. Paneelin kautta käyttäjä saa tietoa logiikan toiminnasta ja automaatiojärjestelmän tilasta. Paneelista voidaan lukea järjestelmän diagnostiikkatietoja, hälytyksiä sekä järjestelmään kytkettyjen laitteiden verkko-ominaisuuksia. Paneelin kautta pystyy muuttamaan logiikan IP-osoitetta, kellonaikaa, aikavyöhykettä ja käyttömoodia. Logiikan muistikortin pystyy myös tyhjentämään paneelin kautta ja logiikan palauttamaan tehdasasetuksille. (System manual 2014, 156–159.)

## 2.1 S7-1500 -logiikan muistialueet

Logiikan muisti on jaettu kolmeen päätyyppiin: latausmuistiin, työmuistiin ja retentiiviseen muistiin (KUVIO 1). Latausmuisti on ei-haihtuvaa muistia koodilohkoille datalohkoille, teknologiaobjekteille ja laitteisto määrittelyille. Ladattaessa ohjelmaa logiikalle nämä asiat latautuvat ensin latausmuistiin. Latausmuisti sijaitsee muistikortilla, joka asetetaan logiikkaan. Logiikka vaatii toimiakseen, että muistikortti on asetettu logiikkaan. Työmuisti on haihtuvaa muistia, joka sisältää koodi- ja datalohkot. Työmuisti on integroitu logiikkaan eikä sitä voi laajentaa. Työmuisti on jaettu kahteen alueeseen koodin työmuistiin ja datan työmuistiin. Koodin työmuisti sisältää ohjelmakoodia. Datan työmuisti sisältää datalohkoja ja teknologiaobjekteja. Logiikan tilan vaihtuessa power on -moodista käynnistys-moodiin ja stop-moodista käynnistys-moodiin globaalien datalohkojen muuttujat, instanssidatalohkojen muuttujat ja teknologiaobjektit alustetaan niiden aloitusarvoilla. Retentiivinen muisti on ei-haihtuvaa muistia, johon voi tallentaa rajatun määrän dataa. Sähkökatkoksen sattuessa muuttujat, jotka on määritelty retentiiviseksi, säilyttävät arvonsa. Normaalit muuttujat menettävät arvonsa ja niihin ladataan aloitusarvot logiikan tilan vaihtuessa edellä mainitulla tavalla. (Function manual 2013, 10.)



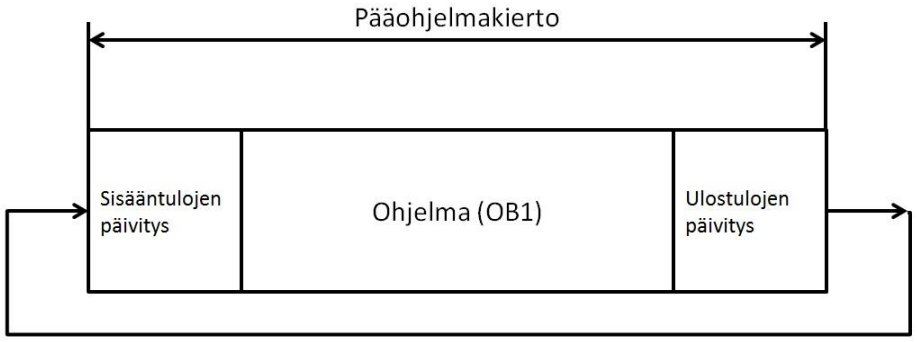

KUVIO 1. 1500 -logiikan muistialueet (Mukaillen Function Manual 2013, 9.)

Edellä mainittujen muistialueiden lisäksi logiikassa on lisämuistialueita. Lisämuistialueet sisältävät ajastimet, laskurit, väliaikaiset muuttujat ja prosessikuvan. (Function manual 2013, 12.)

## 2.2 Ohjelman prosessointi

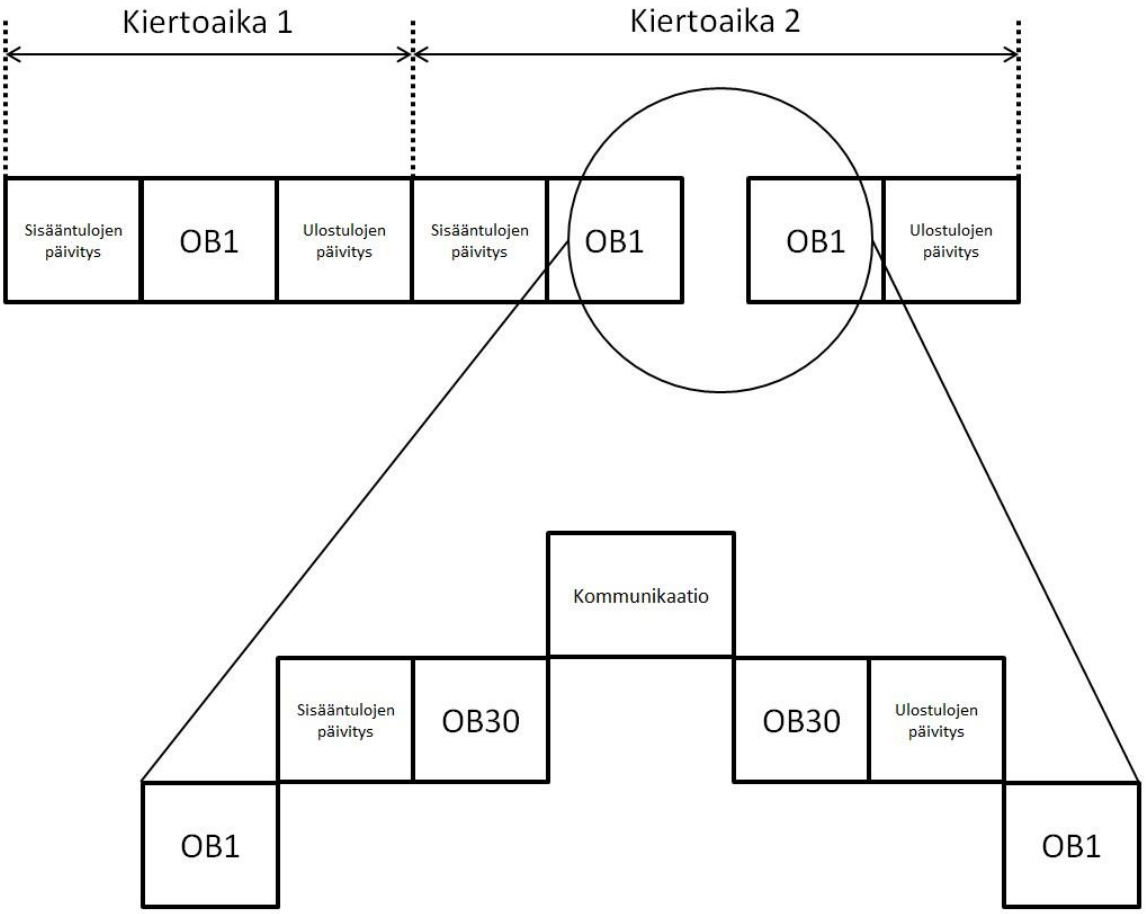
Logiikan suorittamat tehtävät on kirjoitettu ohjelmaan. Ohjelma voidaan suorittaa erilaisissa organisaatioyksiköissä, joilla on eri prioriteetit. Kaikkein yleisin suoritettava organisaatioyksikkö on syklisesti suoritettava OB1 -organisaatioyksikkö. OB1:en suorittamisen jälkeen logiikka aloittaa ohjelman suorittamisen taas alusta. Tätä kutsutaan logiikan pääohjelmakierroksi. Yksinkertaisimmissa sovelluksissa koko ohjelma on yhdessä organisaatioyksikössä. Kaikilla organisaatioyksiköillä on prioriteetti. OB1:llä on alhaisin prioriteetti, joka vastaa numeroa yksi. Korkein prioriteetti, jonka organisaatioyksikölle voi asettaa, on 26. Organisaatioyksiköt, joilla on korkeampi prioriteetti, keskeyttävät alemman prioriteetin omaavat organisaatioyksiköt. (Cycle and response times 2013, 9–10.)

Yhden logiikan suorittaman ohjelmakierron aikana ensin päivitetään sisääntulojen tilat, jotta ne ovat tiedossa ennen kuin OB1 suoritetaan. Käyttäjän luoma ohjelma, joka on OB1:en sisällä, suoritetaan ylhäältä alaspäin aikasiivuina. OB1:en suorittamisen jälkeen logiikka lukee ulostulojen tilat ja päivittää ne (KUVIO 2). Logiikka aloittaa kiertoajan tarkkailun sisääntulojen lukemisen aloittamisesta ja lopettaa sen, kun ulostulot on päivitetty. Kiertoajalla tarkoitetaan aikaa, jonka logiikka vaatii yhden pääohjelmakierron suorittamiseen. (Cycle and response times 2013, 11–12.)




KUVIO 2. Pääohjelmakierto (Mukaillen Cycle and response times 2013, 11.)

Kiertoaika ei ole sama jokaisella syklillä, vaan se vaihtelee. Kiertoaikaan vaikuttaa eri ohjelmalohkojen ajoajat ja muiden organisaatioyksiköiden tekemät keskeytykset. Alapuolelle olevassa kuvassa OB1 ensin suoritetaan normaalisti ja seuraavalla ohjelmakierrolla OB30 keskeyttää OB1:en. OB30:en keskeyttää kommunikaatio, koska sillä on suurempi prioriteetti. Tämä aiheuttaa pidemmän kiertoajan (KUVIO 3). (Cycle and response times 2013, 13.)




KUVIO 3. Kiertoajan pidentyminen (Mukaillen Cycle and response times 2013, 13.)

TIA Portal -ohjelmassa on mahdollista tarkkailla kiertoajan statistiikkaa ja pienintä ja suurinta kiertoaikaa. Näitä tietoja pystytään hyödyntämään pääohjelmakierron vasteaikaa määriteltäessä. (Cycle and response times 2013, 15.)

# 3 TIA PORTAL

S7-1200- ja S7-1500 -logiikoilla on erilainen järjestelmäarkkitehtuuri kuin vanhemmilla logiikoilla. Uusien logiikoiden myötä on myös tullut muutoksia ohjelmointitapoihin ja Step 7 -ohjelmistossa on ominaisuuksia, joita ei aiemmissa ohjelmistoissa ollut. (Programming Guideline 2014, 5.)

TIA Portal on Siemensin kehittämä ohjelmointityökalu. Tähän työkaluun on yhdistetty logiikkojen, käyttöliittymien ja turvaratkaisujen sekä taajuusmuuttajien ohjelmointi. TIA Portalissa on yhdistettynä Simatic Step 7-, Simatic WinCC- ja Sinamics StartDrive -ohjelmistot. (TIA Portal 2014.)

Simatic Step 7 -ohjelmistoa käytetään logiikkaohjainten ohjelmointiin. Ohjelmistoa voidaan käyttää yksittäisiin koneohjauksiin ja laajoja turvatoimintoja sisältäviin kokonaisuuksiin. Ohjelmistoa on saatavilla kahta eri tyyppiä sekä yksi lisäpaketti liikeohjaussovellusten ohjelmointiin ja lisenssi turvahyväksyttyjen ohjausten toteuttamiseen on myös saatavilla. Eri tyypit ovat nimeltään Basic ja Professional. Basic -ohjelmisto on tarkoitettu S7-1200 -logiikoiden ohjelmointiin. Professional on tarkoitettu S7-1200 -logiikoiden lisäksi S7-300-, S7-400-. S7-1500- ja WinAC -logiikoiden ohjelmointiin. Basic ja Professional -ohjelmistot sisältävät myös Simatic WinCC Basic -ohjelman, jota käytetään Simatic Basic -paneelien ohjelmointiin. Laite- ja järjestelmämäärittelyn tekemiseen käytetään graafista verkkonäkymää. Step 7 tukee IEC-standardin mukaisia ohjelmointikieliä. Tuetut ohjelmointikielet ovat LAD, FBD, GRAPH ja SCL. (SIMATIC STEP 7 2014.)

Simatic WinCC -ohjelmisto on suunnitteluohjelmisto SIMATIC HMI -laitteille. Ohjelmisto on integroitu TIA Portal -ohjelmaan ja mahdollistaa visualisointisovellusten luomisen. Integraatio mahdollistaa tiedonhallinnan esimerkiksi STEP 7 -ohjelmiston kanssa. WinCC -ohjelmistoa on saatavilla neljää eri mallia. Ohjelmistot ovat Basic, Comfort, Advanced ja Professional. Basic on tarkoitettu vain Basic -paneelien ohjelmointiin. Comfort on tarkoitettu kaikkien Simatic-paneelien ohjelmointiin. Advanced ja Professional ohjelmistot sisältävät Comfort-toiminnallisuuden. Advanced -ohjelmistolla ohjelmoidaan PC-pohjaisia yksittäispäätteitä ja Professional -ohjelmisto sisältää kaikki edellä olevat ohjelmistot sekä prosessivisualisoinnin toteutuksen. (WinCC V11 2014.)

Sinamics StartDrive -ohjelmisto on Sinamics-taajuusmuuttajien käyttöönottoon, parametrointiin ja diagnostiikkaan tarkoitettu ohjelmisto. Parametrointi tehdään wizard-sovelluksella. StartDrive tukee G120, G120C, G120D ja G120P taajuusmuuttajia. (SINAMICS StartDrive 2014.)

## 3.1 Suositellut ohjelmointitavat S7-1200- ja S7-1500 -logiikoille

Yleisesti ottaen Simatic ohjelmoitavien logiikoiden ohjelmointi on pysynyt samana vanhoista logiikoista uusiin logiikkoihin asti. Käytetyt ohjelmointikielet ovat LAD, FBD, STL, SCL ja GRAPH. Ohjelman rakenteen luomiseen käytetään organisaatioyksiköitä, toimintayksiköitä, toimintoja ja datalohkoja. Yksi suuri ero on kuitenkin symbolisten osoitteiden käyttäminen ja muuttujien nimityksenä käytetään sanaa tag. (Programming Guideline 2014, 6.)

TIA Portalia ja uusia logiikoita käytettäessä ohjelma käännetään suoraan konekieleksi alkuperäisestä ohjelmointikielestä riippumatta. Tällä saavutetaan tehokkaampi suorituskyky. S7-300 ja S7-400 logiikoille ohjelmointikielet ensin käännetään STL-kielelle ja sen jälkeen konekielelle. Suorituskyky heikkenee koska ohjelma pitää kääntää kaksi kertaa. (Programming Guideline 2014, 8–9.)

### 3.1.1 Ohjelmalohkot

TIA Portalin Step 7 -ohjelmistossa käytettävät ohjelmalohkot ovat OB:t eli organisaatioyksiköt, FB:t eli toimintayksiköt, FC:t eli toiminnat ja DB:t eli datalohkot. Nämä lohkot ovat samoja mitä Step 7 -ohjelman aiemmissa versioissa on käytetty. Eri lohkotyypeillä voidaan luoda selkeä rakenne ohjelmaan. Hyvän ohjelmarakenteen myötä monia ohjelmalohkoja voidaan käyttää uudelleen monissa projekteissa. Ohjelman jakaminen osiin eri lohkoilla ja näiden lohkojen uudelleen jakaminen osiin auttaa rakentamaan uudelleenkäytettäviä ohjelmalohkoja. (Programming Guideline 2014, 27.)

Organisaatioyksiköt (OB) ovat käyttöliittymä käyttöjärjestelmän ja käyttäjän luoman ohjelman välillä. Käyttöjärjestelmä kutsuu organisaatioyksiköitä ja ne ohjaavat logiikan

käynnistystoimenpiteitä, syklistä ohjelman suoritusta, keskeytyksiä ja virheidenhallintaa. Ohjelmaan voidaan luoda monia organisaatioyksiköitä ja ne suoritetaan OB:n numeron perusteella. Tiettyjä organisaatioyksiköitä kutsutaan vain tietyissä tapauksissa tai vain tietyin väliajoin. (Programming Guideline 2014, 28–29.)

Toiminnat (FC) ovat lohkoja, jotka eivät sisällä syklistä tiedon talteenottoa. Lohkoille pitää siis syöttää tietoa, joka kerta kun niitä kutsutaan ohjelmassa. Väliaikaiset muuttujat ja ulostulo muuttujat menettävät tietonsa lohkon prosessoinnin jälkeen. Lohkoa kutsuttaessa optimoiduissa datalohkoissa muuttujiin ladataan arvo nolla. FC-lohkojen tiedot voidaan tallentaa globaaleihin datalohkoihin, jos ne halutaan pitää muistissa. FC-lohkoja käytetään monesti toistuvissa sovelluksissa. Esimerkiksi lieriön tilavuuden laskeminen voidaan tehdä ohjelmoimalla FC-lohko SCL-kielellä. Lohkon sisälle kirjoitetaan laskukaava ja käsketään lohkoa palauttamaan laskettu arvo. Lohkoa kutsuttaessa lohkolle kerrotaan lieriön korkeus ja pohjan säde. FC-lohkoa kutsuttaessa SCL-kielellä kirjoitetussa koodissa voidaan tilavuus kirjoittaa suoraan muuttujaan koska lohkon palauttama arvo on tilavuus. (Programming Guideline 2014, 30–31.)

FB:t ovat lohkoja, jotka sisältävät syklisen tiedontallentamisen. Arvot tallentuvat pysyvästi instanssidatalohkoon. Lohkon väliaikaiset muuttujat menettävät arvonsa lohkon kutsun jälkeen. FB-lohkoja käytetään luomaan aliohjelmia ja jäsentämään ohjelmaa. FB-lohkoja voidaan kutsua monta kertaa eri osissa ohjelmaa, ja tämä helpottaa jatkuvasti toistuvien ohjelmaosien ohjelmointia. (Programming Guideline 2014, 32.)

FB:n kutsua sanotaan instanssiksi. Tieto, jota instanssi käyttää, tallennetaan instanssidatalohkoon. Instanssidatalohko sisältää pysyvää muistia, johon tallennetaan input, output, InOut ja static muuttujien arvot. FB:t sisältävät myös haihtuvaa muistia. Haihtuvaan muistiin tallennetaan väliaikaiset muuttujat. Haihtuvassa muistissa olevat muuttujat pitävät arvonsa yhden ohjelmakierron ajan. Väliaikaiset muuttujat täytyy alustaa ohjelmakierroittain. (Programming Guideline 2014, 33.)

Multi-instanssiksi kutsutaan tilannetta, jossa FB:tä kutsutaan toisen FB:n sisällä. Kutsuttu FB tallentaa tietonsa ylemmäntason FB:n instanssidatalohkoon. Käytettäessä ajastimia tai laskureita FB:n sisällä on suositeltavaa käyttää multi-instanssia, jotta lohkojen määrä pysyy pienenä. On suositeltavaa myös ohjelmoida FB:t siten, että instanssidatalohkon sisältöä

pystyy muuttamaan vain sille määrätty FB. Tällä tavoin voidaan luoda lohkoja, joita voidaan käyttää myös muissakin projekteissa. (Programming Guideline 2014, 33–34.)

Instanssidatalohkon vastakohta on globaali datalohko. Globaaleihin datalohkoihin tallennetut tiedot ovat saatavissa kaikkialla ohjelman sisällä. Globaali datalohko voi sisältää eri tietotyyppiä olevia muuttujia. Monissa osissa ohjelmaa käytetyt muuttujat tulisi tallentaa globaaleihin datalohkoihin. (Programming Guideline 2014, 35–36.)

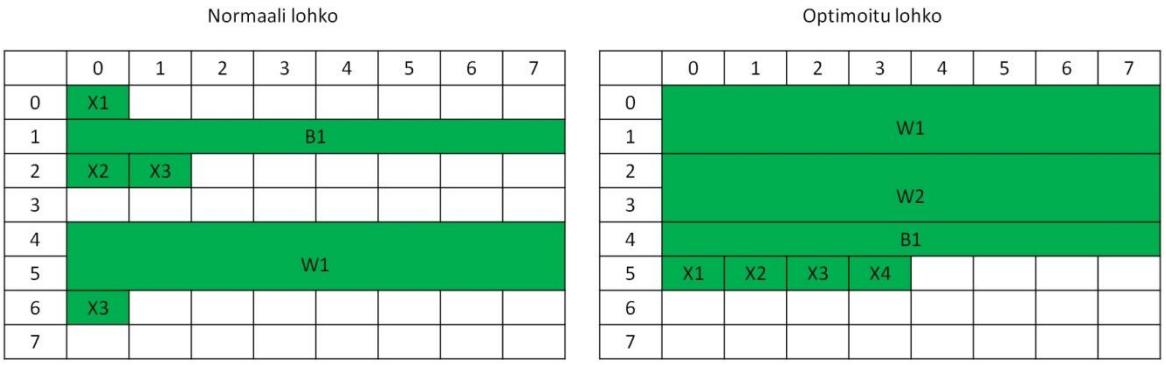
Lohkojen uudelleenkäytettävyys on tärkeää, jos haluaa luoda lohkoja, joita voidaan käyttää eri osissa ohjelmaa tai eri projekteissa. Jos jokaiselle lohkolle määritellään itsenäinen tehtävä, saavutetaan automaattisesti selvä ja hyvin rakennettu ohjelma. Seuraavat asiat tulisi ottaa huomioon, jos haluaa luoda uudelleenkäytettäviä lohkoja:

- Käsittele lohkoja yksittäisinä tehtävinä, jotka ovat osa koko ohjelmaa. Yksi lohko siis edustaa yhtä suoritettavaa tehtävää.
- Käytä useita syklisesti suoritettavia main-organisaatioyksiköitä jakamaan automaatiolaitteisto osiin.
- Tiedon välittäminen pitäisi suorittaa lohkojen in- ja out-liitäntöjen kautta.
- Älä käytä projektikohtaista tietoa lohkon sisällä. (Programming Guideline 2014, 40.)

### 3.1.2 Optimoidut datalohkot

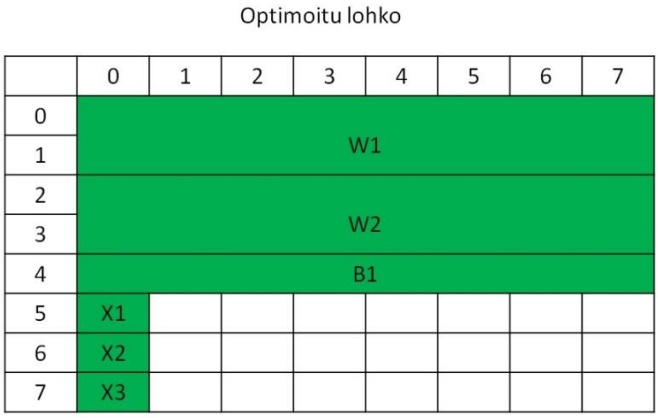
Optimoiduissa datalohkoissa tietotyypit lajitellaan automaattisesti. Lajittelulla pidetään huoli, että tietotyyppien välillä ei ole rakoja. Tieto tallennetaan niin, että logiikan prosessori pääsee siihen käsiksi nopeasti. Optimoidut datalohkot ovat mahdollisia vain S7-1200- ja S7-1500 -logiikoita käytettäessä. Optimoiduissa datalohkoissa tietoihin pääsee käsiksi vain symbolisesti. (Programming Guideline 2014, 10–13.)

S7-1200- ja S7-1500 -logiikoiden optimoidut datalohkot lajittelevat tietotyypit hieman eri tavoin. S7-1200 -logiikan optimoitu datalohko lajittelee isot tietotyypit lohkon alkuun ja pienet lohkon loppuun (KUVIO 4). (Programming Guideline 2014, 10–13.)




KUVIO 4. 1200 -logiikan optimoitu datalohko (Mukaillen Programming Guideline 2014, 10.)

S7-1500 -logiikka lajittelee myös isot tietotyypit alkuun ja pienet datalohkon loppuun. Yksittäisille biteille ja boolean-tietotyypeille varataan tavu tietoa, jotta logiikan ei tarvitse maskata bittejä (KUVIO 5). Tällä tavoin saavutetaan parempi suorituskyky, koska prosessorilla on suora pääsy tietoihin. S7-1500 -logiikan yhden datalohkon maksimikoko voi olla jopa 16 Mt. Optimoidut datalohkot sisältävät mahdollisuuden ladata tietoa käynnissä olevaan prosessiin ilman, että datalohko pitää alustaa uudelleen. Tätä varten täytyy datalohkosta varata tilaa. (Programming Guideline 2014, 10–13.)




KUVIO 5. 1500 -logiikan optimoitu datalohko (Mukaillen Programming Guideline 2014, 11.)

Optimoituja datalohkoja olisi hyvä aina käyttää, koska ne lyhentävät prosessointiaikaa. Tiedon kopiointia optimoitujen ja normaalien datalohkojen välillä on vältettävä, koska se vaatii suurta prosessointiaikaa. (Programming Guideline 2014, 10–13.)

### 3.1.3 Symboliset osoitteet ja PLC-tietotyypit

Symbolisia osoitteita käytettäessä ohjelmoijan ei tarvitse kiinnittää huomiota logiikan sisäiseen muistiin. Nykyaikaiset logiikat osaavat määrätä, missä paras paikka tallennettavalle tiedolle on. Symboliset osoitteet tekevät ohjelmasta helppolukuisempaa ja helpommin ymmärrettävää. Eri muuttujille on helppo antaa kuvaavat nimet, jolloin ohjelmaa tarvitsee kommentoida vähemmän. Esimerkiksi array-tietotyypin, eli taulukon, yksittäisiin tietoihin voidaan viitata epäsuorasti symbolisella nimellä. Luomalla int-tietotyyppiä olevan muuttujan ja antamalla tälle symbolisen nimen voidaan taulukon indeksin tilalle kirjoittaa tämä muuttuja ja näin saadaan taulukon tiedolle symbolinen nimi. (Programming Guideline 2014, 47–49.)

Struct-tietotyyppi edustaa tietorakennetta, joka sisältää eri tietotyyppiä olevia muuttujia. Struktuurin esitteleminen suoritetaan lohkossa, jossa sitä käytetään. PLC-tietotyypit eroavat struktuureista siinä, että niitä voidaan käyttää kaikkialla ohjelmassa. PLC-tietotyyppi voi sisältää myös eri tietotyyppejä olevia muuttujia ja myös struktuureja. PLC-tietotyypin muuttaminen muuttaa myös kaikkia muita kohteita, joissa tätä PLC-tietotyyppiä on käytetty. PLC-tietotyyppejä käytetään yhdistelemään tietoja, jotka ovat yhteyksissä toisiinsa. Esimerkiksi moottorin ohjaamiseen voi luoda oman PLC-tietotyypin, joka sisältää nopeuden asetusarvon ja pyörimissuunnan. Tätä tietotyyppiä voidaan sen jälkeen käyttää kaikissa moottorisovelluksissa. PLC-tietotyypeille antamalla symboliset nimet ohjelmasta tulee selkeämpi. PLC-tietotyyppejä tulisi käyttää struktuurien sijaan, jos samanlaista rakennetta käytetään monessa paikassa ohjelmaa. (Programming Guideline 2014, 51)

### 3.1.4 Kirjastot

TIA Portalilla voidaan luoda itsenäisiä kirjastoja, joihin voidaan tallentaa uudelleenkäytettäviä osia projekteista. Kirjastoon voidaan tallentaa esimerkiksi logiikoiden, paneelien ja taajuusmuuttajien määrittelyt. Lohkot, jotka ovat uudelleenkäytettäviä, kannattaa myös tallentaa kirjastoihin. (Programming Guideline 2014, 54.)

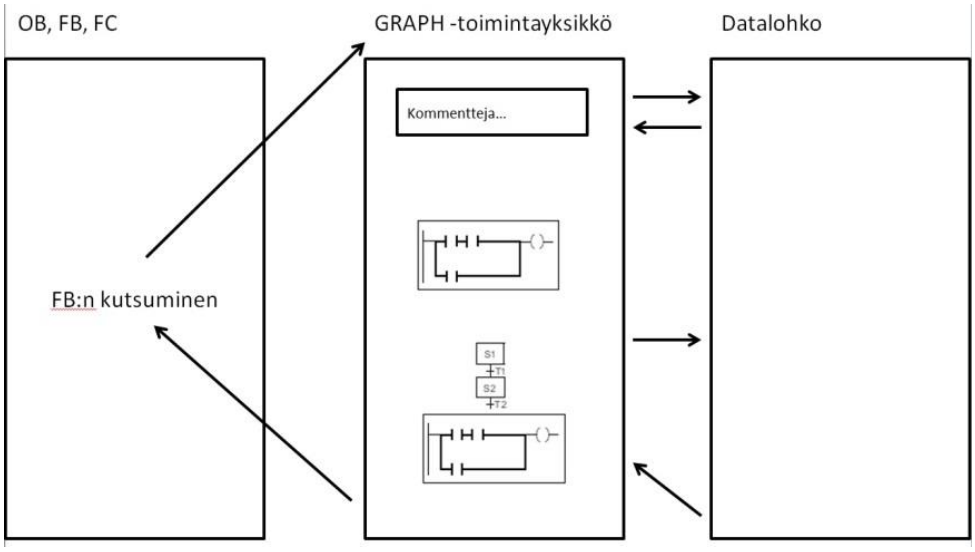
TIA Portalissa voi luoda kahdenlaisia kirjastoja, jotka ovat nimeltään projektikirjasto ja globaali kirjasto. Projektikirjastot ovat integroitu projekteihin ja mahdollistavat uudelleen-

käytettävyyden projektien sisällä. Globaaleihin kirjastoihin tallennetut asiat ovat käytettävissä monissa projekteissa. Molemmat kirjastotyypit sisältävät kaksi erilaista tallennustyyppiä. Tallennustyypit näkyvät TIA Portalissa eri kansioina. Ensimmäinen tyyppi on nimeltään master copies. Master copies -kansioon tallennetut tiedot eivät ole yhteyksissä projektiin ja sisältävät vain kopiot projektissa käytetyistä lohkoista ja määrittelystä. Toinen tyyppi on nimeltään types. Types -kansioon tallennetut asiat ovat yhteyksissä projektiin. (Programming Guideline 2014, 54–55.)

Types-konsepti mahdollistaa standardoitujen automaatiotehtävien luomisen. Types -kansioon tallennettuja lohkoja pystytään päivittämään ja versioimaan. Lohkojen muutokset päivittyvät kaikissa lohkoissa. Master copies -kansioon tallennettuja kopioita ei voi muuttaa yhdestä paikkaa ja päivittää sen jälkeen kaikkia käyttökohteita. (Programming Guideline 2014, 56.)

### 3.2 GRAPH-ohjelmointikieli

GRAPH on graafinen ohjelmointikieli, jolla luodaan sekvenssejä. Prosessi hajotetaan yksittäisiin askeleisiin, joista jokaisella on selkeä tehtävä. Suoritettavat toiminnot on määritelty yksittäisten askeleiden sisälle. Askeleiden välillä on siirtymiä. Siirtymät sisältävät ehtoja seuraavaan askeleeseen siirtymistä varten. Sekvenssi ohjaa prosessia ennalta määrätyssä järjestyksessä tiettyjä ehtoja noudattaen. Sekvenssiin kuuluu aina vähintään kolme eri lohkoa GRAPH -toimintayksikkö, instanssidatalohko ja ohjelmalohkon kutsuminen pääohjelmassa (KUVIO 6). (STEP 7 2013, 2486.)




KUVIO 6. Sekvenssin ohjelmakierto (Mukaillen STEP 7 2013, 2487.)

Jokaisella ohjelmakierrolla vakituiset käskyt, jotka ovat lohkon alussa ja lopussa, suoritetaan vaikka yhtään askelta ei olisi aktiivisena. Esikäskyjen suorittamisen jälkeen suoritetaan aktiivisena olevan askeleen toiminnot. Lopuksi suoritetaan lohkon lopussa olevat vakituiset jälki-käskyt. (STEP 7 2013, 2487.)

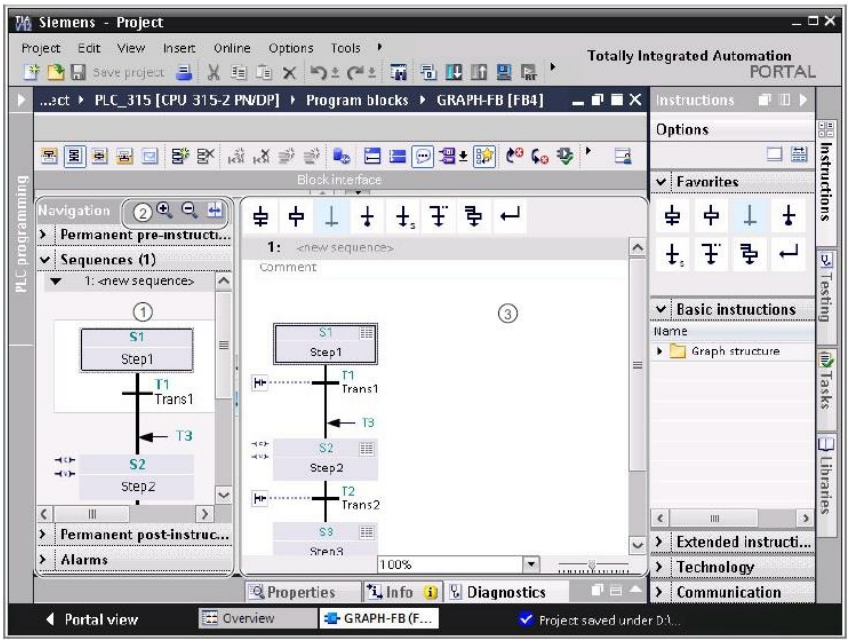
Ehtoja käytetään ohjaamaan askeleen yksittäisiä toimintoja, koko askeleen toimintaa, seuraavaan askeleeseen siirtymistä tai koko sekvenssiä. Ehtoja käytetään vakituisissa esi- ja jälki-käskyissä, interlock- ja supervision-hälytysten luomiseen ja siirtymissä. Ehtoja luodaan LAD- tai FBD-ohjelmointikielellä. (STEP 7 2013, 2508.)

Yksi askel sisältää interlock- ja superivision-hälytysten ehdot ja askeleen suorittamat tehtävät. Interlockia käytetään luomaan ehtoja, joista askeleen yksittäiset toiminnot riippuvat. Askeleen tehtävät, joihin interlock on liitetty, suoritetaan vasta sitten, kun interlockin ehdot on täytetty. Ohjelma ilmoittaa häiriöstä, jos interlockin ehdot eivät täyty. Tässä tapauksessa on mahdollista määritellä hälytys tätä tilannetta varten. Häiriö ei vaikuta seuraavan askeleen suorittamiseen. Supervisionia käytetään luomaan ehtoja, jotka vaikuttavat koko askeleen toimintaan. Askel pysyy päällä ja ohjelma ilmoittaa häiriöstä, jos ehdot eivät täyty. Sekvenssi ei siirry seuraavaan askeleeseen ennen kuin häiriö on korjattu ja hälytys on kuitattu. (STEP 7 2013, 2493.)

Sekvenssin suoritus alkaa kun aloitusaskel tulee aktiiviseksi. Useita aloitusaskeleita voidaan luoda, jos käytetään samanaikaisesti suoritettavia haaroja. Niin kauan kuin askel py-

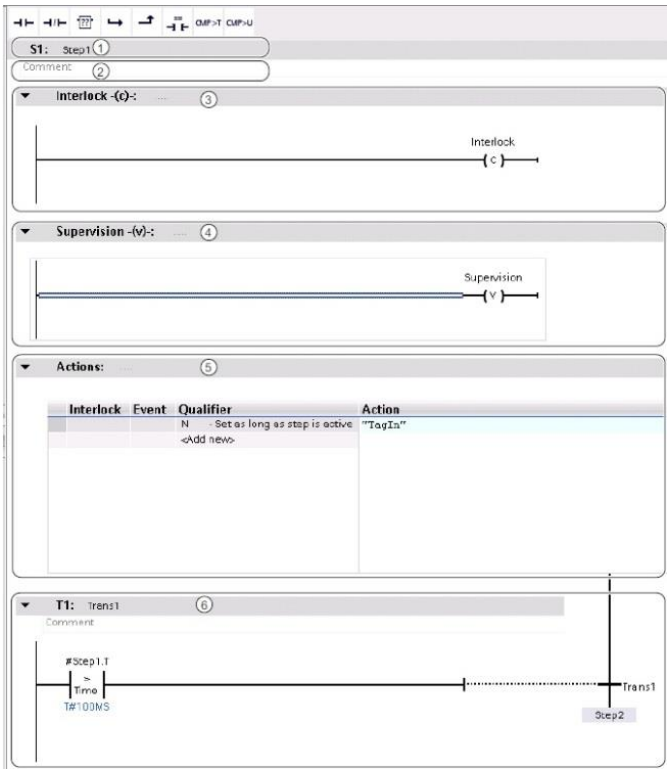
syy aktiivisena, sen toiminnot suoritetaan. Interlock ehdot otetaan huomioon tässä kohtaa. Kaikkien mahdollisten toimintojen suorittamisen jälkeen tehdään tarkistus, että supervision häiriötä ei ole. Seuraavaan askeleeseen siirrytään, jos häiriötä ei ole havaittu ja siirtymäehdot ovat täyttyneet. Askel pysyy aktiivisena, jos supervision ehdot eivät ole täyttyneet tai siirtymäehdot eivät ole täyttyneet. Sekvenssin lopussa voidaan käyttää hyppyjä tai lopettaa sekvenssin suorittaminen. Hyppy-käskyllä voidaan siirtyä tiettyyn askeleeseen tai toiseen sekvenssiin. (STEP 7 2013, 2488.)

Sekvenssi luodaan sille ominaisessa ikkunassa. Ikkunassa määritellään esi- ja jälki-käskyt, sekvenssin kulku ja interlock- ja supervision-hälytykset (KUVIO 7). (STEP 7 2013, 2528.)



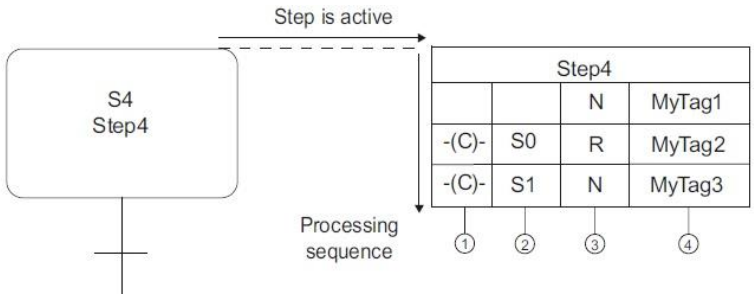

KUVIO 7. GRAPH -ohjelman ohjelmointi-ikkuna (STEP 7 2013, 2529.)

Yksittäisen askeleen näkymästä voidaan määritellä ehtoja interlock- tai supervision- hälytyksille. Näkymässä voidaan myös määritellä askeleen tehtävät ja ehdot seuraavaan askeleeseen siirtymistä varten (KUVIO 8). (STEP 7 2013, 2533.)




KUVIO 8. Yksittäisen askeleen näkymä (STEP 7 2013, 2534.)

Yksi askel voi suorittaa monta tehtävää kerralla ja ne pyritään suorittamaan niin kauan kuin askel on aktiivisena. Askeleen tehtävät suoritetaan ylhäältä alaspäin askeleen aktivoituessa tai käyttäjän määrittelemän tapahtuman sattuessa. Jokainen askeleen tehtävä voidaan yhdistää interlock ehtoon (KUVIO 9). Tietyt tehtävät pitää yhdistää johonkin tapahtumaan ja tiettyjä tehtäviä ei voi yhdistää tapahtumiin ollenkaan. (STEP 7 2013, 2497–2499.)




KUVIO 9. Askeleen suorittaminen (STEP 7 2013, 2497.)

Askeleen tekemät tehtävät voivat esimerkiksi olla ulos- ja sisääntulojen ohjaus, muiden sekvenssin askeleiden ohjaaminen tai lohkojen kutsuminen. Tehtävät suoritetaan, kun askel, jossa tehtävät suoritetaan, tulee aktiiviseksi. Tehtäviä, jotka on linkitetty tiettyihin tapahtumiin tai interlockiin, suoritetaan tapahtuman sattuessa tai kun interlock ehdot ovat

täyttyneet. Askeleessa voidaan käyttää myös ajastimia ja laskureita. Alapuolella olevassa kuviossa 10 ovat kaikki vakiokäskyt. Nämä käskyt voidaan kaikki yhdistää interlockiin. (STEP 7 2013, 2496.)

| Tehtävä | Tietotyyppi | Kuvaus |
|---|---|---|
| N – Aseta niin kauan kuin askel on aktiivinen | BOOL<br>FB, FC, SFB, SFC | Signaalitaso on yksi niin kauan kuin askel on aktiivinen. |
| S – Aseta 1 | BOOL | Askeleen aktivoituessa signaalitaso asetetaan ykköseksi. |
| R – Aseta 1 | BOOL | Askeleen aktivoituessa signaalitaso asetetaan nollaksi. |
| D – Vetohidastettu | BOOL ja<br>TIME/DWORD | Asetetun ajan kuluttua signaalitaso on yksi askeleen ollessa aktiivinen. |
| L – Aseta rajoitetuksi ajaksi | BOOL ja<br>TIME/DWORD | Signaalitaso on yksi rajoitetun ajan tai niin kauan kuin askel on päällä. |

KUVIO 10. Vakiokäskyt (Mukaillen STEP 7 2013, 2498 – 2499.)

Askeleen tekemät tehtävät voidaan linkittää tiettyihin tapahtumiin, jotta tehtävän suorittaminen riippuu tietyistä ehdoista. Tehtävä, joka on linkitetty tapahtumaan, suoritetaan tapahtuman nousevalla tai laskevalla reunalla. Tämä tarkoittaa sitä, että toiminnot suoritetaan vain sen ohjelmakierron aikana, jossa tapahtuma havaitaan. Kuviossa 11 on kaikki tapahtumat, jotka voidaan linkittää tehtäviin. (STEP 7 2013, 2534.)

| Tapahtuma | Signaalitaso | Kuvaus |
|---|---|---|
| S1 | Positiivinen reuna. | Askel aktivoitu. |
| S0 | Negatiivinen reuna. | Askel deaktivoitu. |
| V1 | Positiivinen reuna. | Supervision häiriö. |
| V0 | Negatiivinen reuna. | Supervision häiriö on poistunut |
| L0 | Positiivinen reuna. | Interlock häiriö on poistunut. |
| L1 | Negatiivinen reuna. | Interlock häiriö. |
| A1 | Positiivinen reuna. | Hälytys on kuitattu. |
| R1 | Positiivinen reuna. | Lohkon ulkopuolinen ohjaus. |

KUVIO 11. Linkitettävät tapahtumat (Mukaillen STEP 7 2013, 2534.)

Sekvenssiä kutsutaan ohjelmassa FB-lohkona, jossa on sisään- ja ulostulo parametreja sekvenssin ohjaamiseen ja tarkkailuun. FB-lohko sisältää aina standardi parametrit, jotka luo-

daan automaattisesti kun lohko lisätään ohjelmaan. Lohkoon on mahdollista lisätä lisää parametreja. Lisäparametrit määritellään TIA Portalin asetuksista kaikkiin GRAPH-kielellä luotuihin lohkoihin. Lisäparametrit sisältävät enemmän diagnostiikkaa sekvenssistä. Yksittäisiä parametreja voidaan myös lisätä ja poistaa. Liitteessä 1 on FB-lohkon sisään- ja ulostuloparametrit. (STEP 7 2013, 2510.)

### 3.3 1500 -logiikan verkkopalvelin ominaisuus

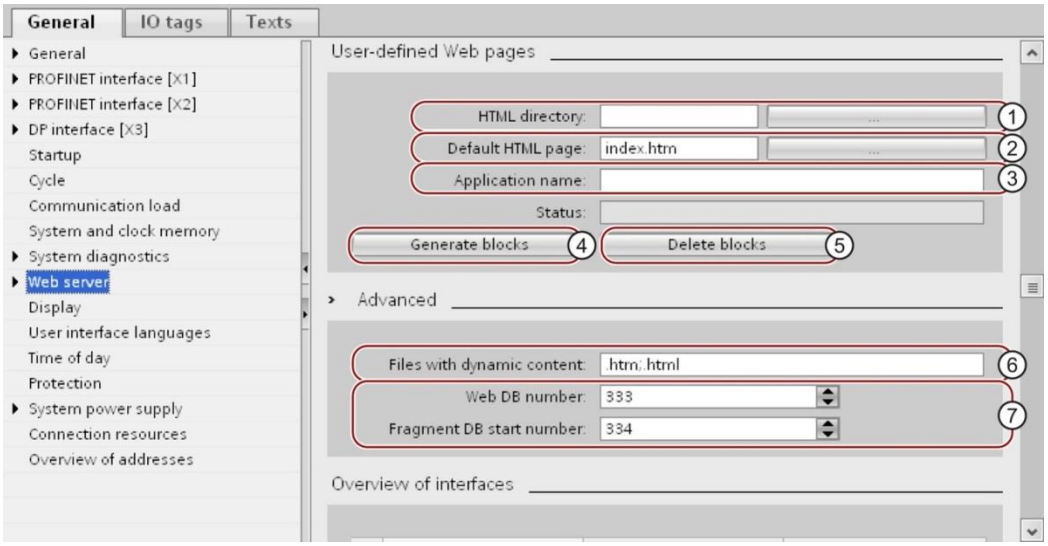
Verkkopalvelinominaisuus mahdollistaa logiikan valvonnan ja hallinnoinnin internetiä käyttäen. Monitorointia voi tehdä siis ilman Step 7 -ohjelmistoa. Logiikkaan liittymiseen tarvitaan vain nettiselain. Palvelin on pois päältä oletuksena ja se täytyy aktivoida lataamalla projekti logiikkaan, jossa palvelin on asetettu aktiiviseksi. (S7-1500 Web server 2013, 9.)

Verkkopalvelin aktivoidaan TIA Portalin projektinäkymässä. Aktivointi tehdään avaamalla käytetyn 1500 -logiikan ominaisuudet ja sieltä valitsemalla verkkopalvelinasetukset. Verkkopalvelinasetuksista voidaan asettaa verkkopalvelin aktiiviseksi, sallia vain HTTPS-protokollaa käyttävät yhteydet, aktivoida automaattinen päivitys, vaihtaa kieltä, lisätä käyttäjiä ja lisätä omia HTML-sivuja. (S7-1500 Web server 2013, 11–15.)

HTML-sivuja voi luoda itse siihen tarkoitetulla ohjelmalla ja niistä voidaan generoida datalohkoja Step 7 -ohjelmassa. Datalohkot ladataan logiikkaan ja WWW-lohkoa käyttäen ne voidaan synkronoida ohjelman kanssa. WWW:n kutsumisen jälkeen verkkopalvelimella näkyy linkki käyttäjän luomaan nettisivuun. Linkkiä klikkaamalla sivu aukeaa uuteen ikkunaan. (S7-1500 Web server 2013, 47.) WWW-lohkolle pitää kertoa vain datalohko, joka kuvaa nettisivua. Nettisivua kuvaava datalohko luodaan automaattisesti kun nettisivuista generoidaan datalohkoja. (S7-1500 Web server 2013, 59.)

Omien sivujen tuominen logiikkaan tehdään kuviossa 12 näkyvässä ikkunassa. Ensimmäisessä kohdassa määritellään polku, josta HTML-sivut ja kaikki siihen liittyvät tiedosto löytyvät. Aloitussivu määritellään seuraavassa kohdassa. Yleensä se on nimeltään index.html. Kolmannessa kohdassa määritellään nimi nettisivuille. Tämä nimi tulee olemaan linkin nimi, joka näkyy verkkopalvelimella. Tämän jälkeen datalohkot voidaan luoda generate

blocks -napilla. Luodut datalohkot voidaan poistaa delete blocks -napilla. Numerolla kuusi merkityssä kohdassa määritellään tiedostotyypit, jotka sisältävät AWP-käskyjä. Viimeisessä kohdassa voi määritellä datalohkojen numerot. (S7-1500 Web server 2013, 58–59.)




KUVIO 12. HTML-sivujen tuominen logiikkaan (S7-1500 Web server 2013, 58.)

AWP-käskyt ovat erikoiskäskyjä tiedon välittämiseen logiikan ja käyttäjän luoman HTML-sivun välillä. AWP-käskyt syötetään HTML-sivuun kommentteina ja niillä voidaan

- lukea muuttujista tietoa
- kirjoittaa muuttujiin tietoa
- määrittää enum-tyyppejä (lueteltu-tyyppi)
- määrätä enum-tyyppejä muuttujille. (S7-1500 Web server 2013, 49.)

Syntaksi muuttujien lukemiseen logiikalta on ”:=”Muuttujan nimi”:”. Muuttujan nimi täytyy olla sama, jota ohjelmassa käytetään. Datalohkon sisällä olevaan muuttujaan viitataan näin ”:=”Datalohkon nimi”.muuttujan nimi:”. Muuttujia voidaan lukea tällä tavalla missä vain HTML-sivun sisällä.

Muuttujien kirjoittamiseen täytyy käyttää eri syntaksia ja esimerkiksi tekstinsyöttölaatikkoa tai pudotusvalikkoa, jolla muuttujan arvo valitaan. Kirjoittaminen onnistuu vain, jos kirjautuneella käyttäjällä on oikeudet kirjoittaa tietoa. Syntaksi on seuraavanlainen ”<!—AWP_In_Variable Name=’”Datalohkon nimi”.muuttuja1’ -->”. (S7-1500 Web server 2013, 51–52.)

Enum-tyypeillä voidaan muuttaa numeerisia arvoja tekstiksi ja päinvastoin. Esimerkiksi voidaan määritellä, että arvo nolla vastaa tekstiä ”päällä”. Syntaksi on seuraavanlainen ”<!—AWP_Enum_Def Name=”Enum1” Values=’0:”päällä”, 1:”pois, 2:”vika’ ”. (S7-1500 Web server 2013, 54–55.)

# 4 PROFINET

Profinet on teollisuuteen suunniteltu kommunikointistandardi, joka perustuu Ethernet-teknologiaan. Toisin sanottuna Profinet on teollisuus-Ethernet-standardi. Ethernetin puutteita on korjattu protokollalisäyksillä, jotta se täyttäisi teollisuusympäristön vaatimukset. Profinetillä pystytään toteuttamaan tiedonsiirto reaaliaikaisesti toisin kuin normaalissa Ethernetissä. Profinetillä on mahdollista toteuttaa liikkeenohjaussovelluksia, jotka vaativat jopa alle yhden millisekunnin vasteaikoja. (Profinet 2014.)

Ethernet on tietoverkko, joka perustuu tietokehyksiin. Ethernet mahdollistaa tiedonsiirron kaikkien samaan verkkoon kytkettyjen laitteiden välillä käyttäen kehyksiä tai toisin sanottuna paketteja. Internet perustuu täysin tähän teknologiaan. OSI-mallin näkökulmasta Profinet määrittelee fyysisen kerroksen ja siirtoyhteyskerroksen. Ethernet on standardisoitu IEEE standardissa 802.3. (Pigan & Matter 2008, 16.)

Teollisuus-Ethernet syntyi vuonna 1985, kun Siemens AG esitteli Ethernetin teollisuuskäyttöön. Tätä Ethernetiä kutsuttiin nimellä ”SINEC H1”. Teollisuus-Ethernetin kehitys oli välttämätöntä, koska teollisuusympäristön vaatimukset ovat täysin erilaisia toimistoympäristön vaatimuksista. Teollisuus-Ethrnetin perus idea on olemassa olevien standardien käyttäminen teollisuuden kommunikoinnissa. Normaalista standardista erotaan vain silloin, kun se ei ota huomioon tuotanto- ja prosessiympäristöä. Tällä tavalla teollisuus-Ethernetin ja perinteisen Ethernetin kanssakäyminen onnistuu ongelmitta. (Pigan & Matter 2008, 17.)

Eri sovellusten toteuttamiseen Profinet tarjoaa kaksi mahdollisuutta Profinet IO -protokollalla hajautetun I/O:n integroiminen ja modulaaristen koneiden toteuttaminen Profinet CBA -protokollalla (Pigan & Matter 2008, 17). S7-1500 -logiikat eivät tue Profinet CBA -protokollaa, joten sitä ei käsitellä tässä (STEP 7 2013, 196). Profinetin kommunikointi voidaan jakaa kolmeen askeleeseen. Profinet CBA käyttää TCP/IP (Transmission Control Protocol/Internet Protocol) kommunikointia ja reaaliaikaista kommunikointia (RT). TCP/IP mahdollistaa 100 ms:n vasteajan ja reaaliaikainen 10 ms:n vasteajan. Profinet IO käyttää ainoastaan reaaliaikaista kommunikaatiota prosessidatan siirtoon. Isokroninen reaaliaikainen kommunikointi (IRT) mahdollistaa 1ms:n vasteajan. Sitä käytetään liikkeenohjauksessa. (Pigan & Matter 2008, 20.)

## 4.1 Ethernet

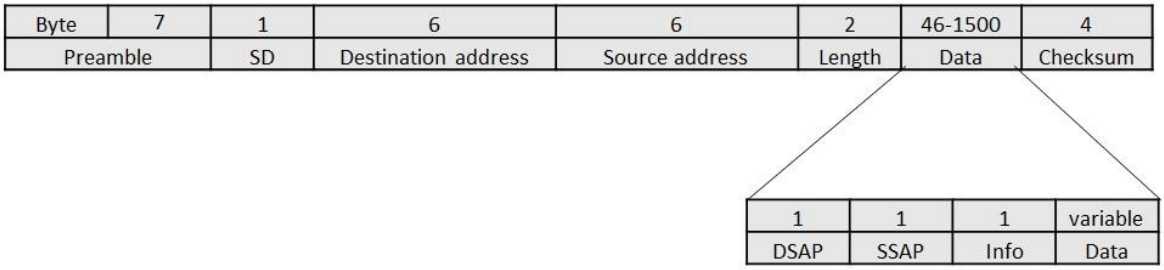
Profinet perustuu Ethernet-teknologiaan. Profinetin toiminnan ymmärtämiseksi paremmin on tärkeää tuntea Ethernetin toiminnan perusteita. (Pigan & Matter 2008, 23.)

Klassisessa Ethernetissä kaikilla asemilla on tasa-arvoiset oikeudet, eli jokainen asema voi välittää tietoa minkä tahansa aseman kanssa milloin tahansa. Tällaista verkkoa kutsutaan jaetuksi Ethernetiksi. Kaikki muut asemat kuuntelevat asemaa, joka lähettää tietoa. Asemat suodattavat tarvittavat paketit, jotka on tarkoitettu sille asemalle eivätkä ota huomioon muita paketteja. Verkkoon yhdistymistä hallinnoi CSMA/CD-menetelmä. Aseman halutessa lähettää tietoa se ensin tarkistaa, että verkko on vapaa. Verkon ollessa vapaa tiedonlähetys voidaan aloittaa. Samaan aikaan tarkistetaan lähettääkö joku muu asema tietoa. Törmäystilanteen sattuessa kaikki asemat, jotka yrittivät lähettää tietoa, lopettavat tiedon lähettämisen. Uusi lähetys aloitetaan satunnaisesti luodun ajan kuluttua. CSMA/CD-menetelmä aiheuttaa sen, että lähetysajan pituus riippuu verkon kuormituksesta eikä sitä voida määritellä etukäteen. Mitä enemmän törmäyksiä verkossa tapahtuu sitä hitaammaksi verkko tulee. (Pigan & Matter 2008, 23.)

Teollisuussovelluksissa käytetään segmentointia, kytkimiä ja suuria tiedonsiirtonopeuksia törmäyksien vähentämiseksi. Profinet Ethernet -verkossa asemat on kytketty toisiinsa verkkolaitteiden kautta kaapeleilla suorina yhteyksinä. Automaatiolaitteistossa verkkokortit ja CP:t (Communications processor) ovat verkon päätepisteitä. Kytkimiä, toistimia ja reitittimiä käytetään välittämään tietoa päätepisteiden välillä. Kierrettyjä kuparijohtimia käytetään kytkemään asemat toisiinsa. Käyttämällä erillisiä lähetys- ja vastaanottolinjoja asemat voivat vastaanottaa ja lähettää tietoa samanaikaisesti ilman törmäyksiä. Tämäntyyppistä kommunikointia kutsutaan full duplexiksi (FDX). Full duplex toimii aina tähden muotoisessa topologiassa, jossa verkon komponentit on kytketty suoraan toisiinsa. Verkko koostuu silloin monista suoraan kytketyistä yhteyksistä. Verkkoon yhdistymisen valvonta on tällöin vain kahden aseman välillä. Tällaisessa verkossa käytetään kytkimiä ja verkkoa kutsutaan kytketyksi Ethernetiksi. Profinet käyttää aina Ethernetiä 100 Mb/s Fast Ethernet yhteydellä full duplexina kytketyssä Ethernetissä. (Pigan & Matter 2008, 23–24.)

Ethernet on pakettipohjainen verkko. Tämä tarkoitta sitä, että siirrettävä tieto jaetaan pienempiin osiin ja näitä osia kutsutaan paketeiksi tai kehyksiksi. Jokainen paketti sisältää

vastaanottoajan ja lähettäjän osoitteet sekä tiedon ja virheiden tarkistuksen. Paketin koko on 64 ja 1526 tavun välillä. Kuviossa 13 on Ethernet paketin rakenne. (Pigan & Matter 2008, 24.)




KUVIO 13. Ethernet-paketti (Mukaillen Pigan & Matter 2008, 24.)

Jokaisella verkon laitteella täytyy olla oma osoitteensa, josta ne tunnistetaan verkossa. Laitteille on valmistajan toimesta asetettu osoite, joka on kiinteä. Tätä osoitetta kutsutaan MAC-osoitteeksi. MAC-osoite on aina 6 tavua pitkä. Kolmea ensimmäistä tavua käytetään tunnistamaan laitteen valmistaja, ja loput ovat valmistajan vapaasti valittavissa. MAC-osoitetta ei yleensä pysty muuttamaan, jotta identtisiltä osoitteilta vältyttäisiin verkossa. (Pigan & Matter 2008, 25.)

### 4.1.1 TCP/IP

TCP/IP on kokoelma protokollia, joista jokainen kantaa kortensa kekoon mahdollistaakseen kommunikoinnin käyttäen yhteistä kieltä. TCP/IP koostuu kahdesta osasta. TCP-protokolla (Transmission Control Protocol) ohjaa tiedon välittämistä. IP-protokollaa (Internet Protocol) käytetään osoittamaan tietokoneeseen tietoverkossa. (Pigan & Matter 2008, 32–33.)

IP-protokolla mahdollistaa datapakettien osoittamisen ja reitittämisen lähettäjältä vastaanottajalle useiden verkkojen yli. Jokainen asema tunnistetaan IP-osoitteesta. IP-osoitetta voidaan verrata nimeen kirjekuoressa. Datapaketteja kutsutaan IP-protokollan yhteydessä datagrammeiksi. Datagrammin oikeellisuutta ei tarkisteta IP tasolla. TCP tasolla on yhteyspainotteinen palvelu tarkistusta varten. IP-datagrammin rakenne on esitetty kuviossa 14.

IP-datagrammi koostuu otsikkoalueesta ja datalohkosta, joka on puolestaan pakattu esimerkiksi Ethernet-pakettiin. (Pigan & Matter 2008, 33.)

<table>
<tr><td>Bit</td><td>0</td><td>4</td><td>8</td><td>16</td><td>31</td></tr>
<tr><td>Byte 0</td><td>Version</td><td>IHL</td><td>Type of service</td><td colspan="2">Packet length (bytes)</td></tr>
<tr><td>Byte 4</td><td colspan="3">Identification</td><td>Flags</td><td>Fragmentation offset</td></tr>
<tr><td>Byte 8</td><td colspan="2">Time to live</td><td>Protocol</td><td colspan="2">Header checksum</td></tr>
<tr><td>Byte 12</td><td colspan="5">Source IP address</td></tr>
<tr><td>Byte 16</td><td colspan="5">Destination IP address</td></tr>
<tr><td>Byte 20</td><td colspan="4">Options</td><td>Padding</td></tr>
<tr><td></td><td colspan="5" rowspan="2">User data</td></tr>
<tr><td></td></tr>
</table>

KUVIO 14. IP-datagrammi (Mukaillen Pigan & Matter 2008, 33.)

Pelkkää IP-protokollaa käytettäessä tiedonsiirto on hyvin epäluotettavaa. Paketit voivat kadota kaapelivikojen tai verkon kuormituksen takia. Paketit voivat myös saapua eri järjestyksessä. TCP-protokolla on yhteys-painotteinen siirtoprotokolla. Tämä tarkoittaa sitä, että lähetetyn tiedon tarkastamisen hoitaa tiedon vastaanottaja. TCP-protokolla käsittelee lähetettävää tietoa tavujonona. Tavujono jaetaan segmentteihin ja numeroidaan. Numerot otetaan segmentin ensimmäisestä tavusta. Tällä tavoin segmentit saapuvat oikeassa järjestyksessä. TCP-protokolla muodostaa yhteyden kommunikointi väylän ja tietokoneen ohjelman välille käyttämällä portteja. Tietokoneen ohjelmille on määrätty tietyt portit, joita ne käyttävät. TCP käyttää IP-osoitteen ja porttinumeron yhdistelmää. Jokainen TCP-paketti sisältää lähettäjän ja vastaanottajan porttinumerot. (Pigan & Matter 2008, 38.)

UDP-protokolla (User Datagram Protocol) toimii samalla periaatteella kuin TCP-protokolla, mutta UDP ei sisällä pakettien järjestyksen tarkistusta. Tällaista tarkistus ei tarvita esimerkiksi lähiverkossa. UDP-protokollaa pidetään yksinkertaisempana ja nopeampana vaihtoehtona TCP-protokollalle. UDP voi olla noin kolme kertaa nopeampi mitä TCP. Profinet käyttää UDP:tä esimerkiksi järjestelmän käynnistyksen yhteydessä. (Pigan & Matter 2008, 38.)

### 4.1.2 IP-osoite

IP-osoite on riippumaton käytetystä laitteistosta, valmistajasta ja verkon fyysisestä tiedon välittämiseen käytetystä menetelmästä. IP-osoitteita käytetään IP-datagrammissa tunnistamaan lähettäjä ja vastaanottaja. Jokaisella Profinet-laitteella täytyy olla IP-osoite. IP-osoite koostuu neljästä tavusta, jotka on erotettu pisteellä. Tavun koon takia IP-osoitteelle voidaan antaa numeroita välillä 0–255. IP-osoite voi olla esimerkiksi tällainen: 192.168.147.112. (Pigan & Matter 2008, 34.)

IP-osoite on jaettu kahteen osaan. Ensimmäistä osaa käytetään tunnistamaan verkko, jossa laite on ja jälkimmäistä osaa tunnistamaan laite kyseisen verkon sisällä. Tätä menetelmää voidaan verrata puhelinnumeroon, jossa aluekoodista tunnistetaan alue ja loppuosasta yksittäinen käyttäjä. (Pigan & Matter 2008, 34.)

Aliverkonpeitteellä jaetaan IP-osoite verkko-osaan ja varsinaiseen laitteeseen. Aliverkonpeitteellä on sama rakenne kuin IP-osoitteella, mutta se kertoo ainoastaan, kuinka monta tavua IP-osoitteen alkuosasta edustaa verkkoa. IP-osoitteen loppuosa jää silloin laitteiden tunnistamiseen. Standardi aliverkonpeitteet vastaavat verkon luokkia A, B ja C (KUVIO 15). Kaikki laitteet, jotka on yhdistetty kytkimillä, kuuluvat samaan aliverkkoon. IP-osoitteiden jakaminen osaverkkoihin muodostaa aliverkon. Tämä jako yleensä vastaa fyysisesti toteutettua jakoa. Verkon luokan jakamista aliverkkoihin kutsutaan aliverkottamiseksi. Kaikki laitteet, jotka kuuluvat samaan aliverkkoon, voivat keskustella keskenään vapaasti. Aliverkonpeite on identtinen kaikille samassa aliverkossa oleville laitteille. Aliverkko rajoitetaan fyysisesti reitittimellä. (Pigan & Matter 2008, 35.)

Osoitteiden ryhmittämiseksi osoitteet on jaettu osoiteluokkiin. IP-osoitteet on jaettu viiteen luokkaan A:sta E:hen. Luokka D on tarkoitettu ryhmälähetyksiä varten eikä luokka E ole käytössä. Luokkia vastaavat aliverkonpeitteet ja IP-osoitteiden jako on esitetty kuviossa 15. (Pigan & Matter 2008, 35.)

| Network class | IP address ranges | Network ID | Host ID | Subnet mask |
|---|---|---|---|---|
| A | 0.0.0.0 -127.255.255.255 | 1 byte | 3 byte | 255.0.0.0 |
| B | 128.0.0.0 - 191.255.255.255 | 2 byte | 2 byte | 255.255.0.0 |
| C | 192.0.0.0 - 223.255.255.255 | 3 byte | 1 byte | 255.255.255.0 |

KUVIO 15. Verkon luokat (Mukaillen Pigan & Matter 2008, 35.)

Tietyt IP-osoitteet jokaisessa luokassa on varattu erityisiin tehtäviin eikä niitä tulisi käyttää. Laitteen tunnuksen sisältäessä vain nollia osoittaa IP-osoite omaa verkkoaan ja sitä kutsutaan verkko-osoitteeksi. Esimerkiksi 192.12.31.0 on luokan C verkossa verkon verkko-osoite. Tätä IP-osoitetta käytettäessä IP-protokolla lähettää jokerimerkki osoitteen verkkoon. Kaikkia asemia, jotka kuuluvat samaan verkkoon, pyydetään lähettämään vastaus. Tämä aiheuttaa verkon ruuhkaantumisen. Reitittimen oletus osoite on yleensä verkko-osoitteesta heti seuraava osoite. Tämä ei ole kuitenkaan vakio sääntö, mutta IP-osoitteiden asettamisen hyviin tapoihin kuuluu ottaa tämä huomioon. 127.0.0.1 on localhost osoite, joka osoittaa aina käytettyyn tietokoneeseen. Paketit, jotka lähetetään tähän osoitteeseen, saapuvat heti takaisin eivätkä pääse verkkoon asti. Osoitetta käytetään testaamaan, että asemat pystyvät lähettämään paketteja itselleen. IP-osoitetta, jonka laite-osan tavut sisältävät vain bittejä, jotka ovat ykkösiä, käytetään lähettämään tietoa jokaiseen laitteeseen samassa aliverkossa. Tätä kutsutan Broadcast-osoitteeksi. (Pigan & Matter 2008, 36.)

## 4.2 Reaaliaikainen Profinet kommunikointi

Reaaliaikainen tarkoittaa sitä, että järjestelmä prosessoi ulkoiset tapahtumat tietyssä ajassa. Järjestelmän reaktion ollessa ennustettavissa puhutaan deterministisestä järjestelmästä. Yleiset vaatimukset reaaliaikaiselle kommunikoinnille ovat siis deterministisyys ja määritelty vasteaika. Vasteaika on yleensä maksimissaan noin 5 millisekuntia. Logiikan prosessorin päätehtävänä on ohjelman suorittaminen, joten tiedon välittäminen reaaliaikaisesti Ethernetiä käyttäen täytyy tehdä kuormittamatta prosessoria liikaa. Samanaikaisesti täytyy ajan, joka kuluu laitteiden välisessä kommunikoinnissa, pysyä pienenä. Yksi mahdollisuus reaaliaikaiseen kommunikointiin on käyttää TCP/IP- tai UDP/IP-protokollaa. Näissä protokollissa on kuitenkin haittapuolensa. TCP- ja UDP-pakettien prosessointi vie aikaa ja silloin lähetys sykli kasvaa. (Pigan & Matter 2008, 44.)

Profinet käyttää optimoitua kommunikointikanavaa reaaliaikaiseen kommunikointiin. Se perustuu OSI-mallin toiseen kerrokseen (KUVIO 16). Tämä reaaliaikainen kanava käyttää datapakettien osoittamiseen pelkästään vastaanottavan laitteen MAC-osoitetta. (Pigan & Matter 2008, 45.)

| Layer | Task | Standard communication | Real-time communication |
|---|---|---|---|
| 7 | Processing | ORPC/RPC | - |
| 6 | Presentation | - | - |
| 5 | Communication | Socket interface | - |
| 4 | Transport | TCP<br>UDP | Real-time protocol (RTC1) (RT over UDP) |
| 3 | Network | IP | IP (RT over UDP) |
| 2 | Data link | Ethernet drive | Real-time protocol (RTC 1-3) |
| 1 | Bit transmission | Ethernet network adapter | Ethernet network adapter |

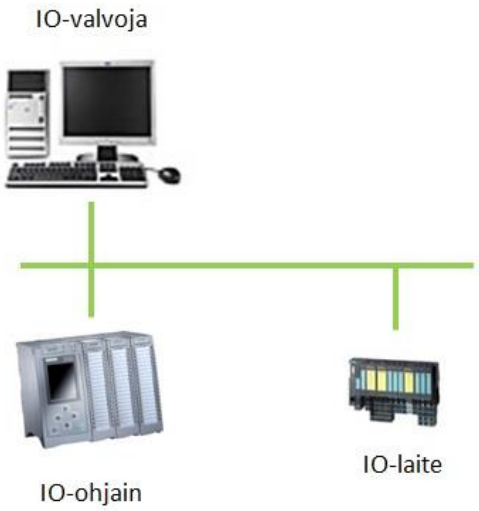
KUVIO 16. Profinet kommunikointiprotokollat OSI-mallissa (Mukaillen Pigan & Matter 2008, 45.)

Syklisen tiedonsiirron aikana yhteyden hallinnan suorittaa ylemmän tason protokolla (esim. TCP/IP). Ensin logiikka yrittää luoda yhteyden vastaanottajan kanssa. Onnistuneen yhteyden luomisen jälkeen logiikka aloittaa tiedonsiirron vastaanottajan kanssa reaaliaikaista protokollaa käyttäen. (Pigan & Matter 2008, 51.)

### 4.3 Profinet IO

Profinet IO on kommunikointikonsepti modulaaristen ja hajautettujen laitteiden toteuttamiseen teollisuus-Ethernetillä. Hajautettu I/O ja kenttälaitteet ovat integroituna Ethernet -kommunikointiin Profinet IO:n avulla. Profinet IO tukee rengas, puu, tähti sekä väylä topologioita. Tämä toteutetaan kytkin toiminnolla, joka on integroitu Profinet IO -laitteisiin. Muita kenttäväylä ratkaisuja voidaan yhdistää Profinet IO sovelluksiin käyttämällä välityspalvelimia. Syklinen tiedonsiirto IO-ohjaimen ja IO-laitteen välillä perustuu lähettäjä/vastaanottaja-malliin. Yksi lähetys sykli voi sisältää maksimissaan 1440 tavua tietoa. IO-laitteet on kuvattu GSD-tiedostoissa. (Pigan & Matter 2008, 69.) GSD-tiedosto kuvaa IO-laitteen ominaisuuksia. Laitteiden valmistajien täytyy luoda tällainen tiedosto kaikkiin laitteisiin. (Pigan & Matter 2008, 94.)

Kommunikoinnin näkökulmasta kaikilla Profinet IO -laitteilla on yhtäläiset oikeudet Ethernetissä. Tämä eroaa esimerkiksi Profibus-väylästä, jossa kommunikointi noudattaa master/slave-mallia. Profinet IO -laitteita on kuitenkin erityyppisiä ja kommunikointitapa riippuu laitteen tyypistä. Profinet IO -laitteiden luokat on esitelty kuviossa 17. Luokkia on itse asiassa neljä. Yksi niistä on parametri palvelin. Palvelimen toimintaa ei kuitenkaan käsitellä tässä. (Pigan & Matter 2008, 69–71.)

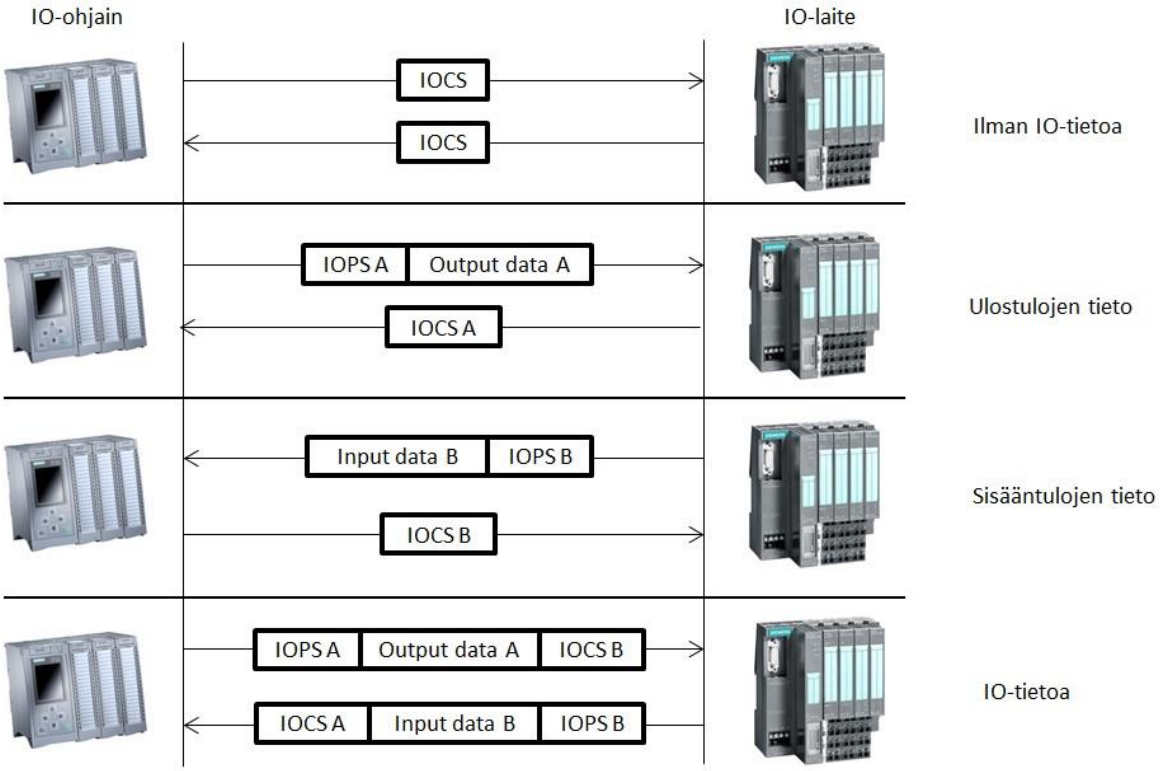



| Laiteluokka | Toiminta |
|---|---|
| IO-valvoja | Esimerkiksi PC tai HMI, jota käytetään IO-laitteiden ja IO-ohjainten käyttöönottoon ja diagnostiikkaan. |
| IO-ohjain | Ohjelmoitava logiikka. Vähintään yksi IO-ohjain täytyy konfiguroida Profinet IO -kokoonpanoon. |
| IO-laite | Hajautettu kenttälaite, joka välittää lähettää ja vastaanottaa tietoa yhdeltä tai useammalta IO-ohjaimelta. |

KUVIO 17. Profinet IO -laitteiden luokat (Mukaillen Pigan & Matter 2008, 71.)

Järjestelmän käynnistyksen jälkeen I/O-tietoa lähetetään syklisesti IO-ohjaimen ja IO-laitteen välillä. IO-laitteen ja IO-ohjaimen välinen kommunikointi on aina reaaliaikaista. Jokainen I/O-tieto osio sisältää kaksi attribuuttia. Attribuutit mahdollistavat IO-ohjaimelle ja IO-laitteelle tiedon laadun arvioinnin. Attribuutit ovat nimeltään IOPS (IO Provider Status) ja IOCS (IO Consumer Status). Lähettäjä siirtää IOPS-attribuutin tiedonlähetyksen yhteydessä. Attribuuteilla voi olla kaksi arvoa joko hyvä tai huono. IOCS-attribuutti voidaan lähettää lähettäjällä vasta kun vastaanottaja on saanut tiedon vastaan. Attribuutit kuvaavat lähettäjän tai vastaanottajan tilaa. Esimerkiksi, jos IO-laite lähettää sisääntuloista tietoa IO-ohjaimella ja IOPS-attribuutti on huono, on IO-laitteessa jotain vikana. Kuviossa

18 on kuvattuna attribuuttien toiminta IO-ohjaimen ja IO-laitteen välisessä kommunikoinnissa. (Pigan & Matter 2008, 79–80.)




KUVIO 18. IOPS ja IOCS (Mukaillen Pigan & Matter 2008, 80.)

IO-laitteille annetaan nimi ennen kuin varsinainen yhteys muodostetaan. Nimen määrittää IO-valvoja, ja se tallennetaan IO-laitteen retentiiviseen muistiin. Nimi on vapaasti käyttäjän valittavissa, mutta sen pitää olla DNS (Domain name services) merkintätavan mukainen. Järjestelmän käynnistyksen aikana IO-ohjain etsii laitteita niiden nimien perusteella. Laitteen löydettyään määritellään MAC-osoitetta vastaava IP-osoite. TCP/IP-protokolla yrittää etsiä laitteen MAC-osoitetta vastaavaa IP-osoitetta. TCP/IP- protokolla lähettää kaikille aliverkon laitteille pyynnön etsiä IP-osoitetta, jos IP-osoitetta ei löydy kyseistä laitetta ei ole olemassa aliverkossa. (Pigan & Matter 2008, 83–84.)

### 4.4 IWLAN

Profinettiä voidaan käyttää myös radioverkon suunnitteluun teollisuuden vaatimusten mukaisesti. Radioverkko on yleensä IWLAN-verkko (Industrial WLAN). Tiedonsiirtonopeu-

det 1 Mb/s tai 54 Mb/s ilman full duplexia ovat sallittuja IWLAN-verkossa. Radioverkkoja käytetään sovelluksissa, joissa vaaditaan liikkuvuutta ja joustavuutta. Radioverkoissa käytetään sähkömagneettisia aaltoja välittämään tietoa. (Pigan & Matter 2008, 323.)

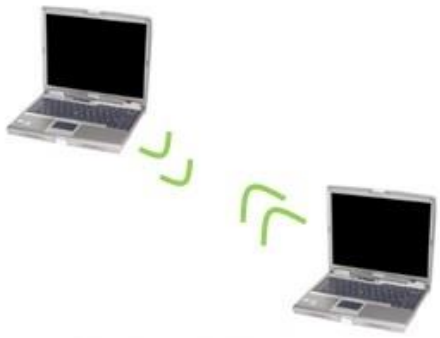
WLAN (Wireless Local Area Network) on määritelty IEEE standardissa 802.11. Standardi noudattaa Ethernetistä tuttua konseptia, mutta WLAN-verkossa käytetään LLC-kehystä (Logical Link Control) Ethernet-kehyksen sijaan. TCP/IP-protokollan sijaan voidaan käyttää mitä tahansa protokollaa, mutta TCP/IP on suosituin. WLAN-verkoissa käytetään erilaista vuoronvarausta, jota kutsutaan CSMA/CA-vuoronvaraukseksi. Se on samankaltainen kuin Ethernetissä käytettävä CSMA/CD-kilpavaraus, mutta törmäysten havaitsemisen sijaan niiltä pyritään välttymään. Törmäyksien välttämiseksi lähettävä asema kertoo muille asemille, että kuinka kauan se arvioi kehyksen lähettämisen kestävän. Fyysistä kantoaallon kuuntelua käyttäen asema varmistaa, että kanava on vapaa. Lähettävä asema varaa kanavan itselleen verkonvarausvektorilla ja muut asemat kuulevat sen. Kehyksien lähetysten välillä käytetään erilaisia viiveitä. Lähetyksen loppuessa aikaisemmin kuin arvioitiin alkaa viiveen laskenta vasta kanavan varauksen päättymisen jälkeen. (Puska 2005, 26–31.)

Radiotekniikka perustuu sähkömagneettisten aaltojen lähettämiseen ja vastaanottamiseen. Radioaaltoja lähetettäessä ne levittyvät kolmeen ulottuvuuteen. Erilaiset esteet vaikuttavat aaltojen leviämiseen heijastamalla, hajottamalla ja absorboimalla niitä. Seinät, huonekalut ja ihmiset heijastavat tai absorboivat radioaaltoja aiheuttaen aaltojen vaimenemisen. Aallot vaimenevat eri tavalla riippuen esteen materiaalista ja aallon taajuudesta. Johtavat materiaalit heijastavat aallot täysin ja ei-johtavat materiaalit heijastavat osan ja absorboivat loput. Esineen kulmaan osuessa aallot hajaantuvat. Useat päällekkäiset radioaallot aiheuttavat myös häiriötä signaaliin. (Pigan & Matter 2008, 324.)

WLAN-verkossa signaalia ei lähetetä vain yhdellä taajuudella. Riippuen signaalin modulointitavasta käytetään erilaisia taajuuksia kantoaallon ylä- ja alapuolella. Tästä johtuen eri radiolähettimiä ei voi asettaa lähelle samaa taajuusaluetta ilman häiriötä. Häiriövapaan toiminnan saavuttamiseksi WLAN-verkossa taajuusalue on jaettu kanaviin. Esimerkiksi 2,4 GHz alue on jaettu 13 kanavaan 2,412–2,472 GHz välillä. Kanavien väli on silloin 5MHz. (Pigan & Matter 2008, 325.)

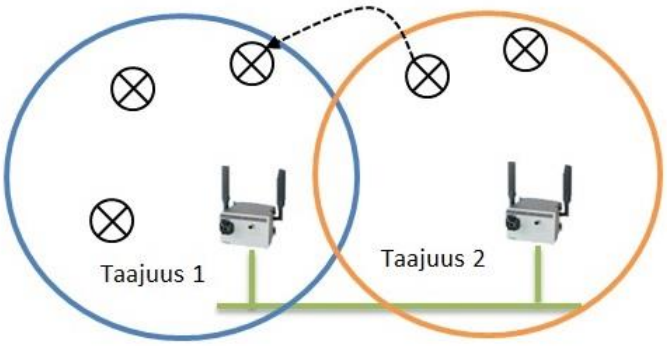
WLAN-verkossa on kaksi perusteellisesti eroavaa verkkoa Ad-hoc-verkko ja erilaiset topologiat infrastruktuuriverkossa. Ad-hoc-verkko muodostuu suorasta yhteydestä kahden aseman välillä. Infrastruktuuriverkossa asemat ovat yhteyksissä toisiinsa tukiasemien (access point) kautta. Tukiasema on WLAN-asema, jossa on vähintään yksi WLAN-liitäntä (antenni) ja yleensä yksi LAN-liitin (Ethernet-portti). (Pigan & Matter 2008, 326.)

Ad-hoc-verkkoa käytetään väliaikaiseen tiedonsiirtoon lyhyillä välimatkoilla esimerkiksi kahden kannettavan tietokoneen välillä (KUVIO 19). Radioverkon tunnistamiseksi useiden radioverkkojen joukosta käytetään SSID-tunnusta. (Pigan & Matter 2008, 325.)



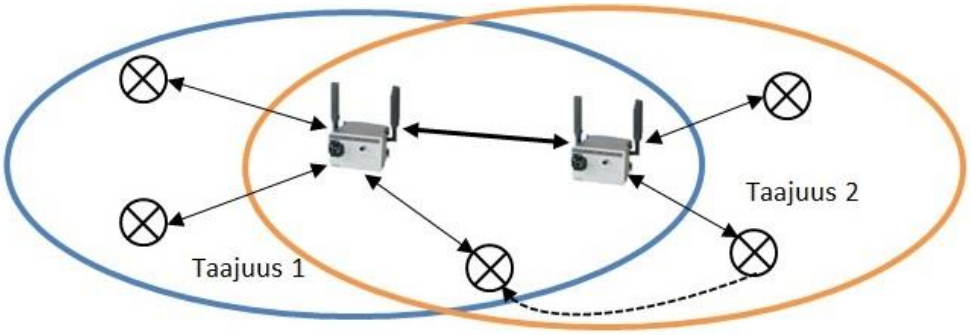
KUVIO 19. Ad-hoc-verkko (Mukaillen Pigan & Matter 2008, 326.)

Infrastruktuuriverkossa tukiasema ohjaa liikennettä asemien välillä. Yksinkertaisimmassa tapauksessa ryhmä asemia on yhteyksissä yhteen tukiasemaan. Useiden tukiasemien ollessa yhteydessä toisiinsa Ethernetin kautta voidaan radioverkon kantamaa pidentää. Siirtymistä yhdestä tukiasemasta toiseen kutsutaan verkkovierailuksi (roaming). Tukiasemien radioalueiden täytyy olla päällekkäin, jotta vaihto toimii ilman keskeytyksiä (KUVIO 20). (Pigan & Matter 2008, 326.)




KUVIO 20. Infrastruktuuriverkko (Mukaillen Pigan & Matter 2008, 327.)

Infrastruktuuriverkon erikoistapaus on WDS-verkko, jossa tukiasemat eivät ole yhteyksissä toisiinsa kaapeleilla vaan radioverkon kautta (KUVIO 21). Tästä johtuen asemien välimatka on lyhyempi kuin aseman kantama ja tiedonsiirtonopeudet ovat pienempiä koska tukiasemien täytyy jakaa sama ilmatila väliaineena. (Pigan & Matter 2008, 326.)

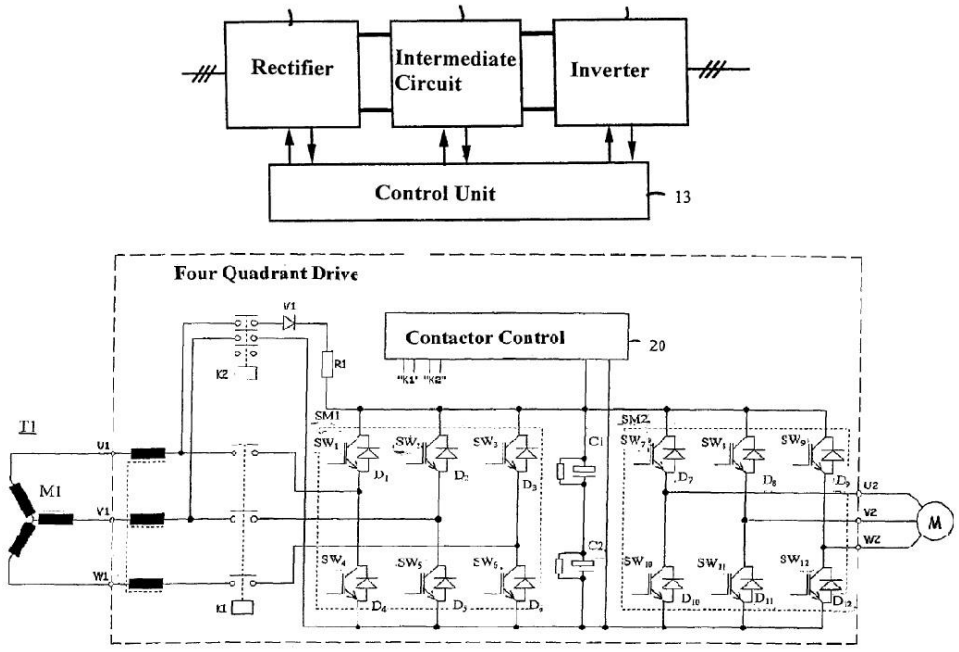



KUVIO 21. WDS-verkko (Mukaillen Pigan & Matter 2008, 327.)

# 5 SINAMICS G120 TAAJUUSMUUTTAJA JA CU250S-2 OHJAUSYKSIKKÖ

Taajuusmuuttajat muuttavat sähköverkon vakiotaajuista ja -jännitteistä sähköä säätääkseen vaihtosähkömoottorin nopeutta. (Niiranen, J 1999, 39). Taajuusmuuttaja voi käyttää erilaisia tapoja moottorin säätöön, mutta tässä käsitellään vain välipiirilliset taajuusmuuttajat. Välipiirillinen taajuusmuuttaja koostuu tasasuuntaajasta, tasajännitepiiristä ja vaihtosuuntaajasta. (Niiranen, J 1999, 48.)

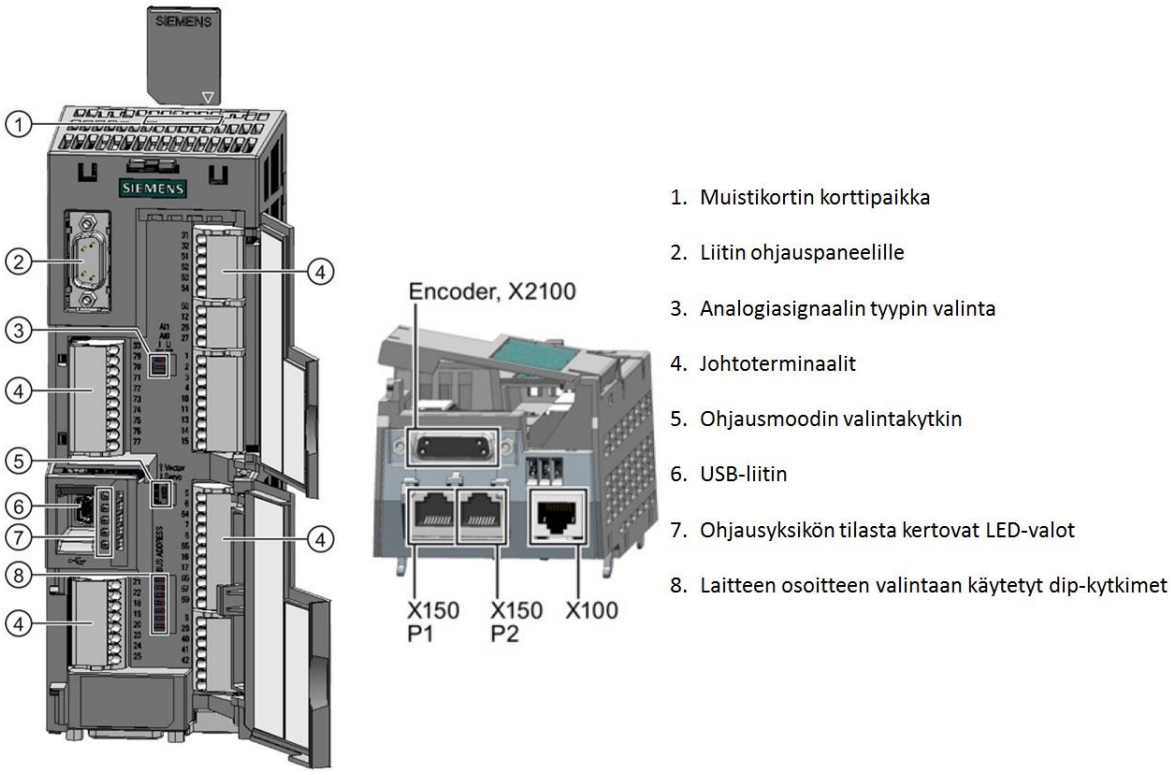
Tasasuuntausyksikössä käytetään diodeja tai tyristoreita muuttamaan vaihtosähköä tasasähköksi. Diodit ovat puolijohdekomponentteja, jotka päästävät virtaa vain yhteen suuntaan. Tyristorit ovat diodien kaltaisia komponentteja, mutta niitä voidaan ohjata sähkövirralla johtavaksi. Tyristoreilla voidaan siis muuttaa tasajännitteen suuruutta. Tasajännitepiirissä on kondensaattori, joka tasaa tasavirran aaltoisuutta tasasuuntauksen jälkeen. Vaihtosuuntaajassa käytetään myös tyristoreita, mutta niillä kytketään tasajännitettä eri taajuuksilla sähkömoottorin käämeihin. Tämä saa vaihtosähkömoottorin pyörimään eri nopeuksilla. Tyristoreiden tarvitseman loistehon takia niitä korvaamaan käytetään IGBT-transistoreja. Kuviossa 22 on modernin taajuusmuuttajan periaatekaavio ja kytkentäkaavio. (Niiranen, J 1999, 40–47.)




KUVIO 22. Taajuusmuuttajan rakenne (United States Patent 2005, 2–3.)

Sinamics G120 taajuusmuuttuja koostuu ohjausyksiköstä ja tehoyksiköstä. Ohjausyksikkö ohjaa ja valvoo tehoyksikköä ja sähkömoottoria. (CU250S-2 Control Unit 2013, 23.) G120 on rakenteeltaan modulaarinen eli ohjausyksikkö ja tehoyksikkö voidaan valita erikseen käyttökohteeseen sopivaksi. Ohjausyksikkö asetetaan tehoyksikön päälle. (Sinamics G 2014.) CU250S-2 -ohjausyksikkö mahdollistaa vektorisäädön takaisinkytkennällä pulssianturilta. Pulssiantureiden liitäntöjä siitä löytyy eri tyyppejä. Tyypilliset käyttökohteet ovat suuren I/O määrän omaavat sovellukset sekä suuren nopeuden säätöä vaativat sovellukset. CU250S-2 -ohjausyksikköä on saatavilla Profibus DP, Profinet, RS485 ja CANopen yhteensopivana. (Automation.com 2013.)

Pulssianturi kiinnitetään moottorin akseliin ja yhdistetään ohjausyksikköön kaapelilla. Kaapeli voidaan liittää suoraan yksikön johtoterminaaliin tai SUB-D -liittimeen (KUVIO 23). Ohjausyksikössä on myös DRIVE-CLiQ -liitin. (CU250S-2 Control Unit 2013, 66.) DRIVE-CLiQ on Siemensin kehittämä Ethernet-pohjainen tiedonsiirtoprotokolla (Leine & Linde 2012). Kuviossa 23 X2100 on D-SUB -liitin, X150 liittimet ovat Profinet-liittimiä ja X100 on DRIVE-CLiQ -liitin. Samassa kuviossa on ohjausyksikkö päältäpäin kuvattuna. (CU250S-2 Control Unit 2013, 56–57.)

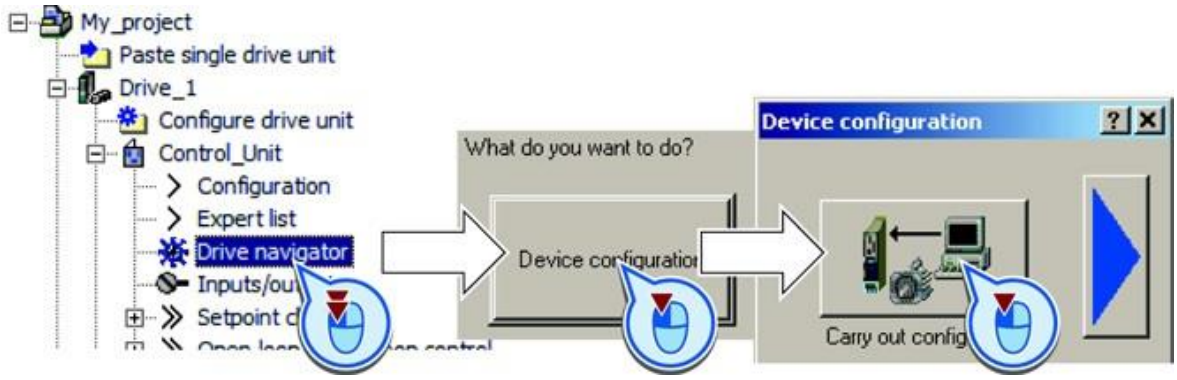



KUVIO 23. CU250S-2 liitännät (Mukaillen CU250S-2 Control Unit 2013, 56–57.)

## 5.1 Taajuusmuuttajan käyttöönotto

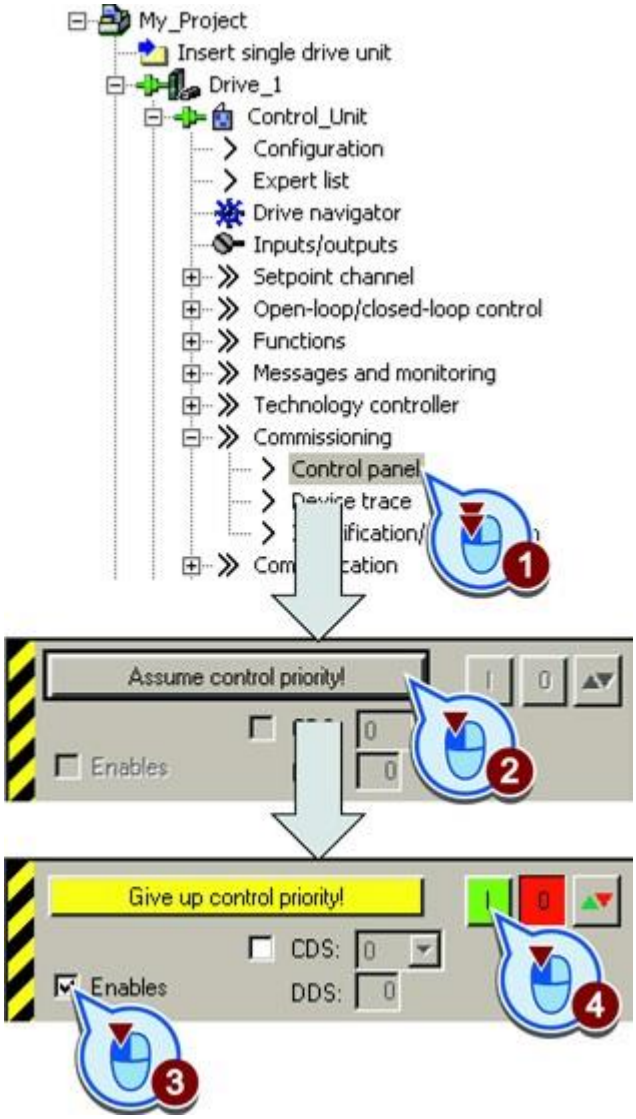
Starter -ohjelmaa käytetään taajuusmuuttajien parametrien asettamiseen, käyttöönottoon ja vianetsintään (STARTER 2014). Käyttöönotto koostuu kolmesta askeleesta projektin luomisesta Starterissa, taajuusmuuttajan määrittelystä ja määrittelyn lataamisesta taajuusmuuttajaan. Projektin luomisen ja kommunikointitavan valitsemien jälkeen voi aloittaa taajuusmuuttajan määrittelyn. (CU250S-2 Control Unit 2013, 85.)

Kuviossa 24 on näytetty, miten määrittely aloitetaan. Määrittelyn alussa valitaan ohjausmoodi ja mitä toimintoja halutaan käyttää. Käytettäessä pulssianturia valitaan Basic positioner -toiminto ja ohjausmoodiksi speed control with encoder. Seuraavaksi määritellään johtoterminaalin kytkentä. Johtoterminaalin kytkennällä valitaan myös, käytetäänkö kenttäväylää taajuusmuuttajan ohjaamiseen. Kytkennän valinnan jälkeen valitaan moottori. Moottori valitaan valmiilta listalta. Moottorin kilpitiedot voi syöttää myös itse, jos käytetty moottori ei ole listalla. Moottorin tietojen tunnistamiseksi valitaan identify motor data at standstill and with motor rotating. Tällä asetuksella taajuusmuuttaja optimoi nopeussäätimen pyörittämällä moottoria. Näiden asetuksien jälkeen valitaan nopeusrajat, virtaraja sekä kiihdytys- ja jarrutusramppien ajat. Seuraavaksi valitaan, halutaanko taajuusmuuttaja palauttaa tehdasasetuksille ja laske moottorin tiedot vai halutaan pelkästään laskea moottorin tiedot. Pulssianturin kytkentä ja tyyppi valitaan seuraavaksi. Ensin valitaan, mihin pulssianturi on kytketty ja sitten valitaan pulssianturi listalta. Pulssianturin tiedot voi syöttää myös itse, jos käytetty pulssianturi ei ole listalla. Pulssianturin määrittelyn jälkeen määritellään paikoitussovelluksen tarkkuus ja välityssuhde. Tämän voi kuitenkin jättää myöhemmäksi koska tarkkuuta ja välityssuhdetta voi muuttaa myöhemmin sopivaksi. (CU250S-2 Control Unit 2013, 87–90.)




KUVIO 24. Määrittelyn aloittaminen (CU250S-2 Control Unit 2013, 87.)

Moottorin tunnistaminen tehdään määrittelyn jälkeen. Tunnistus tehdään valitsemalla control panel Starterissa. Klikkaamalla assume control priority -nappia, ja valitsemalla enables-valintaruutu, voidaan tunnistus aloittaa (KUVIO 25). Moottorin käynnistäminen aloittaa moottorin tunnistamisen. Mittaukset voivat kestää useita minuutteja. Mittauksien jälkeen taajuusmuuttaja sammuttaa moottorin. Seuraava moottorin käynnistäminen aloittaa nopeussäätimen optimoinnin, jos nopeussäätimen optimointi on valittu määrittelyssä. Tunnistamisen jälkeen tallennetaan tiedot moottoriin klikkaamalla save RAM to ROM -nappia. (CU250S-2 Control Unit 2013, 93.)




KUVIO 25. Moottorin tunnistaminen (CU250S-2 Control Unit 2013, 93.)

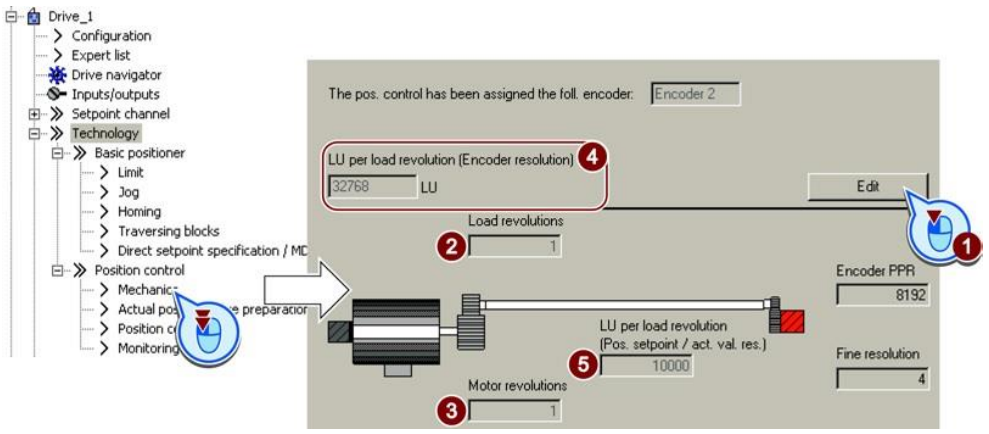
Ohjausyksiköitä on saatavilla eri versioita riippuen siitä mitä kommunikointi tapaa haluaa käyttää. Profinet versiossa on kaksi RJ45 pistoketta Profinettiin liittymistä varten. Taa-

juusmuuttaja liitetään logiikkaan ja ohjelmointikoneeseen Profinet-kaapeleilla. Taajuusmuuttajan ja logiikan välisen kommunikoinnin mahdollistamiseksi tarvitaan laitetiedosto. Laitetiedostoa kutsutaan GSDML-tiedostoksi. GSDML-tiedosto ladataan logiikkaan, joka ohjaa Profinet-verkkoa. GSDML-tiedosto löytyy internetistä tai se voidaan siirtää taajuusmuuttajalta muistikortille. (CU250S-2 Control Unit 2013, 112–114.)

Käyttöönotossa täytyy valita johtoterminaalin kytkentä, joka on tarkoitettu kenttäväylän kanssa käytettäväksi. Kenttäväylän käyttämä telegrammi valitaan Starterilla tai taajuusmuuttajan käyttöpaneelilla. Parametrillä 0922 valitaan käytetty telegrammi. Paikoitus käyttöön tarkoitettuja telegrammeja on neljä ja yksi avoin telegrammi, joka on vapaasti määriteltävissä. (CU250S-2 Control Unit 2013, 115.)

## 5.2 Paikoittajan käyttöönotto

Normaalin käyttöönoton jälkeen määritellään paikoittaja. Tämä tehdään myös Starter -ohjelmalla (KUVIO 26). Taajuusmuuttaja laskee akselin asennon käyttäen LU-yksikköä. LU on riippumaton siitä mitä taajuusmuuttaja ohjaa. Ensimmäisenä täytyy määritellä, mikä pituus tai kulma vastaa yhtä LU:ta eli paikoituksen resoluutio. Resoluutio pitää valita sopivaksi akselin kulkeman maksimimatkan mukaan, jos resoluutio on liian tarkka, paikoittaja ei pysty näyttämään paikkatietoa koska se kasvaa liian suureksi. Taajuusmuuttaja antaa vikailmoituksen, jos tapahtuu ylivuoto. (Basic Positioner 2013, 29.)




KUVIO 26. Pulssianturin signaalin skaalaus (Basic Positioner 2013, 30.)

Taajuusmuuttaja rajoittaa paikoitusalueen ohjelmallisesti ja hyväksyy vain paikkaohjeet, jotka ovat rajojen sisällä. Näiden rajoitusten lisäksi voidaan käyttää myös digitaalisia sisääntuloja. Sisääntuloihin voi kytkeä esimerkiksi anturit, jotka on sijoitettu paikoitusalueen loppuun ja alkuun. Liikuteltavan kuorman ajautuessa jompaankumpaan anturiin, taajuusmuuttaja pysäyttää moottorin. Rajat ja digitaaliset tulot määritellään Starterin Limit-ikkunassa. Siellä voidaan myös rajoittaa moottorin kiihtyvyyttä ja nopeutta. (Basic Positioner 2013, 36.)

Taajuusmuuttajan saatuaan paikkaohjeen lähtee se ajamaan kuormaa kohti tätä paikkaa. Saavuttuaan lähelle paikkaohjetta ryhtyy se valvomaan akselin liikettä. Akselin liikkuessa sen jälkeen kun taajuusmuuttaja on jo havainnut, että paikkaohje on saavutettu taajuusmuuttaja antaa vikailmoituksen. Vikailmoitus tulee myös siitä, jos paikkatieto ei saavuta paikkaohjetta tietyssä ajassa akselin pysähtymisestä. Aluetta, jossa taajuusmuuttaja aloittaa akselin liikkeen tarkkailun, voidaan muuttaa Starterin Monitoring-ikkunassa. Tarkkailuaikoja akselin pysähtymisellä ja paikkaohjeen saavuttamisella voidaan muuttaa myös sieltä. Monitoring-ikkunassa voidaan myös määritellä kuinka suuri ero paikkaohjeella ja todellisella paikkatiedolla saa olla. Taajuusmuuttaja ilmoittaa viasta, jos ero kasvaa yli asetetun rajan. (Basic Positioner 2013, 44–46.)

Käytettäessä inkrementaalianturia paikkatiedon ilmoittamiseen taajuusmuuttaja menettää paikkatiedon sähkökatkoksen jälkeen. Sähköjen palattua takaisin päälle taajuusmuuttaja ei enää tiedä, missä kuorma on. Absoluuttianturit muistavat paikkansa sähkökatkon jälkeen. Ajamalla kuorma referenssipisteeseen taajuusmuuttaja tietää taas kuorman paikan. Starterissa voidaan määritellä erilaisia referenssitapoja eri pulssianturityypeille. Yksi tapa on käyttää anturia, joka ilmoittaa, milloin kuorma on referenssipisteessä. Starterissa voidaan määritellä LU:na paikka, jota referenssipiste vastaa. Referenssipiste voidaan määritellä Homing-ikkunassa. Referenssipisteen saavuttamisen jälkeen voidaan taajuusmuuttajaa käskeä ajamaan tietty matka LU:na tai ajamaan pulssianturin nollapisteeseen. (Basic Positioner 2013, 49–54.)

Taajuusmuuttajaan voidaan luoda 16 lohkoa, jotka antavat paikoitusohjeen taajuusmuuttajalle. Ne voidaan suorittaa järjestyksessä tai valita niistä yksi ja suorittaa vain se. Lohkojen yli voidaan myös hypätä seuraavaan lohkoon. Yksi lohko sisältää järjestysnumeron, tehtävän, paikoitusmoodin, paikkaohjeen, nopeuden, kiihtyvyyden kiihdyttäessä ja jarruttaessa

ja ehdon, jolla siirrytään seuraavaan lohkoon. Tehtävällä kerrotaan mitä taajuusmuuttajan halutaan tekevän. Taajuusmuuttaja voidaan käskeä esimerkiksi paikoittamaan tai kulkemaan kunnes tullaan paikoitusalueen päähän. Paikoitusmoodilla valitaan, halutaanko suhteellinen vai absoluuttinen paikoitus. Paikkaohjeeseen kirjoitetaan haluttu paikka LU:na. Seuraavaan askeleeseen siirtymiseen voidaan luoda ehtoja tai käskeä taajuusmuuttajaa pysähtymään lohkon suorittamisen jälkeen. Seuraavaan lohkoon siirtyminen voidaan tehdä esimerkiksi kun akseli on pysähtynyt tai akselin pysähdyttyä ulkoisella signaalilla. Kuviossa 27 on esimerkki kolmesta lohkosta. Lohkot voi määritellä Starterin Traversing blocks -ikkunassa. Ikkunassa voi valita signaalit lohkojen ohjaamiseen sekä tarkkailla diagnostiikkatietoja online-moodissa. (Basic Positioner 2013, 69–70.) Taajuusmuuttajalle on mahdollista antaa myös paikka- ja nopeusohje suoraan. Kertomalla taajuusmuuttajalle, että halutaan käyttää MDI-moodia, voidaan paikkaohje antaa suoraan logiikalta. (Basic Positioner 2013, 82.)

| Ind. | No. | Job | Par. | Mode | s | v | a | -a | Advance |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | POSITIONING | 0 | RELATIVE | 10000 | 2000 | 100 | 100 | END |
| 2 | 2 | POSITIONING | 0 | RELATIVE | 10000 | 5000 | 100 | 100 | END |
| 3 | 3 | POSITIONING | 0 | ABSOLUTE | 0 | 5000 | 100 | 100 | END |

KUVIO 27. Paikoituslohkoja (Mukaillen Basic Positioner 2013, 81.)

# 6 ESIMERKKIOHJELMAT

Tässä luvussa käsitellään kaikki esimerkkiohjelmat, jotka loin Apexin käyttöön TIA Portal-ohjelmistolla. Esimerkkiohjelmien tekemisen tarkoituksena oli tutkia TIA Portalin mahdollisuuksia ja uusia Siemensin valmistamia laitteita. Näiden ohjelmien pohjalta loin ohjeita Apxein sisäiseen käyttöön.

## 6.1 Yritys

Apex Automation Oy on kokkolalainen insinööritoimisto, joka on perustettu vuonna 1993. Yritys on erikoistunut sähkö- ja automaatiosuunnitteluun. Yrityksen toiminta-ajatuksena on parantaa asiakkaan kilpailukykyä suurentamalla sen tuotantokapasiteettia ja parantamalla tuotteen laatua. Yrityksen arvot ovat asiakasläheisyys, luottamus, laatu ja osaaminen. Apex Automation Oy tarjoaa palvelujaan energian tuotanto- ja jakeluyhtiöille, teollisuuden laite- ja järjestelmätoimittajille, prosessiteollisuudelle sekä sähköasennusliikkeille. (Apex Automation Oy 2011.)

Apex Automation Oy:n keskeisimpiä palveluja ovat sähkösuunnittelu, instrumentointi, sähköturvallisuus, sovellusohjelmointi, koneturvallisuus, keskusvalmistus ja koulutus- ja konsultointipalvelut. Sovellusohjelmointipalveluihin kuuluvat laitosten, tuotantolinjojen ja yksikköprosessien automatisointi. Palveluihin kuuluu myös ylläpito. Yrityksellä on ohjelmointityökalut ja kokemusta monista logiikkamerkeistä. Yritys tekee läheistä yhteistyötä suurien toimittajien kanssa. Tärkeimmät toimittajat ovat Siemens, Omron ja Allen-Bradley. Käyttöliittymä puolella Apex Automation Oy:llä on kokemusta WinCC, InTouch, ClearSCADA, Citect ja CX-Supervisor ohjelmistoista. (Apex Automation Oy 2011.)

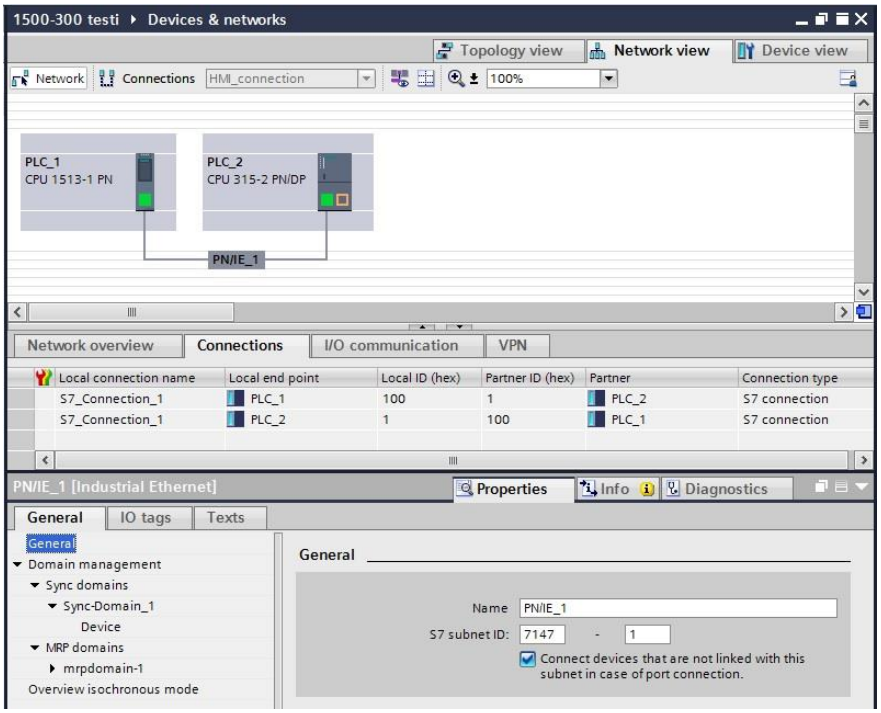
## 6.2 S7-1500- ja S7-300 -logiikan välinen kommunikointi

Ensimmäisiä ongelmia, joita lähdin ratkomaan Apexilla, oli 1500- ja 300 -logiikan välinen kommunikointi. Tällainen kokoonpano oli käytössä eräässä projektissa ja kahden logiikan välillä haluttiin välittää tietoa Profinet-väylää käyttäen. Työkaveri neuvoi ottamaan yhteyt-

tä Siemensiin ja kysyä, että mitä lohkoja tarvitaan. Tein tukipyyntöilmoituksen Siemensin nettisivuilla ja kuvailin ongelman heille. Sain vastauksen sähköpostiini parin päivän kuluttua, jossa neuvottiin käyttämään PUT- ja GET-lohkoja.

Ryhdyin etsimään tietoa PUT- ja GET-lohkoista Siemensin nettisivuilta ja Step 7- ohjelmiston manuaalista. Siemensin nettisivuilta löysin PDF-dokumentin, jossa oli ohjeet ohjelman tekemiseen kahden 1500 -logiikan väliseen kommunikointiin käyttäen PUT- ja GET-lohkoja. Tätä ohjetta soveltamalla tein esimerkkiohjelman 1500- ja 300 -logiikan välisestä tiedonsiirrosta.

Ohjelman tekeminen aloitetaan yleensä lisäämällä käytettävät komponentit projektiin. Testilaitteistona käytettiin 1513-1 PN -logiikkaa ja 315-2 PN/DP -logiikkaa sekä virtalähdettä. Virtalähde ei kuitenkaan näy ohjelmassa, koska sen tehtävä on vain tuottaa käyttöjännite logiikoille. Logiikat toimivat 24 voltin tasajännitteellä. Kuviossa 28 on TIA Portal -ohjelmiston Devices & networks -ikkuna, jossa voidaan määritellä käytettävät yhteydet ja niiden asetukset.

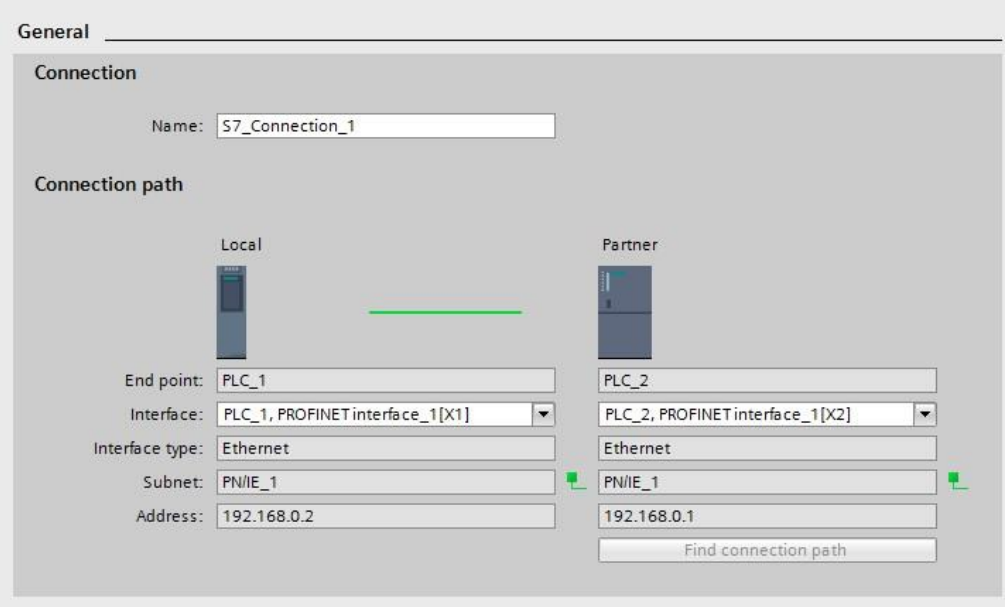



KUVIO 28. Devices & neworks –ikkuna

Logiikoiden välinen kommunikointi toteutettiin käyttäen S7-kommunikaatiota. S7-yhteys voidaan luoda kahden logiikan välille valitsemalla ensin Connections ja sitten alasvetova-

likosta valitsemalla S7 connection. Tämän jälkeen valitaan hiirellä jompikumpi logiikoista ja raahataan hiirellä sininen viiva logiikasta toiseen. Tätä ei ole kuitenkaan pakko tehdä manuaalisesti. S7-yhteys luodaan automaattisesti, kun ohjelmaan lisätään PUT- tai GET-lohko. Lohkojen asetuksissa voidaan määritellä S7-yhteys.

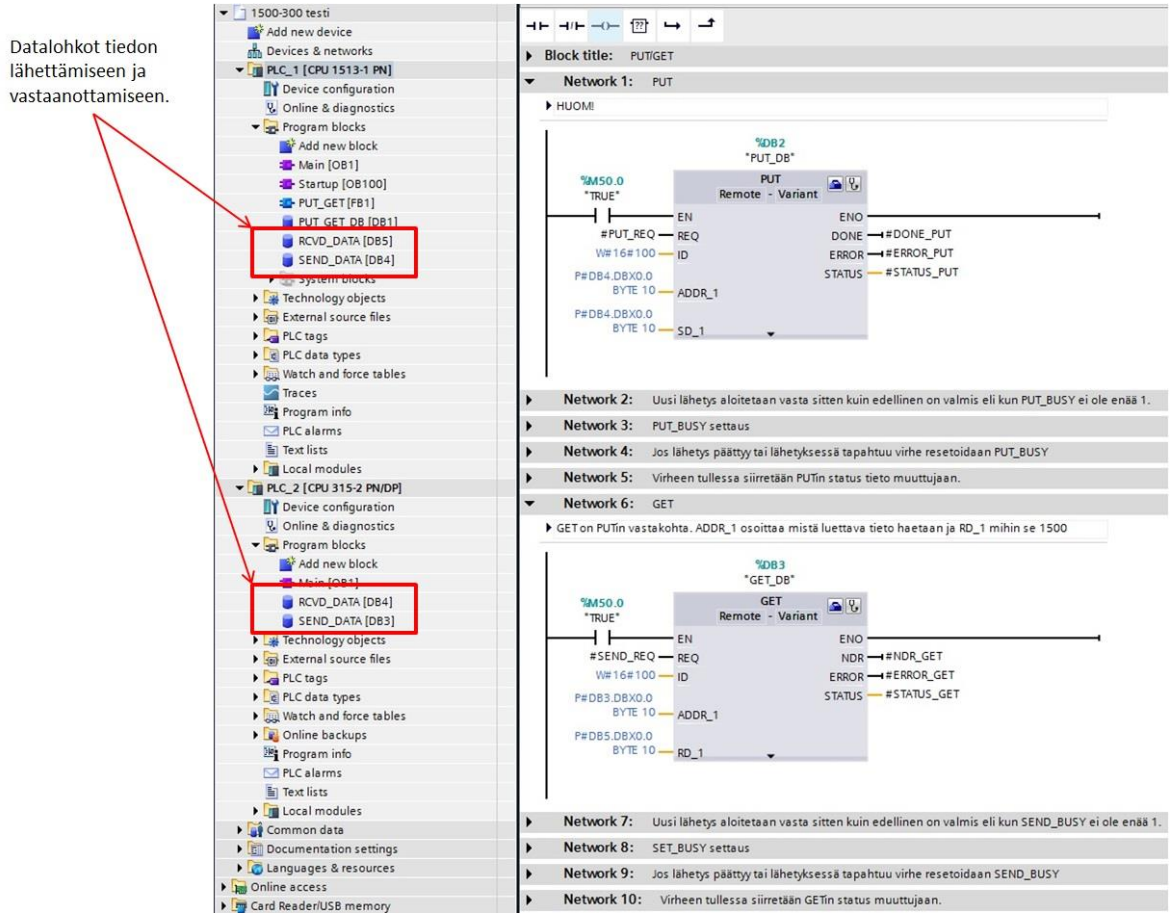
Kuviossa 29 on luodun S7-yhteyden yleiset asetukset. Molemmille logiikoille pitää kertoa mihin logiikkaan yhdistytään, mihin logiikan porttiin Ethernet-kaapeli on kytketty, mitä verkkoa käytetään ja logiikoiden IP-osoitteet. Kuviossa 29 on 1500 -logiikan asetukset, joten 300 -logiikka näytetään sen partnerina. 300 -logiikan puolella asiat ovat toisinpäin. S7-yhteyden luomisen yhteydessä logiikoille luodaan ID-numerot, joista ne voidaan tunnistaa. Nämä määräytyvät automaattisesti mutta niitä voi vaihtaa. Esimerkkiohjelmassa 1500 -logiikka muodostaa yhteyden 300 -logiikkaan. Se kirjoittaa tietoa tai lukee sitä 300 -logiikan datalohkoista. Tämän takia on S7-yhteyden asetuksista otettu Active connection establishment-asetus pois päältä 300 -logiikan puolelta. 300 -logiikka sisältää vain datalohkot, joihin kirjoitetaan tai niistä luetaan tietoa. Minkäänlaista ohjelmakoodia ei siis tarvita 300 -logiikan puolella. 1500 -logiikan asetuksista täytyy kommunikointi PUT- ja GET-lohkoilla sallia.




KUVIO 29. S7-yhteyden asetukset

PUT- ja GET-lohkot eroavat toisistaan vain niiden toiminnassa. Molemmat käyttävät sisään- ja ulostuloissaan samoja nimityksiä paitsi GET-lohkon NDR ulostulo, joka on toi-

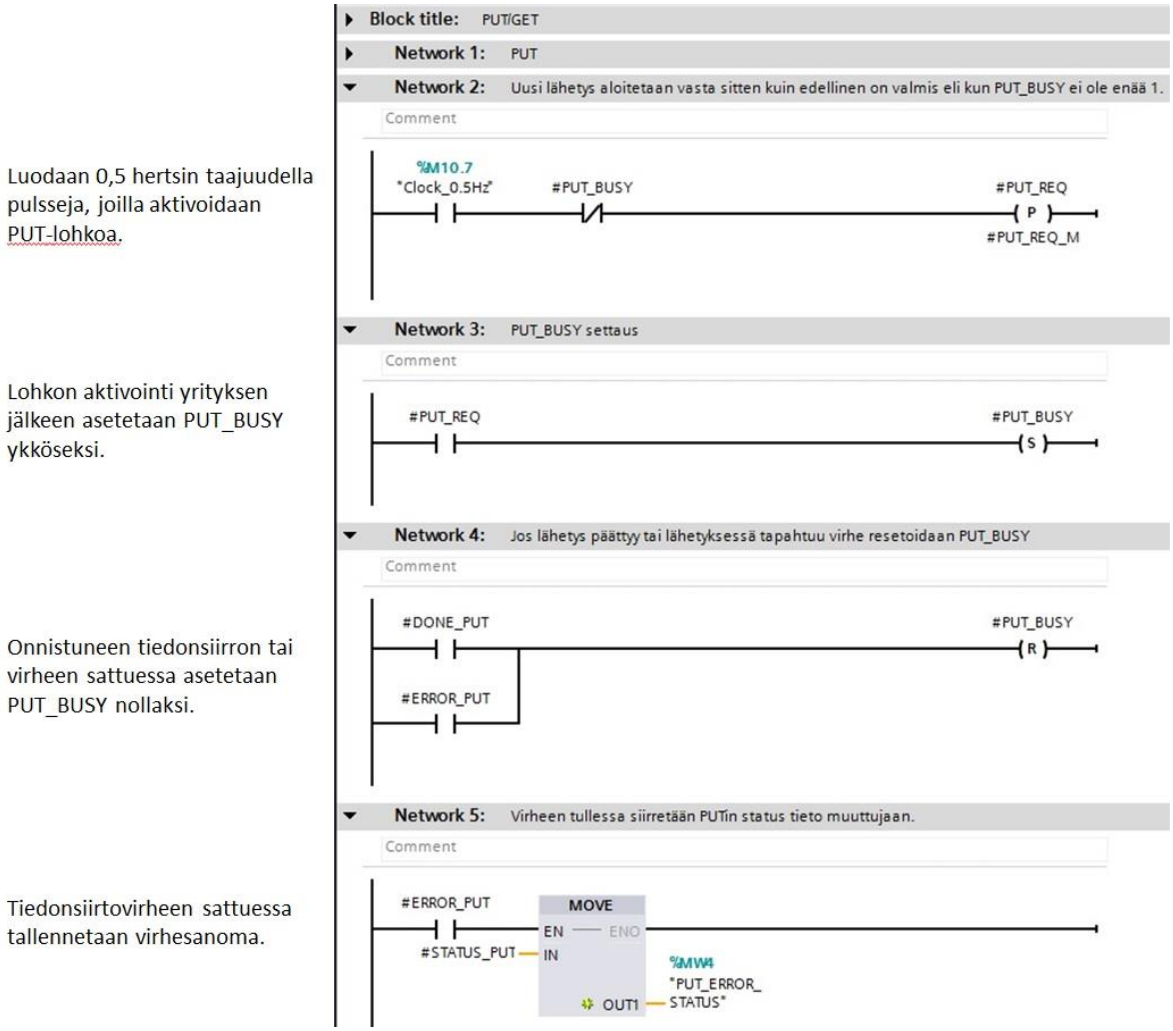
minnaltaan samanlainen kuin PUT-lohkon DONE ulostulo. PUT-lohko kirjoittaa tietoa toiselle logiikalle ja GET-lohko lukee tietoa toiselta logiikalta. Molemmille logiikoille täytyy luoda kaksi datalohkoa lähetettävän tiedon ja luetun tiedon tallentamista varten. Nämä datalohkot täytyy määritellä ei-optimoiduiksi koska PUT- ja GET-lohkot käsittelevät tietoa absoluuttisilla osoitteilla. Kuviossa 30 on esimerkkiohjelma, jossa on näkyvillä vain PUT- ja GET-lohkot ja projektipuu. PUT-lohkon ADDR_1 sisääntuloon kirjoitetaan toisella logiikalla olevan datalohkon osoite, johon tietoa halutaan kirjoittaa. SD_1 sisääntuloon kirjoitetaan taas sen datalohkon osoite, josta kirjoitettava tieto haetaan. REQ sisääntuloon tuomalla nousevan reunan aktivoidaan tiedonsiirto. Ulostulot antavat tietoa lohkon tilasta. DONE ulostulon tila muuttuu ykköseksi, kun tiedonsiirto on saatu virheittä suoritetuksi. ERROR ulostulo muuttuu ykköseksi, jos tiedonsiirrossa tapahtuu virhe. Virheestä saadaan tietoa STATUS ulostulosta. STATUS ulostuloon tulee virheen sattuessa numero, joka vastaa tiettyä virhettä.




KUVIO 30. PUT- ja GET-lohkot

GET-lohkon ulostuloilla on samat toiminnot mitä PUT-lohkossa. GET-lohkon ADDR_1 sisääntuloon kirjoitetaan toisella logiikalla olevan datalohkon osoite, josta tieto haetaan. SD_1 sisääntuloon kirjoitetaan taas datalohkon osoite, johon haettu tieto tallennetaan. NDR ulostulo GET-lohkossa vastaa DONE ulostuloa. STATUS ulostuloa voidaan käyttää diagnostiikkaan molemmissa lohkoissa. Molemmissa lohkoissa olevaan ID sisääntuloon kirjoitetaan sen logiikan ID-numero, jossa lohko on.

Kuviossa 31 on PUT-lohkon ohjaamiseen ja virhesanomien tallentamiseen käytetty koodi. REQ sisääntulon vaatima positiivinen reuna tehdään logiikan luomalla kellopulssilla. Positiivinen reuna tehdään vain silloin kun PUT-lohkoa ei ole pyydetty lähettämään tietoa. PUT_BUSY muuttuja kertoo, että onko tiedonsiirto käynnissä. ERROR ulostulolla tallennetaan virhesanomat siirtämällä lohkon STATUS ulostulossa oleva luku pysyvään muuttujaan. GET-lohkon ohjaaminen on tehty täysin samalla periaatteella.




KUVIO 31. Lohkojen ohjaaminen

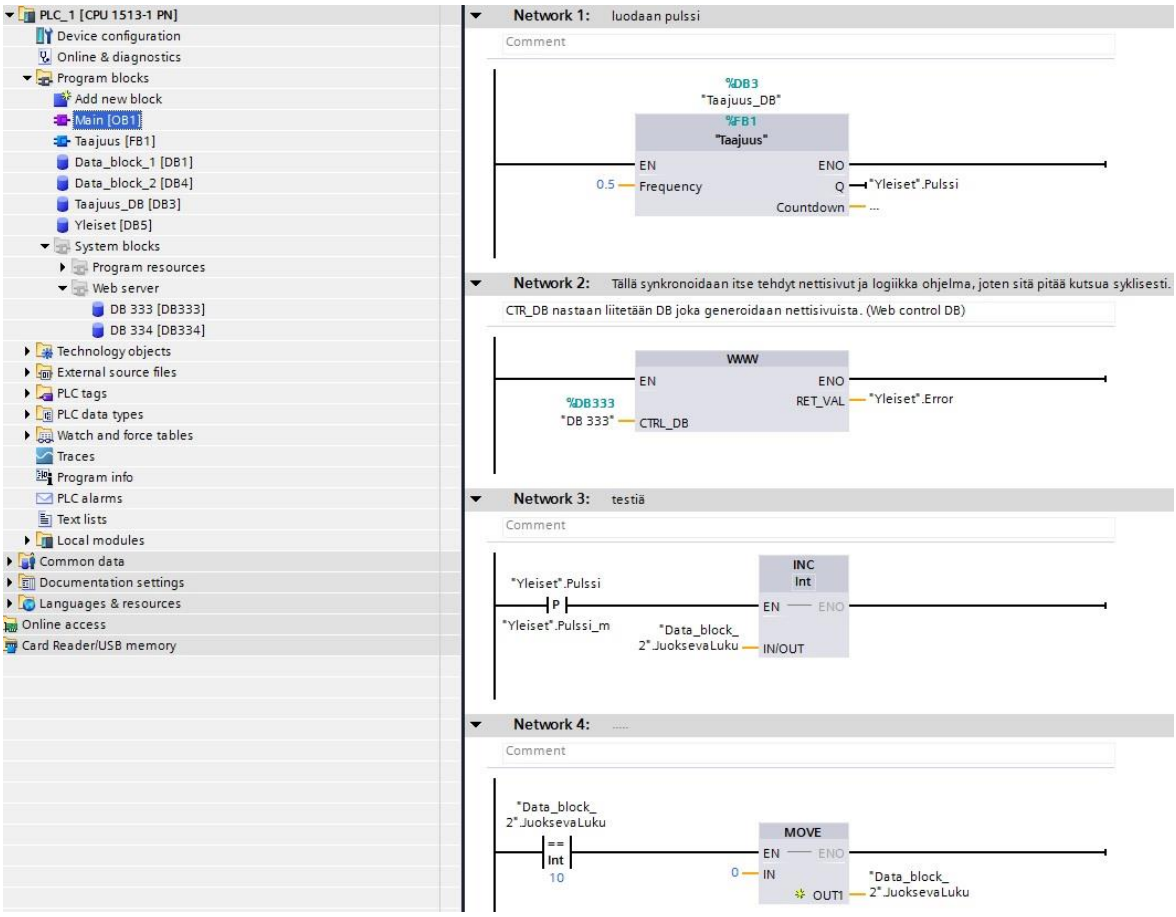
Tämän ohjelma oli ensimmäisiä, joita tein tätä opinnäytetyötä varten. Ohjelman tekeminen onnistui hyvin koska Siemensin sivuilta löytyi ohjeet ohjelman tekemiseen. Hieno asia oli myös, että tekemäni ohjelma tuli käyttöön erääseen projektiin. Asia oli myös uusi minulle. En ollut aikaisemmin tehnyt logiikoiden välistä kommunikointia.

### 6.3 HTML-sivujen käyttäminen valvomona

Tämä työ lähti liikkeelle siitä, että haluttiin tutkia, voiko logiikan verkkopalvelin ominaisuutta käyttää hyödyksi ja luoda pieni valvomo HTML-sivuna logiikkaan. Tiedon etsiminen lähti liikkeelle internetiä käyttäen. Siemensin sivuilta löysin PDF-dokumentin, jossa oli ohjeet verkkopalvelimen määrittelyyn ja omien HTML-sivujen tuomiseen logiikkaan. Seuraavaksi ryhdyin verestämään muistia HTML-sivujen tekemisestä ja käytin apuna koulussa harjoitustyönä tehtyjä HTML-sivuja. HTML-sivujen tekemiseen käytin Notepad++ -ohjelmaa. Minulle annettiin ohjeet, että valvomon tulisi olla hyvin yksinkertainen. Valvomo voisi sisältää esimerkiksi käynnistys ja pysäytys painikkeet ja indikointia laitteen toiminnasta. Suurin osa ajasta tätä ohjelmaa tehtäessä meni HTML-sivujen tekemiseen ja testaamiseen. Ohjelmassa ei ole muuta kuin yksi lohko HTML-sivujen ja logiikkaohjelman synkronointiin ja pari testausta varten tehtyä networkia, joilla luodaan juokseva numero. Logiikkana käytin tässä ohjelmassa 1513-1 PN -logiikkaa.

Ohjelma on hyvin yksinkertainen ja itse HTML-sivujen toimiminen ei vaadi muuta kuin WWW -lohkon kutsumisen. Kuviossa 32 on koko ohjelma ja datalohkot, joita on käytetty. DB 333 ja DB 334 datalohkot ovat HTML-sivuista luodut datalohkot. DB 333 on HTML-sivujen ohjaamiseen käytetty datalohko ja WWW -lohkon CTRL_DB sisääntuloon asetetaan tämä datalohko. Datalohkojen luominen on selitetty teoriaosuudessa luvussa 3.3. Networkit 3 ja 4 luovat juoksevan luvun. Juoksevaa lukua käytin testaamaan HTML-sivujen päivittymistä.

HTML-sivuissa käytin lomakkeita tiedon syöttämiseen ja taulukkoa luomaan rakennetta sivuun. Lomakkeisiin loin nappeja ja syöttökentän logiikan muuttujien muuttamiseen. Tyylitiedostoa käytin lisäämään väliä taulukon eri elementtien väliin ja muuttamaan tekstin väriä ja fonttia. Javascript-kieltä käytin muuttamaan tekstiä tekstikentässä riippuen muuttujan tilasta ja luomaan funktion sivun päivittämiseen.




KUVIO 32. HTML-sivujen synkronointi ja ohjelmakoodia

Kuviossa 33 on HTML-sivun otsikkoalue. Otsikkoalueessa olen määritellyt sivun otsikon ja tyylitiedoston, jota haluan käyttää tässä sivussa. Otsikkoalueelle olen luonut myös funktion, joka päivittää sivun. Funktiota kutsutaan otsikkoalueella itse javascript-koodissa kymmenen sekunnin välein ja myöhemmin kun päivitys-nappia klikataan. AWP-käskyt on kirjoitettu kommentteina HTML-sivuun. AWP-käskyt on käsitelty luvussa 3.3.

```
<!-- AWP_In_Variable Name='"Data_block_2"' -->
<html>
    <head>
        <title>Apex Automation Valvomo</title> <!-- HMTL-sivun otsikko, joka näkyy selaimessa-->
        <meta http-equiv="content-type" content="text/html; charset utf-8">
        <link rel= "stylesheet" type="text/css" href="tyylit.css"> <!-- Tyyylitiedoston lisääminen -->
        <script language="javascript"> <!-- javascrip koodi sivun päivittämiseen 10 sekunnin välein -->

        function päivitä()
        {
            window.location = window.location;
        }

        setInterval("päivitä()",10000)

        </script>
```

KUVIO 33. HTML-sivun otsikkoalue

Tekstikentät ja napit loin yhteen isoon taulukkoon, jotta valvomo näyttäisi siistiltä. Taulukolla on helppo luoda selkeä rakenne HTML-sivulle. Yhdelle riville loin esimerkiksi muuttujan arvon näyttämisen yhteen sarakkeeseen ja seuraavaan sarakkeeseen syöttökentän ja napin arvon siirtämiseen logiikkaan. Muuttujan arvon muuttamiseen käytin lomaketta. Lomakkeessa on nappi arvon muuttamiseen ja syöttökenttä uuden arvon kirjoittamiseen. (KUVIO 34).

```
<center><table border=1>
    <tr class = spaceUnder>
    <td>
    <!--AWP_In_Variable Name='"Data_block_2".JuoksevaLuku' --> <!-- AWP-käsky kertomaan mistä muuttuja löytyy -->
    Muuttuva arvo: :="Data_block_2".JuoksevaLuku:
    </td>
    </tr>
    <tr class = spaceUnder>
        <td>
        Muutettava arvo: :="Data_block_2".Luku:
        </td>


        <!--AWP_In_Variable Name='"Data_block_2".Luku' --> <!-- AWP-käsky kertomaan mistä muuttuja löytyy -->
        <td>
        <form method="post">
            Uusi arvo: <input name='"Data_block_2".Luku' type="text">
            <input type="submit" value="Lataa">
        </form>
        </td>
    </tr>
```

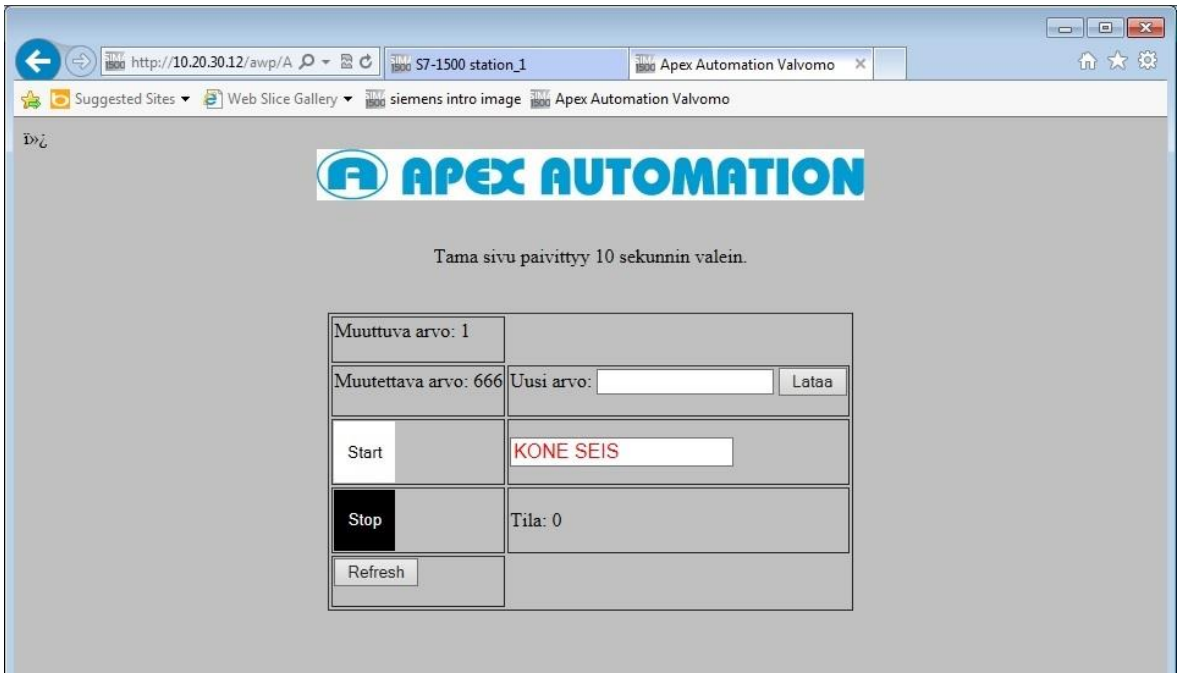
KUVIO 34. Lomake tiedon lähettämiseen

Lomaketta käytin uudestaan muuttamaan toista muuttujaa ja näyttämään tekstiä tekstikentässä. Javascript-kielellä kirjoitettu koodi vaihtaa tekstikentässä näkyvää tekstiä riippuen mikä on StartStop-muuttujan tila. Start-napilla muuttuja muutetaan ykköseksi eli totuusarvoksi tosi. Stop-napilla muuttuja muutetaan nollaksi eli epätodeksi (KUVIO 35).

```
    <form method="post" action="">
        <!--AWP_In_Variable Name='"Data_block_2".StartStop' --> <!-- AWP-käsky kertomaan mistä muuttuja löytyy -->
        <input type="submit" value="Start" style="background-color: white; border: 1; width: 50px; height: 50px;">
        <input type="hidden" name='"Data_block_2".StartStop' value="1"> <!-- Start-napilla muutetaan StartStop ykköseksi -->
    </form>
</td>
<td>
    <input type="text" value="" id="start" class="intext"> <!-- Tekstikenttä, jossa lukee koneen tila -->
    <script language="javascript">
    /*AWP_In_Variable Name='"Data_block_2"'*/ /* AWP-käsky kertomaan mistä muuttuja löytyy */
    /*AWP_In_Variable Name='"Data_block_2".StartStop'*/ /* AWP-käsky kertomaan mistä muuttuja löytyy */
    var s =  :="Data_block_2".StartStop: ; /*Siirretään StartStop tieto muuttujaan s*/
    if(s == 1)
        {
            /*getElementById syöttää "KONE KÄYNNISSÄ" tekstin tekstikenttään, jonka id on "start" */
            document.getElementById("start").value = "KONE KÄYNNISSÄ"
        }
    else
        {
            /*"KONE SEIS" teksti syötetään, jos StartStop on jotain muuta kuin yksi*/
            document.getElementById("start").value = "KONE SEIS"
        }
    </script>
```

KUVIO 35. Lomake ja javascript-koodi

Kuviossa 36 on valmis HTML-sivu. Sivu päivittyy kymmenen sekunnin välein tai kun Refresh-nappia klikataan. Muuttuva arvo päivittyy niissä tapauksissa. Uuden arvon lataaminen onnistuu kirjoittamalla haluttu luku syöttökenttään ja painamalla enteriä tai klikkaamalla Lataa-nappia. Uuden arvon lataaminen myös päivittää sivun. Start- ja stop-painikkeilla muutetaan tekstikentässä näkyvää tekstiä vastaamaan StartStop-muuttujan tila. Muuttujan tila on vielä lukuarvona stop-napin oikealla puolella.




KUVIO 36. Valmis HTML-sivu

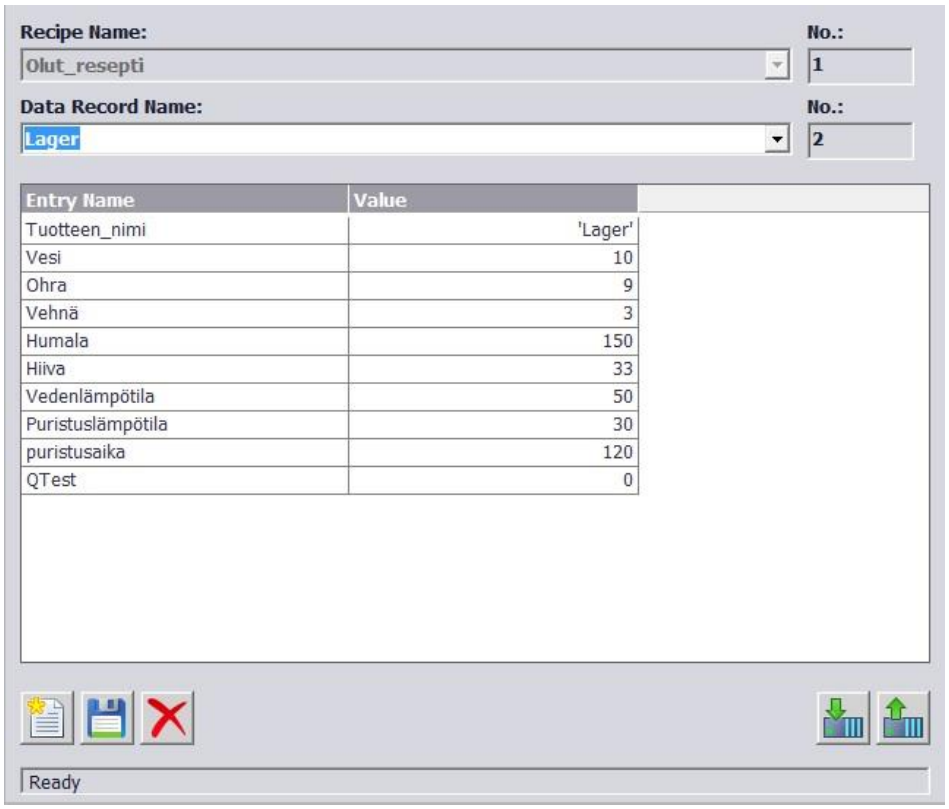
Tässä ohjelmassa en joutunut paljon tekemään ohjelmaa TIA Portalilla vaan kaikki aika meni HTML-sivujen tekemiseen. Olin aikaisemmin tehnyt pienet kotisivut ja harjoitellut javascript-kieltä, mutta kaikki piti opetella uudestaan. Javascriptin kirjoittamisessa oli ongelmana se, että ohjelma, jolla tein sivuja, ei kertonut kirjoitusvirheistä ollenkaan. Paljon aikaa kului siihen, että etsin virheitä koodista. Koodi saattoi olla ajatuksena oikein, mutta jostain sanasta saattoi puuttua kirjain ja silloin koodia ei suoritettukaan oikein.

### 6.4 Uusien reseptien tuominen paneeliin

Reseptejä käytetään, jos halutaan esimerkiksi asetella tietyt arvot sopiviksi jotain tiettyä tuotetta varten. Reseptit luodaan TIA Portal -ohjelmassa yleensä etukäteen. Uusin reseptien

tuominen HMI-paneeliin esimerkiksi csv-tiedostona oli yksi asia, jota ei ollut kokeiltu aiemmin ja minua pyydettiin tutkimaan asiaa. Aloitin tutkimisen TIA Portalissa katsomalla mitä mahdollisuuksia reseptien hallitsemiseen siellä on valmiina. Recipe view niminen ikkuna, jonka voi lisätä paneelin ruutuun, oli ainoa tähän tarkoitukseen sopiva asia jonka löysin. Paneelia minulla ei ollut käytössä ohjelmaa tehtäessä, mutta pystyin käyttämään WinCC Runtime Advanced -ohjelmaa. WinCC:tä pystyin ajamaan ohjelmointikoneelta ja tällä tavoin testaamaan ohjelmaa.

Recipe view -ikkunan toimintaan tutustuakseni luin TIA Portalin manuaalia ja testasin omaa reseptiä. Reseptin rakenteesta löytyi esimerkki TIA Portalin manuaalista, ja sitä käyttäen loin kolme eri reseptiä. Recipe view -ikkunassa on valmiiksi monta toimintoa. Sillä pystytään tallentamaan valitun reseptin tiedot logiikkaan ja tuomaan ne sieltä. Sillä pystytään myös lisäämään ja poistamaan reseptejä. Kuviossa 37 on Recipe view -ikkuna ja valittu resepti.

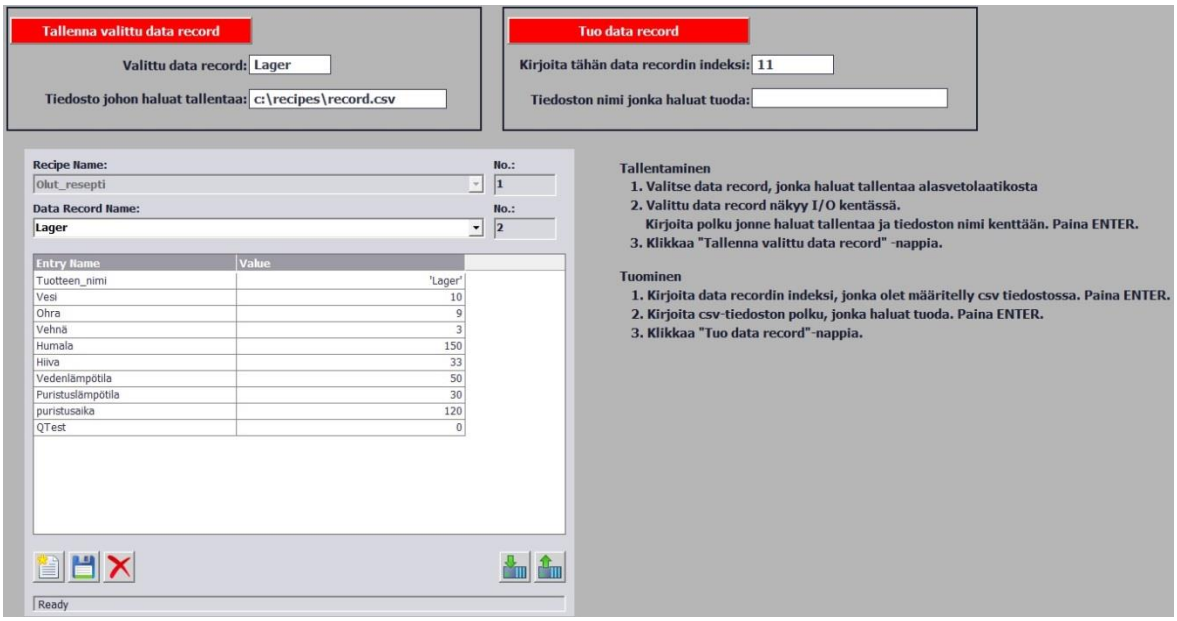



KUVIO 37. Recipe view -ikkuna

Paneeleihin voidaan luoda painikkeita ja näille erilaisia toimintoja. Näitä toimintoja tutkiessa törmäsin kahteen toimintoon, joilla pystyi tallentamaan reseptejä csv-tiedostona tai

tuomaan csv-tiedostoja paneelin muistikortilta. ExportDataRecords nimisellä toiminnolla voidaan recipe view -ikkunassa valittu resepti tallentaa paneelin muistikortille. Recipe view:n asetuksiin pitää vain lisätä tagi, johon valitun reseptin data recordin nimi tallentuu. Tämä tagi laitetaan ExportDataRecords toiminnon asetuksissa kohtaan, johon data recordin nimi kirjoitetaan. Vastaavasti ImportDataRecords toiminnolla voidaan tuoda muistikortilta resepti paneeliin. ImportDataRecords tarvitsee uuden reseptin data recordin indeksin ja polun tiedostoon, joka halutaan siirtää paneeliin.

Kuviossa 38 on paneelisovellus, jonka loin käyttäen ExportDataRecords ja ImportDataRecords toimintoja. Vasemmanpuoleisella napilla voidaan tallentaa recipe view -ikkunassa valittu data record paneelin muistikorttiin. Tiedostopolku, johon tiedot halutaan kirjoittaa, kirjoitetaan tekstikenttään. Oikeinpuoleisella napilla tuodaan data record paneelin muistikortilta. Data recordin indeksi kirjoitetaan ylempään tekstikenttään ja alempaan kirjoitetaan tiedostopolku tiedostolle, joka halutaan tuoda paneeliin.




KUVIO 38. Paneelisovellus reseptien käsittelyyn

## 6.5 Data log-tiedostot

Tämän työn tarkoituksena oli selvittää miten tietoa ohjelmasta voitaisiin tallentaa csv-tiedostoon logiikan muistikortille. Csv-tiedostoa voi esimerkiksi katsoa Excel-ohjelmalla ja

tutkia ohjelmasta saatuja tietoja. Tähän tarkoitukseen TIA Portalissa käytetään data log-tiedostoja. Ryhdyin etsimään tietoa data log-tiedostoista internetistä ja löysin Siemensin luoman ohjeen miten data log-tiedostoja voidaan luoda. Ohjeessa oli esimerkkejä siitä miten eri lohkoja käytetään. Tämän ohjeen pohjalta loin esimerkkiohjelman, jolla voidaan luoda uusia tiedostoja, kopioida tiedoston rakenteen, kirjoittaa tietoa tiedostoon ja poistaa olemassa olevia tiedostoja. Ohjelmassa käytin DataLogCreate-, DataLogWrite-, DataLogClose-, DataLogOpen-, DataLogNewFile- ja DataLogDelete-lohkoja. Muistikortilla olevia tiedostoja voi ladata omalle koneella käyttäen logiikan verkkopalvelin ominaisuutta tai asettamalla logiikan muistikortin tietokoneen muistikortinlukijaan.

Ohjelman alussa luodaan uusi tiedosto DataLogCreate-lohkolla. Lohkolla määritellään uuden tiedoston rakenne, tiedostopääte, sarakkeiden otsikot ja mistä tieto tuodaan tiedostoon. Jokaisella tiedostolla on myös oma ID-numeronsa, josta ne tunnistetaan. Loin kuviossa 39 näkyvät muuttujat ohjelmaan. MyData on struct-tietotyyppiä oleva muuttuja ja sen eri elementteihin syötetään tiedostoon tallennettavat tiedot. MyDataLogId-muuttujaa käytin eri tiedostojen ID-numeroiden tallentamiseen. MyDataLogHeaders-muuttujaan tallensin sarakkeiden otsikot. Kahta ylintä muuttujaa käytin nimeämään tiedostoja.

My_Datalog_Vars

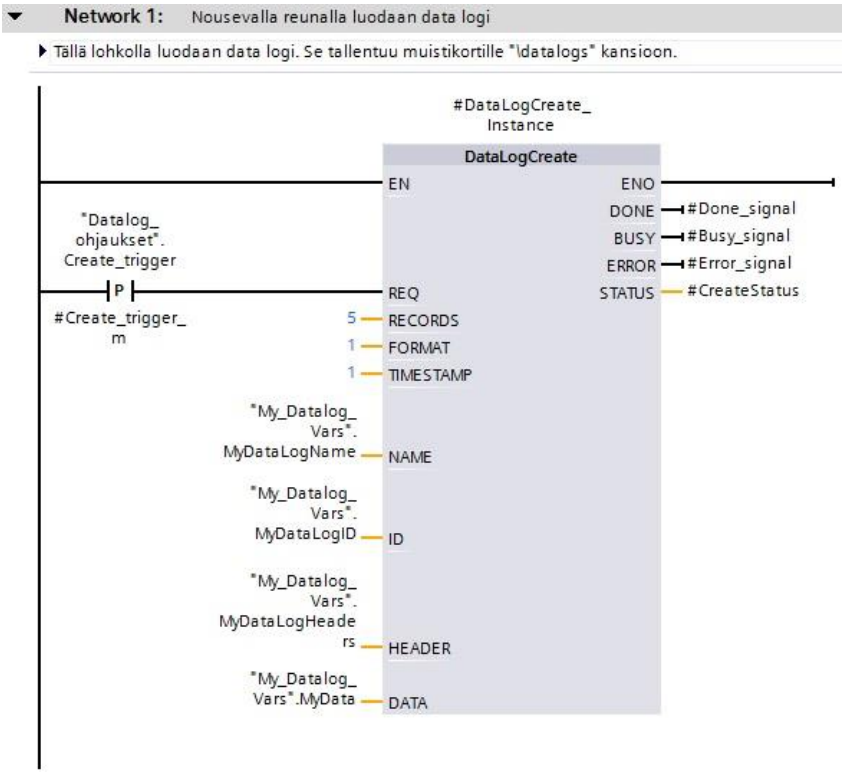
| | Name | Data type | Start value |
|---|---|---|---|
| 1 | Static | | |
| 2 | MyNEWDataLogName | String | 'MyNewDataLog' |
| 3 | MyDataLogName | String | 'MyDataLog' |
| 4 | MyDataLogID | DWord | 16#0 |
| 5 | MyDataLogHeaders | String | 'Count;Temperature;Pressure' |
| 6 | MyData | Struct | |
| 7 | MyCount | Int | 0 |
| 8 | MyTemperature | Real | 0.0 |
| 9 | MyPressure | Real | 0.0 |

KUVIO 39. Muuttujat data log-tiedostojen käsittelyyn

Kuviossa 40 on esimerkkiohjelman ensimmäinen network, jossa luodaan uusi tiedosto. RECORDS, FORMAT ja TIMESTAP sisääntuloilla määritellään tiedoston rakenne. RECORDS sisääntulolla kerrotaan mikä on maksimimäärä rivejä tietoa, jota tiedostoon voidaan kirjoittaa. Tiedoston ollessa täynnä tietoa seuraava kirjoituskäsky kirjoittaa vanhimman tiedon yli. FORMAT sisääntulolla kerrotaan tiedostopääte. Numero yksi tarkoittaa, että halutaan luoda csv-tiedosto. TIMESTAMP sisääntulolla kerrotaan halutaanko luoda aikaleima sarake automaattisesti tallennettujen tietojen eteen. Aikaleimassa näkyy kellon-

aika ja päivämäärä. ID-sisääntuloa on kaikissa lohkoissa In/Out tyyppinen, eli siitä luetaan ja siihen kirjoitetaan tietoa. Esimerkiksi tiedoston avaamisen jälkeen, avatun tiedoston ID-numero kirjoitetaan ID:hen. Tämän jälkeen voidaan kyseinen tiedosto sulkea käyttäen tätä ID-numeroa.

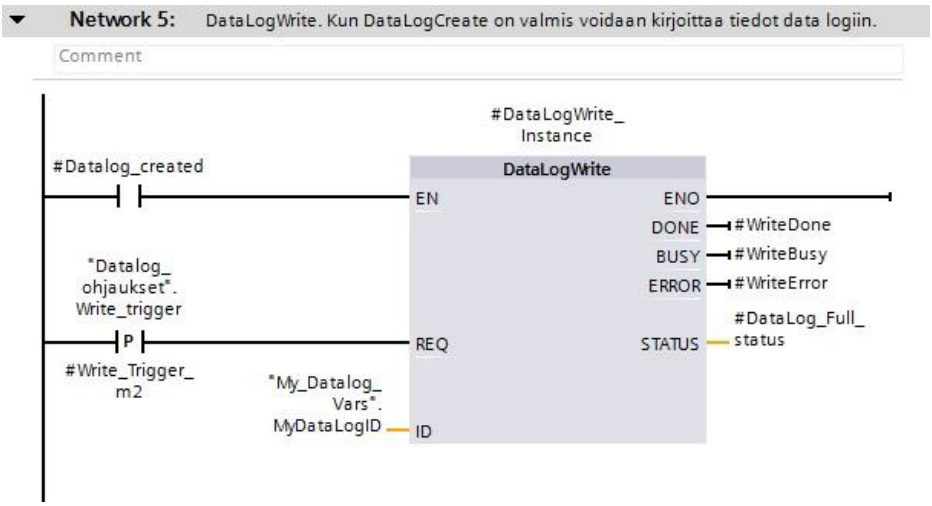
Lohkon ulostuloja käytin tallentamaan virheviestejä ja kertomaan ohjelmalle milloin uuden tiedoston luominen on valmis. Lohkon ilmoittaessa virheestä ohjelma tallentaa STATUS tiedon muistiin. Tämä oli todella kätevää ohjelman testauksen yhteydessä. Virhesanoma yleensä auttaa kertomaan hyvin yksityiskohtaisesti mikä on vialla. Virhesanoma on heksadesimaali muodossa ja se täytyy tulkita TIA Portalin ohjeesta löytyvältä listalta.




KUVIO 40. DataLogCreate-lohko

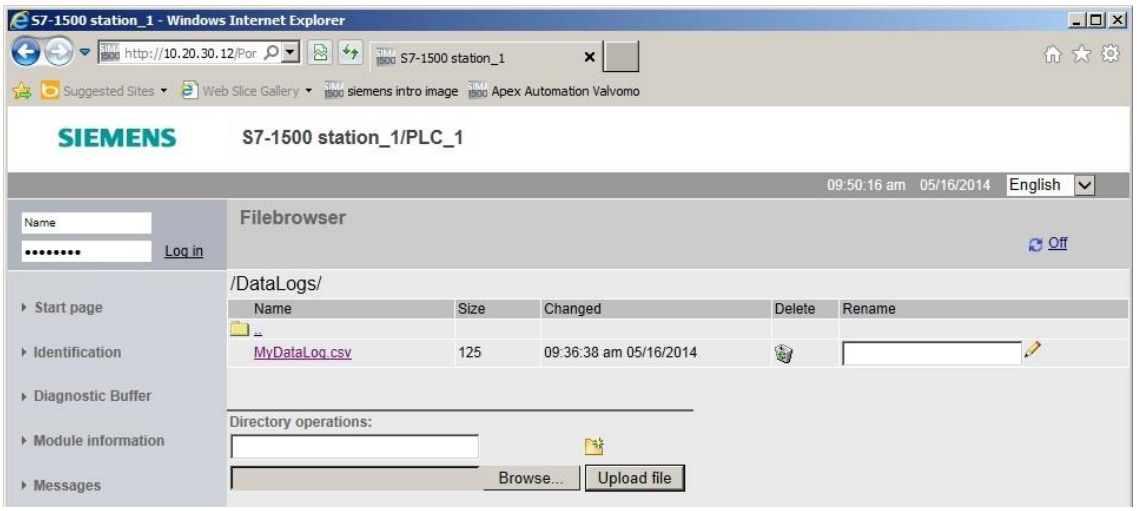
Data log-tiedostoja käsiteltäessä eräs tärkeimmistä asioista on pitää huoli, että tietää mitä eri lohkot tekevät. Tietyt lohkot avaavat muistikortilla olevia tiedostoja ja jos niitä ei erikseen käske sulkea, ne pysyvät auki ja aiheuttavat ongelmia myöhemmin. Avoinna olevaa tiedostoa ei mistään voi tarkistaa, joten tiedostojen käsittely voi olla hankalaa. Huomasin ohjelmaa tehdessä, että aina kun suorittaa tehtävän, joka avaa tiedoston, on järkevää suorittaa DataLogClose-lohko heti sen jälkeen.

Uuden tiedoston luomisen jälkeen ohjelmassa kirjoitetaan halutut tiedot tiedostoon. DataLogWrite-lohkolle kerrotaan vain tiedoston ID-numero, johon halutaan kirjoittaa tietoa. Tämä ID-numero on valmiiksi jo tiedossa koska aikaisemmin kun tiedosto luotiin ID-numero tallennettiin muuttujaan. Tiedostoa luodessa on jo määritelty mistä tiedoston tiedot tulevat, joten sitä ei tarvitse määritellä uudestaan tässä. Kuviossa 41 on network, jossa kirjoitetaan tietoa tiedostoon. Tiedoston kirjoittaminen avaa aina kyseisen tiedoston, joten käytin kirjoituslohkon Done-ulostuloa sulkemaan tiedoston automaattisesti kirjoittamisen jälkeen. Tiedoston sulkeminen tehdään DataLogClose-lohkolla ja tiedosto, joka halutaan sulkea, tunnistetaan sen ID-numerosta.



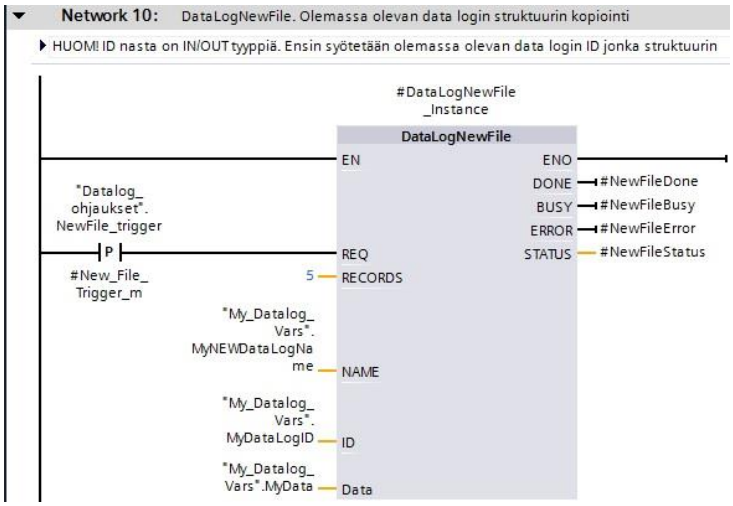

KUVIO 41. DataLogWrite-lohko

Tässä vaiheessa ohjelmaa on luotu yksi uusi tiedosto ja siihen on kirjoitettu tietoa. Tiedoston voi tässä vaiheessa käydä lataamassa omalle tietokoneelle käyttäen internetselainta, jos logiikan verkkopalvelin ominaisuus on laitettu päälle. Selaimessa on myös mahdollista poistaa ja uudelleen nimetä tiedostoja. Poistamista ei saa kuitenkaan tehdä tätä kautta koska tiedosto ei poistu kokonaan. Uutta tiedostoa luodessa, jolla on sama nimi kuin selaimen kautta poistetulla tiedostolla, antaa DataLogCreate-lohko virheilmoituksen. Virheilmoitus kertoo, että tiedosto on jo olemassa vaikka sellaista ei muistikortilta löydy mistään. Tällaiset haamutiedosto saa poistettua vain formatoimalla logiikan muistikortin. Kuviossa 42 on logiikan verkkosivut ja tiedostojen selaamiseen käytetty näkymä. Selaimessa näkyy tiedoston nimi ja koko.



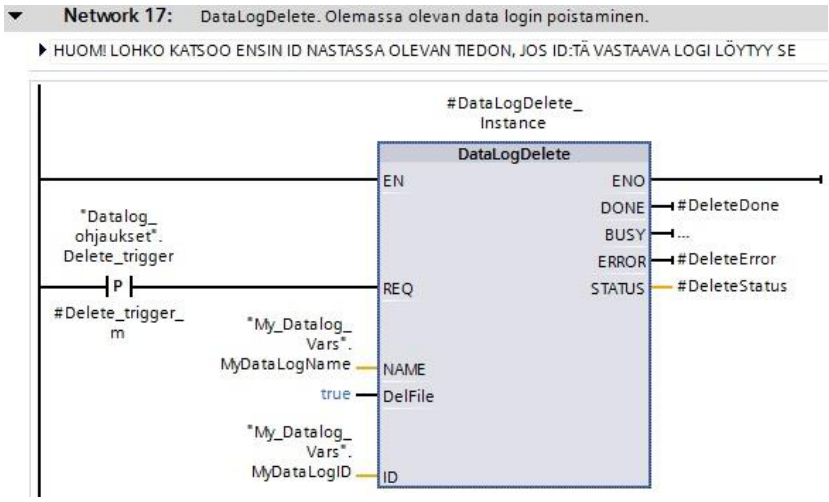

KUVIO 42. Tiedostojen selaus selaimella

Tiedostojen rakenteen kopioimiseen on myös oma lohkonsa. DataLogNewFile-lohkolla voidaan kopioida olemassa olevan tiedoston rakenne ja luoda uusi tiedosto, joka käyttää tätä rakennetta. Tämä siis kopioi ainoastaan tiedoston rakenteen eli ei siihen tallennettuja tietoja. Tiedosto, joka halutaan kopioida, täytyy ensin avata DataLogOpen-lohkolla. DataLogOpen kirjoittaa MyDataLogId-muuttujaan tiedoston ID-numeron. ID-numeroa käytetään tiedoston kopioinnissa tunnistamaan tiedosto, joka halutaan kopioida. Uudelle tiedostolle pitää antaa vain nimi, tiedon maksimimäärä ja mistä tieto tallennetaan. Kopioinnin jälkeen lohko tallentaa uuden tiedoston ID-numeron muuttujaa. ID-numeroa voidaan käyttää taas sulkemaan tämä uusi tiedosto, joka luotiin. Tässä täytyy ottaa huomioon vielä tiedosto, joka haluttiin kopioida. Tämä tiedosto avattiin aikaisemmin ja sitä ei ole vielä suljettu. Tämän tiedoston ID-numero täytyy ottaa selville uudestaan käyttämällä avauslohkoa ja sen jälkeen sulkea se käyttäen sen ID-numeroa. Kuviossa 43 on network, jossa luodaan uusi tiedosto, joka käyttää vanhan tiedoston rakennetta.




KUVIO 43. DataLogNewFile-lohko

Aikaisemmin kerroin, että tiedostoja ei saa poistaa selaimessa. Tiedostojen poistaminen täytyy tehdä ohjelmassa DataLogDelete-lohkolla. Lohkolle pitää kertoa poistettavan tiedoston nimi ja ID-numero. Lohko ottaa huomioon vain ID-numeron, jos se löytää sitä vastaavan tiedoston. Nimen perusteella se poistaa tiedoston vain, jos ID-numeroa vastaavaa tiedostoa ei löydy. Kuviossa 44 on network, jossa poistetaan tiedosto muistikortilta.




KUVIO 44. DataLogDelete-lohko

Data log-tiedostojen käsittely on tehty melko vaikeaksi. Tiedostojen luominen ja poistaminen voi luoda ongelmia, joita ei heti tajuaisi. Siemensin ohjeissa oli mainittu vain yhdellä lauseella, että tiedostoja ei saa poistaa selaimen kautta. Ehdin poistaa tiedostoja selaimessa ennen kuin luin tämän tiedon ja muistikortti piti formatoida. Lohkot myös tekevät asioita, joita ei odottaisi ja TIA Portalin sisältämät ohjeet niiden käytöstä pitää lukea tarkkaan.

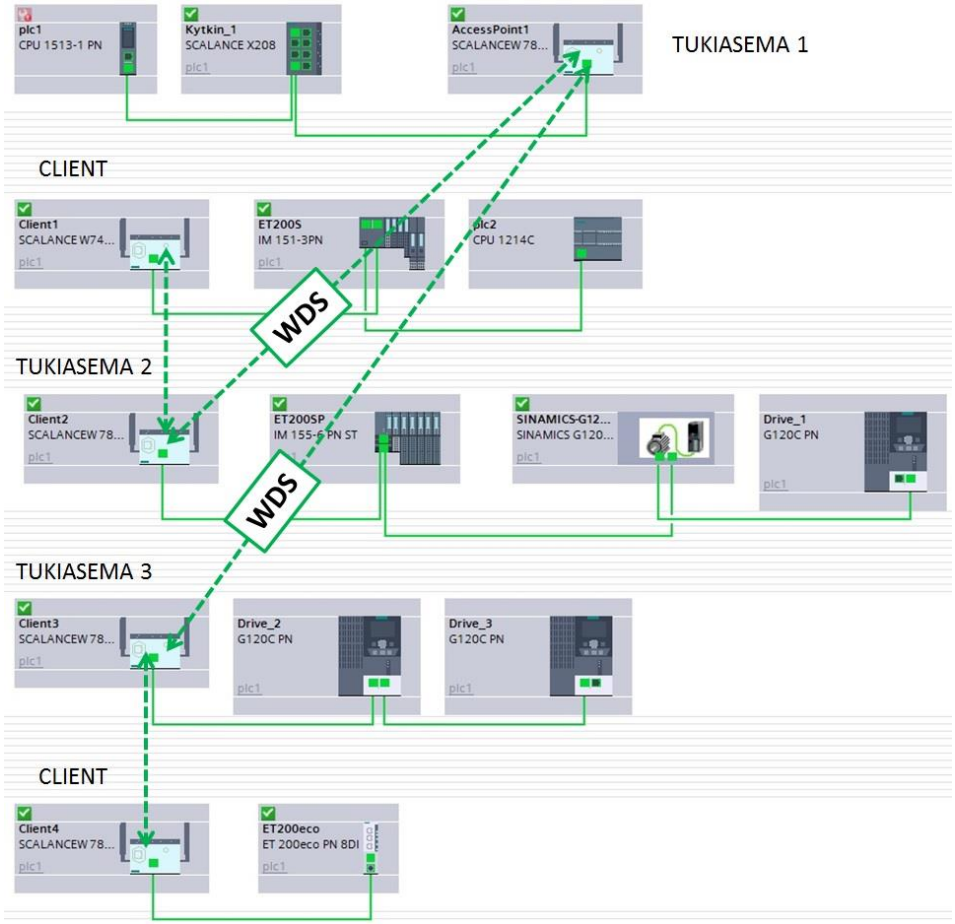
Tiedostojen käsittely oli tehty hankalaksi koska auki olevista tiedostoista ei ollut mitään tietoa.

### 6.6 IWLAN tukiasemien ja client-moduulien määrittely WDS-verkoksi

Eräässä projektissa oli törmätty ongelmaan WLAN-yhteyksien kanssa. Minulle annettiin tehtäväksi rakentaa projektin laajuutta vastaava testilaitteistoa ja etsiä ongelmaan ratkaisu. Ongelmana oli, että kaikki laitteet eivät saaneet yhteyttä toisiinsa kun laitteita oli kytkettynä paljon WLAN-laitteiden taakse. WDS-verkon määrittely haluttiin myös testata samalla. Projektissa oli käytössä myös PUT ja GET ohjelma, jonka olin tehnyt aikaisemmin.

Aluksi käytössä oli vain yksi tukiasema (access point) ja yksi client-moduuli. Lisäksi näihin oli kytketty 1500 -logiikka, 1200 -logiikka sekä IO-yksikkö. Myöhemmin kokoonpanoa laajennettiin kolmella tukiasemalla, neljällä taajuusmuuttajalla ja kahdella IO-yksiköllä. Lopulta laitteita oli 14 ja yksi kytkin, johon ohjelmointikoneita voitiin liittää. Testilaitteistossa käytettiin G120 taajuusmuuttajia sekä ET200S, ET200SP ja ET200eco IO-yksiköitä. WLAN:n rakentamiseen käytettiin Siemensin valmistamia Scalance W788-1 PRO tukiasemia ja W746-1 PRO client-moduulia. Tukiasemat voidaan määritellä client-moduuleiksi, joten yhtä tukiasemaa käytettiin clienttina (KUVIO 45).

Tukiasemien välille haluttiin luoda WDS-yhteys ja tukiasemien ja clientien välille luoda normaalit WLAN-verkot. Tavoitteena oli, että yksi tukiasema on yhteyksissä yhteen clientiin ja nämä muodostavat yhden verkon ja toinen tukiasema muodostaa toisen verkon yhden clientin kanssa. Tällä tavoin tiedonsiirtoreitti on aina vakio ja tukiaseman vikaantuessa vain yhden clientin liikenne lakkaa. Kuviossa 45 on testilaitteisto ja verkon toimintaperiaate esitettynä. Testilaitteisto todellisuudessa on kuviossa 46. Yksi client-moduuli ei mahtunut kuvaan.



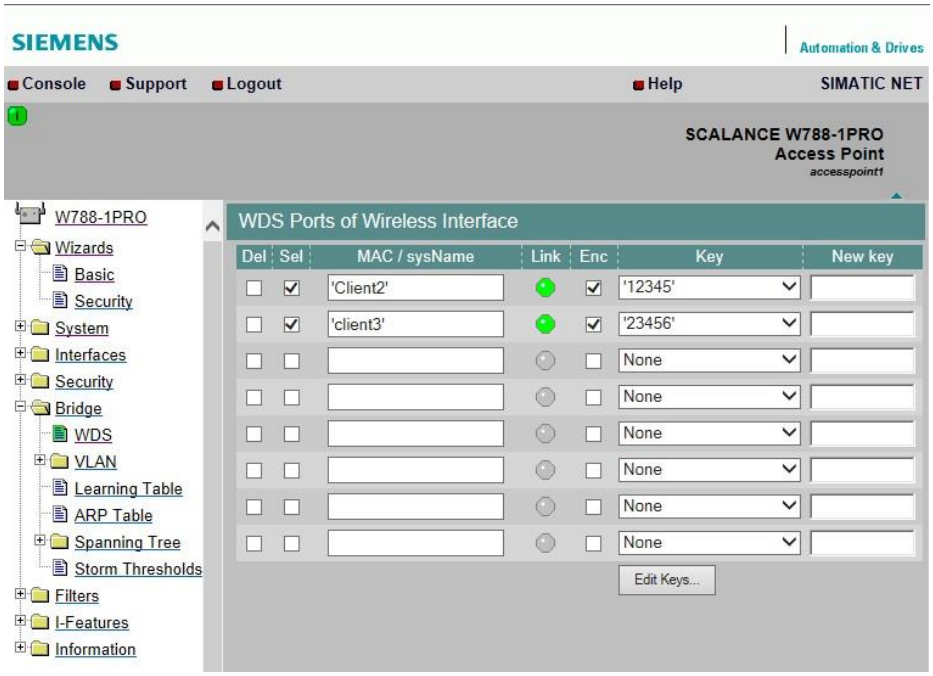

KUVIO 45. IWLAN testilaitteisto TIA Portalissa

KUVIO 46. Testilaitteisto todellisuudessa

Ongelmia syntyi jo ennen kuin itse WLAN:ia päästiin testaamaan. Taajuusmuuttajille asetetut Profinet nimet eivät jostain syystä pysyneet taajuusmuuttajan muistissa vaan taajuusmuuttaja aina vaihtoi itsestään vanhan nimen uuden tilalle. Taajuusmuuttajien parametrointia tehtiin siinä vaiheessa kytkimien ja WLAN:n kautta käyttäen TIA Portalin StartDrive -ohjelmistoa. Oikeat nimet saatiin laitettua ensin palauttamalla taajuusmuuttaja tehdasasetuksille ja sen jälkeen kytkemällä ohjelmointikone suoraan Ethernet-kaapelilla taajuusmuuttajaan ja parametroimalla se käyttäen Starter -ohjelmistoa.

Saatuamme työkaverin kanssa IP-osoitteet ja nimet kohdalleen ryhdyimme määrittelemään WLAN-laitteita. Tukiasemille ja clienteille asetimme nimet ja IP-osoitteet TIA Portalin kautta mutta itse määrittely tehtiin nettiselaimessa. Määrittelyssä asetetaan laitteelle nimi, josta se tunnistetaan. Tätä nimeä käytetään WDS-yhteyden luomiseen tukiasemien välillä. Laitteelle määritellään myös SSID-tunnus. SSID-tunnuksella määritellään, missä verkossa laite toimii. Asetimme tukiasemalle 2 ja tukiasemalle 3 eri SSID-tunnukset ja niiden clienteille näitä vastaavat tunnukset. Laiteille pitää myös määritellä radiotaajuus, jolla ne toimivat, ja kanava jota ne käyttävät valitulla taajuudella. Tukiasemien ja clientien määrittely on melkein samanlainen prosessi mutta clientin MAC-mode täytyy asettaa Layer 2 Tunnel moodiin. Tätä asetusta pitää käyttää koska muuten clientiin kytketyt laitteet eivät saa yhteyttä WLAN-verkkoon.

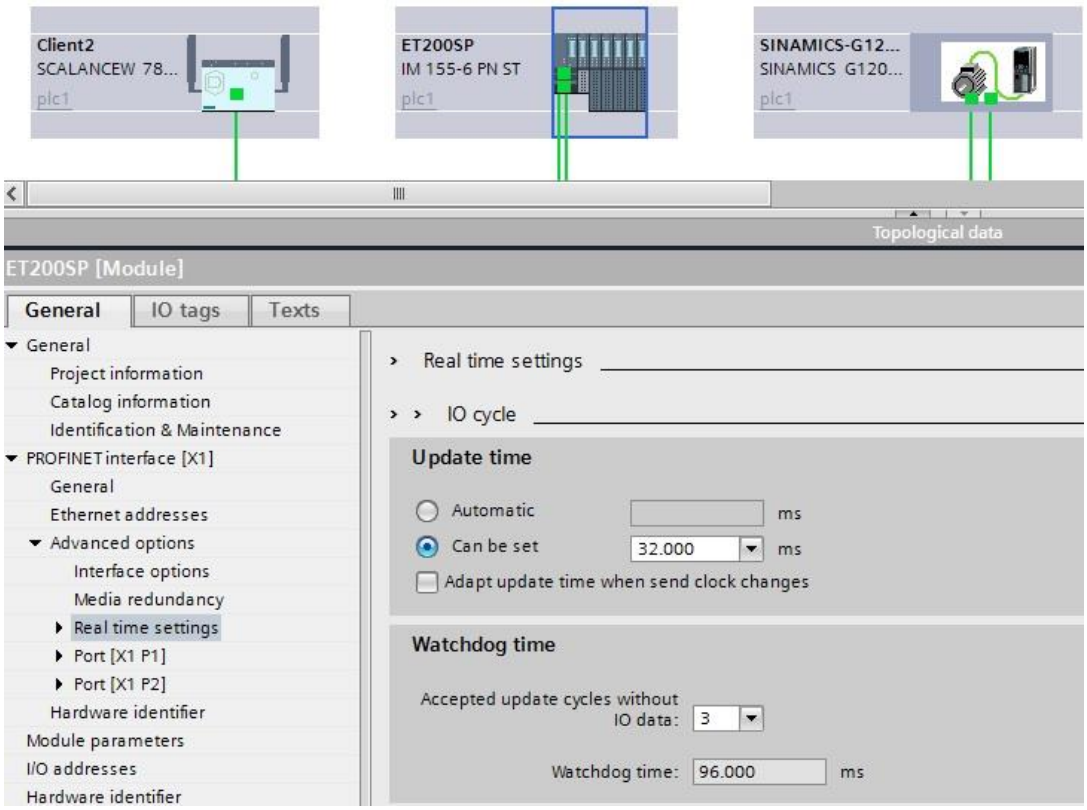
WDS-yhteys pitää määritellä kaikille tukiasemille. Kuviossa 47 on WDS-asetukset tukiasemalle 1, joka on yhteydessä 1500 -logiikkaan. SysName-kenttään syötetään nimi, joka tukiasemalle määriteltiin. Tämä nimi täytyy olla täysin samassa muodossa kuin se syötettiin. Key-kenttään voidaan syöttää yksinkertainen salausavain, jos verkkoa halutaan suojata. Vastaavasti Client2 ja client3 nimisille tukiasemille täytyy määritellä WDS-yhteys tähän tukiasemaan. Tukiaseman 2 WDS-asetuksissa näkyy vain tukiaseman 1 nimi ja käytetty salausavain. WDS-verkosta ja IWLANista on kerrottu tarkemmin luvussa 4.4.




KUVIO 47. Tukiaseman 1 WDS-asetukset

Saatuamme työkaverin kanssa WDS-yhteydet määriteltyä ryhdyimme selvittämään yhteysongelmaa. Huomasimme jo kolme tai neljä laitetta kytkettyämme WLAN-laitteisiin, että kaikki laitteet eivät saaneet Profinet-verkkoon yhteyttä. Osassa laitteista paloivat BF-valot punaisina, jotka merkkaavat väylävikaa. Välillä laite, joka oli aikaisemmin ilmoittanut väyläviasta, satunnaisesti sai yhteyden ja menetti sen heti. Tietokoneen komentokehoteikkunassa käyttämällä ping-käskyä, joka lähettää IP-paketteja laitteille ja mittaa aikaa, joka kestää, että kyseinen laite vastaa, testasimme laitteita. Laitteet satunnaisesti lähettivät vastauksen ja paketteja katosi matkalla.

Yksi idea, jota kokeilimme, oli watchdog-ajan pidentäminen. Watchdog-aika määrittelee, että kuinka kauan IO-laite saa olla ilman IO-dataa ennen kuin se ilmoittaa viasta. Eri watchdog-aikoja kokeiltuamme huomasimme, että sillä ei ollut suurta vaikutusta tilanteeseen. Seuraavaksi kokeilimme pidentää IO-laitteen päivitysaikaa. Tällä ajalla kerrotaan aikaväli, jolla IO-ohjain saa tietoa IO-laitteelta ja lähettää sille tietoa. Käytännössä tämä tarkoittaa sisään- ja ulostulojen päivitystä. Tästä prosessista on lisää tietoa Profinet IO luvussa tietoperustassa. Kuviossa 48 on IO-laitteen päivitysajan asetukset TIA Portalissa.



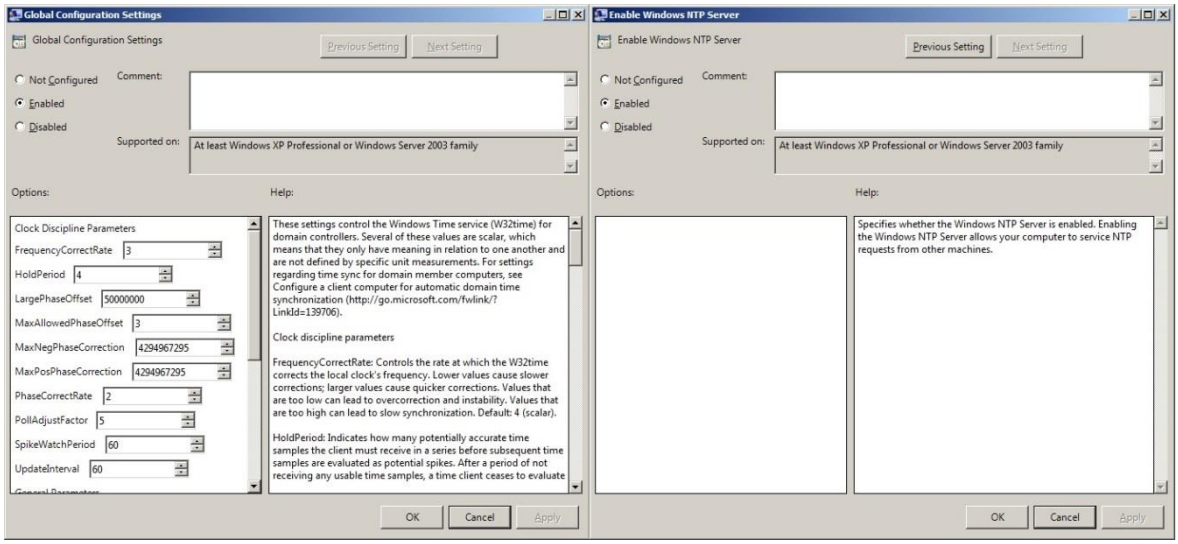

KUVIO 48. IO-laitteen päivitysajan muuttaminen

Päivitysajan pidentämisen jälkeen kaikki laitteet saivat yhteyden Profinettiin. WLAN-verkon hitauden takia kaikki Profinet IO-laitteet eivät saaneet lähetettyä IO-tietojaan eivätkä saaneet IO-ohjaimelta tietoa. Uskoisin myös, että WLAN-verkoissa käytettävä kilpavaraus menetelmä aiheutti tämän ongelman. Muutimme kaikkien laitteiden päivitysajat 32 millisekuntiin, ja ongelmasta selvittiin.

### 6.7 Logiikan ja paneelin kellonajan synkronointi käyttäen NTP-palvelinta

Kellonaikojen synkronointi on tärkeää automaatiolaitteistoissa. Logiikan kellonaikaa käytetään tapahtumien aikaleimoina ja logiikan diagnostiikassa. Kellojen pitäisi pysyä siis oikeassa ajassa myös kesä- ja talviaikaan siirtyessä. Kellonaikojen synkronoinnista haluttiin tehdä ohjeet ja esimerkkiohjelma käyttäen uudempia logiikoita. Siemensin sivuilta löytyivät ohjeet, miten tietokoneen saa määritteltyä NTP-palvelimeksi (Network Time Protocol). NTP-protokollalla voidaan synkronoida tietokoneita Ethernet-verkossa, joten se toimii myös Profinetissä. Kuviossa 49 on kaksi ikkunaa, joilla määritellään NTP-palvelin. Vasemmanpuoleisessa ikkunassa määritellään Windows Time Servicen asetukset. Käytin

Siemensin suosittelemia asetuksia tässä. Oikeanpuoleisessa ikkunassa asetetaan NTP-palvelin päälle.



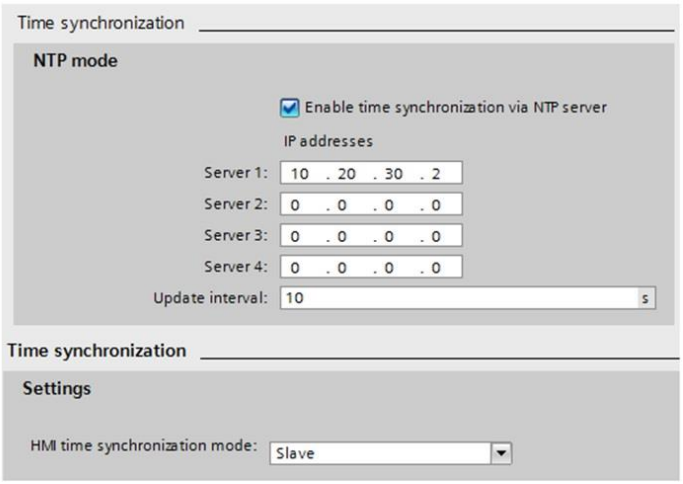

KUVIO 49. NTP-palvelimen määrittely

NTP-palvelimen määrittelyn jälkeen tarvitsee vain käytettävien logiikoiden asetukset laittaa kohdalleen. Logiikan asetuksissa voidaan määritellä NTP-palvelimen IP-osoite ja kellonajan päivityksen aikaväli (KUVIO 50).

Paneelin kellonajan synkronointi logiikan kanssa tehdään yhdessä valikossa. Paneeli voi olla joko master- tai slave-moodissa ajan synkronoinnissa. Master-moodissa paneelin kellonaikaa käytetään muissa laitteissa ja slave-moodissa paneeli käyttää logiikan kellonaikaa. Asetus löytyy logiikan ja paneelin välisen yhteyden asetuksista (KUVIO 50).

Logiikan asetukset

Logiikan ja paneelin välisen yhteyden asetukset




KUVIO 50. Logiikan ja paneelin kellonaikojen synkronointi

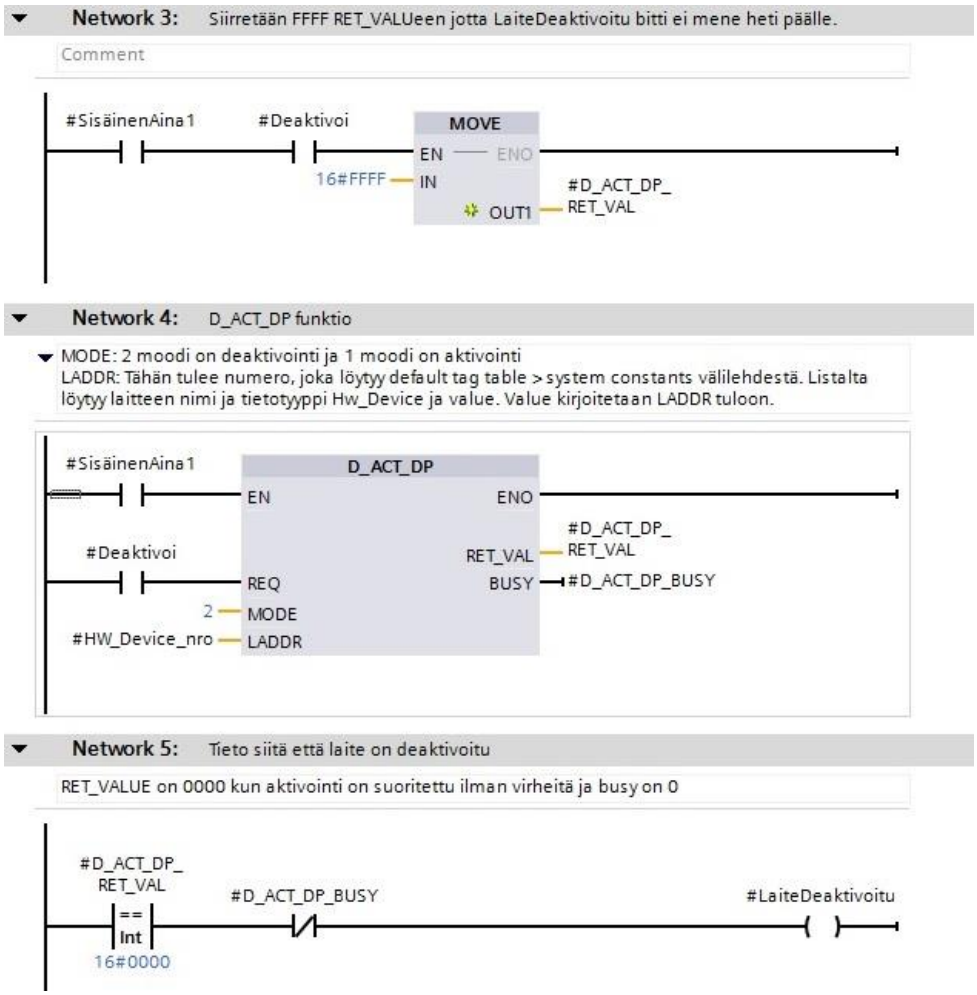
Kellonaikojen synkronointi on tehty helpoksi koska minkäänlaista ohjelmakoodia ei tarvita kellonaikojen synkronointiin.

### 6.8 Fast start-up laitteen aktivointi ja deaktivointi

Fast start-up ominaisuudella varustetut laitteet käynnistyvät sähkökatkon jälkeen nopeammin kuin normaalit laitteet ja luovat yhteyden IO-ohjaimeen myös nopeammin. Tällaista toimintoa käytetään esimerkiksi robottien työkalunvaihtajissa. Tätä toimintoa haluttiin testata Siemensin ET 200eco IO-yksiköllä, jossa on fast start-up ominaisuus. Tehtäväksi sain etsiä ohjelmalohkot, joilla voidaan deaktivoida ja aktivoida Profinet IO -laitteita. Normaalin Profinet IO -laitteen kadottaessa yhteyden ilman ohjelman tekemää käskyä aiheuttaa logiikan menemisen vikatilaan. Tämä ei ole toivottavaa, jos ohjelman tarkoituksena on poistaa laite käytöstä. Siemensin ohjeita ja muita ohjeita lukemalla selvitin mitä asioita pitää ottaa huomioon tällaista laitetta määriteltäessä.

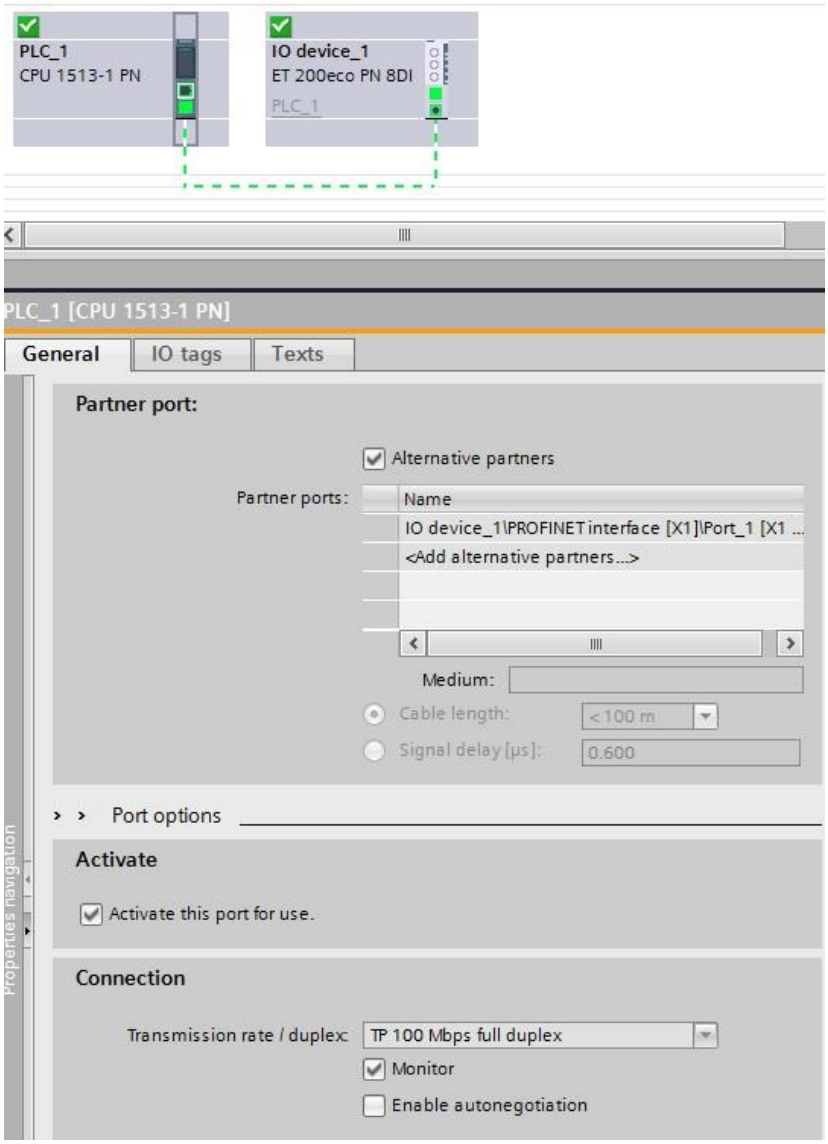
Suuren määrän internet-hakuja tehtyäni löysin Siemensin sivuilta tieto D_ACT_DP lohkosta, jota on käytetty Profibus-laitteiden poistamiseen käytöstä ja aktivoimiseen. Tätä samaa lohkoa voi käyttää myös Profinet-laitteiden ohjamiseen. Samalla lohkolla siis voidaan ohjata laite päälle ja pois. Moodissa yksi lohko aktivoi laitteen ja moodissa kaksi laite deaktivoidaan. Laitteen, jota halutaan ohjata, lohko tunnistaa Hw_Device-numerosta. TIA Portalista tämä numero löytyy Default tag table-valikon välilehdestä System constants. Laitetta vastaava numero etsitään listalta ja se kirjoitetaan LADDR-sisääntuloon (KUVIO 51).

Ohjelmassa ensin alustetaan lohkon RET_VAL-muuttuja, jotta kun laite deaktivoidaan onnistuneesti, muuttuu arvo nollaksi. Tällöin voidaan ilmoittaa, että laite on deaktivoitu onnistuneesti (KUVIO 51). Laitteen aktivointi tehdään täysin identtisellä ohjelmalla mutta lohkon moodi on muutettu aktivointiin.




KUVIO 51. Profinet IO -laitteen deaktivointi

FSU-ominaisuuden toimiakseen täytyy laitteiden määrittelyihin tehdä muutoksia. IO-laitteen Profinet-portin asetuksista täytyy tiedonsiirtonopeudeksi valita TP 100 Mbps full duplex ja autonegotiation ottaa pois päältä. Nämä samat muutokset pitää tehdä IO-ohjaimen portin asetuksiin, johon IO-laite on kytketty. IO-ohjaimen portista täytyy vielä laittaa alternative partners asetus päälle (KUVIO 52). Tällä tavoin vaikka deaktivoidusta laitteesta katoaa sähköt se ei aiheuta logiikan menemistä vikatilaan. Siemensin mukaan näillä asetuksilla voidaan päästä jopa 500 millisekunnin käynnistysaikoihin.




KUVIO 52. 1500 -logiikan portin 2 asetukset

Tuloksena syntyi esimerkkiohjelma, jolla voidaan poistaa laite käytöstä ja laittaa se takaisin käyttöön. Tämä voidaan suorittaa ilman, että logiikka menee vikatilaan. Tällaista kokoonpanoa ei oikeassa elämässä kuitenkaan ole vaan logiikan ja vaihdettavien laitteiden välissä on erillinen telakointiasema. Idea on kuitenkin sama oikeissa sovelluksissa.

### 6.9 Paikoittaminen käyttäen CU250S-2 -ohjausyksikköä ja pulssianturia

CU250S-2 on Siemensin valmistama ohjausyksikkö, johon on mahdollista kytkeä pulssiantureita. Ohjausyksikössä on sisäinen paikoitus eli logiikalta annetaan vain paikkaohje tai käsky suorittaa tietty paikoitustehtävä. Tietoperustan luvussa 5 on tarkemmin selitettynä

mitä ohjausyksiköllä voidaan tehdä ja miten se otetaan käyttöön. Kyseistä laitetta ryhdyin tutkimaan koska Apexilla haluttiin tietää miten se toimii ja mitä sillä on mahdollista tehdä. Se julkaistiin vuonna 2013 joten siitä ei ollut kenelläkään aiempaa kokemusta.

Ohjausyksikön testaaminen ei aluksi meinannut onnistua ollenkaan. Kului monta päivää, että sain pulssianturin toimimaan. Pulssianturi, jota aluksi käytin, oli jäänyt yli jostain projektista. Kytkin siihen kaapelin ja D-liittimen kaapelin toiseen päähän. Ohjausyksikössä on D-liitin pulssianturin kytkemistä varten. Ensimmäisen liittimen tehtyäni jätin käyttöjännitteen takaisenkytkennän ja invertoidun nollapisteen kytkemättä koska kaapelissa oli liian vähän johtimia. Ohjausyksikkö ilmoitti viasta heti kun sain pulssianturin kytkettyä siihen. Uuden kaapelin ja liittimen tehtyäni pääsin testaamaan moottoria. Testaus ei kuitenkaan onnistunut koska, aina kun yritin ohjata moottorin päälle, taajuusmuuttaja ilmoitti viasta. Yhden vian korjaamisen jälkeen tuli uusi vika. Kauan ongelmaa tutkittuani päädyin siihen lopputulokseen, että moottorissa on jotain vikaa. Moottori aluksi pyöri hyvin, mutta myöhemmin moottorin ottama virta ylitti virtarajan aina kun sitä yritettiin ajaa. Kuormittamaton moottori ei normaalisti saisi tehdä tällaista. Moottori myös kuumeni huomattavasti, kun sitä ajettiin.

Seuraava ongelma syntyi kun uusi moottori oli hankittu. Pulssianturi, jota aikaisemmin olin käyttänyt, ei toiminut. Jostain selittämättömästä syystä vain yksi pulssianturin kanavista antoi pulsseja. Totesin tämän mittaamalla yleismittarilla kanavia samalla kun pyöritin anturin akselia. Uuden anturin saatuani kytkin sen suoraan ohjausyksikön johtoterminaaliin. Uutta moottoria ja uutta pulssianturi käyttäen sain taajuusmuuttajan käyttöönoton tehtyä ja paikoittajan määriteltyä Siemensin ohjeiden mukaan. Kuviossa 53 on käytetty testilaitteisto. Kuvasta puuttuu vain 1500 -logiikka, jota käytin. Moottori on Bevin valmistama nimellisteholtaan 0,37 kW vaihtosähkömoottori. Pulssianturi on ifm:n valmistama inkrementaalianturi, joka antaa 3600 pulssia kierroksella.


KUVIO 53. Taajuusmuuttaja, pulssianturi ja moottori

Seuraavaksi ryhdyin selvittämään miten ja mitä logiikalla voidaan ohjata. Päädyin käyttämään telegrammia 111 tiedonsiirtoon. Tämä telegrammi sisältää paikoituslohkojen ohjauksen sekä suoraohjauksen. Ohjausyksikölle pitää kertoa mitä telegrammia käytetään ja TIA Portalissa täytyy määritellä telegrammi myös. TIA Portalissa määritellään myös mitä sisään- ja ulostulo-osoitteita käytetään. Tässä ohjelmassa en kuitenkaan käyttänyt IO:ta suoraan ohjaamaan taajuusmuuttajaa.

Nykyaikaisissa logiikoissa käytetään symbolisia osoitteita, joten on luontevaa käyttää niitä myös sisään- ja ulostuloihin viittaamiseen. Tähän voidaan käyttää PLC-tietotyyppejä kokoamaan Profinet IO -laitteen käyttämä IO-alue datalohkoon. Aikaisemmin esimerkiksi sisääntuloihin täytyi viitata osoitteella I0.0, joka tarkoittaa ensimmäisen tavun ensimmäistä bittiä. Tällaisia osoitteita käyttäessä ohjelmoijan täytyy ottaa bittiavaruus huomioon ja osoitteet eivät ole havainnollisia. Löysin Siemensin sivuilta valmiit PLC-tietotyypit eri telegrammeille. Käytin telegrammi 111 PLC-tietotyyppiä luomaan datalohkon, jota käytin

tiedonsiirrossa. PLC-tietotyypillä sain helposti symboliset nimet kaikille ohjausbiteille sekä jokainen muuttuja sisälsi myös kommentin, joka kertoi mitä kyseisellä bitillä tehdään. Kopioimalla PLC-tietotyypistä lähetykseen käytetyn struktuurin ja liittämällä sen FB-lohkon ulostuloon pystyin viittaamaan samoilla nimillä muuttujiin FB-lohkon sisällä. Kuviossa 54 on datalohko, jota käytin tiedonsiirtoon. Telegrammi on yhteensä 12 sanan mittainen eli se sisältää 192 bittiä. Lähettämiseen ja vastaanottamiseen käytetään molempiin 12 sanaa. PLC-tietoyyppiä käyttäessä en joutunut missään vaiheessa miettimään mitä mikäkin bitti missäkin sanassa tarkoittaa.

**Telegram 111**, positioning operation with extended functions

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| STW1 | POS_ STW1 | POS_ STW2 | STW2 | OVER RIDE | MDI_TARPOS | MDI_VELOCITY | MDI_ ACC | MDI_ DEC | Free |
| ZSW1 | POS_ ZSW1 | POS_ ZSW2 | ZSW2 | MELDW | XIST_A | NIST_B | WARN_ CODE | FAULT_ CODE | Free |

Tiedonsiirto

| | Name | Data type | Start value | Retain | Accessible f... | Visible in ... | Setpoint | Comment |
|---|---|---|---|---|---|---|---|---|
| 1 | Static | | | | | | | |
| 2 | Send | Struct | | | | | | |
| 3 | STW_1 | Struct | | | | | | Control Word Interconnection 1 |
| 4 | POS_STW_1 | Struct | | | | | | Positioning control word |
| 5 | POS_STW_2 | Struct | | | | | | Positioning control word |
| 6 | STW_2 | Struct | | | | | | Control Word Interconnection 2 |
| 7 | OVERRIDE | Int | 0 | | | | | override speed setpoint |
| 8 | MDI_TARPOS | DInt | 0 | | | | | Position Setpoint for direct setpoint input (MDI) |
| 9 | MDI_VELOCITY | DInt | 0 | | | | | MDI velocity |
| 10 | MDI_ACC | Int | 0 | | | | | MDI acceleration |
| 11 | MDI_DEC | Int | 0 | | | | | MDI deceleration |
| 12 | USER_REC | Int | 0 | | | | | User |
| 13 | Receive | Struct | | | | | | |
| 14 | ZSW_1 | Struct | | | | | | Status Word Interconnection 1 |
| 15 | POS_ZSW_1 | Struct | | | | | | Status word for basic positioner |
| 16 | POS_ZSW_2 | Struct | | | | | | Status word for basic positioner |
| 17 | ZSW_2 | Struct | | | | | | Status word Interconnection 2 |
| 18 | MELDW | Struct | | | | | | Status word for messages |
| 19 | XIST_A | DInt | 0 | | | | | Actual position value [LU] |
| 20 | NIST_B | DInt | 0 | | | | | Actual speed value 32 Bit |
| 21 | WARN_CODE | Int | 0 | | | | | Number of the actual alarm |
| 22 | FAULT_CODE | Int | 0 | | | | | Number of the actual fault |
| 23 | USER_SEND | Int | 0 | | | | | User |

KUVIO 54. Telegrammi 111 ja tiedonsiirtoon käytetty datalohko

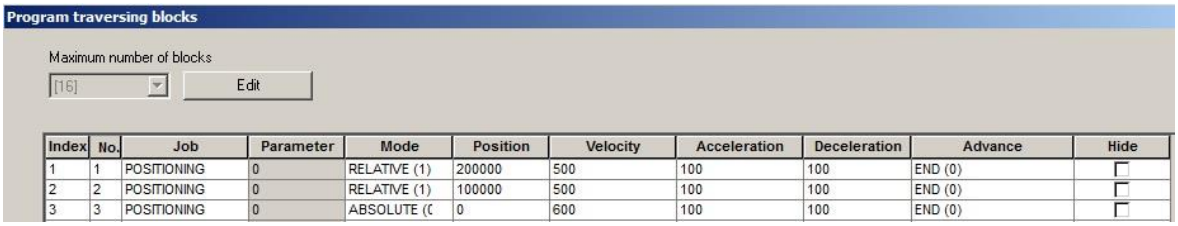
PLC-tietotyypin käyttäminen aiheutti tiedonsiirrossa kuitenkin odottamattomia ongelmia. Yritin ohjata taajuusmuuttajaa logiikan kautta ja jostain syystä, vaikka kaikki oli oikein ohjelmassa, taajuusmuuttaja ei lähtenyt käyntiin. Tutkimalla Profinetin diagnostiikkatietoja Starter-ohjelmistossa huomasin, että ne bitit joiden piti olla ohjaussanan alussa, olivatkin sanan lopussa. Sanan ylempi ja alempi tavu olivat siis vaihtaneet paikkaa. Kysyin tästä työkaveriltani ja hän sanoi, että tämä yleinen ongelma, johon on törmätty aikaisemminkin. PLC-tietotyypin hienous tuli esiin tässä koska muuttamalla vain PLC-tietotyypissä tavut ristiin vaihtuivat ne datalohkossakin. Tämä ei tietenkään säästänyt aikaa kuin muutaman

minuutin mutta, jos taajuusmuuttajia olisi ollut useampia, olisi aikaa säästynyt huomattavasti.

Tiedon siirtämiseen käytin kahta lohkoa. DPRD_DAT- ja DPWR_DAT-lohkoilla voidaan kirjoittaa ja lukea yhtenäistä (consistent) dataa Profinet IO -laitteelta. Lohkot tarvitsevat toimiakseen vain datalohkon, jota käyttää tiedonsiirtoon, ja käytetyn telegrammin HW-tunnuksen. HW-tunnus löytyy TIA Portalissa käytetyn telegrammin ominaisuuksista. Lohkot antavat myös virhesanoman, jos tapahtuu virhe. Kuviossa 55 on molemmat lohkot. DPRD_DAT kirjoittaa tiedonsiirto-datalohkon receive-struktuuriin tietoa taajuusmuuttajan toiminnasta ja DPWR_DAT lukee tietoa datalohkosta ja lähettää sen taajuusmuuttajalle.


KUVIO 55. Kommunikointi-network

Kommunikoinnin saatuani toimimaan oikein ryhdyin testaamaan, miten paikoitus toimii. Ensimmäisenä testasin, miten paikoituslohkoja voidaan ohjata logiikalta. Loin kolme lohkoa Starter-ohjelmalla ja tallensin ne ohjausyksikköön (KUVIO 56). Kaksi ensimmäistä lohkoa käyttävät suhteellista paikoitusta, eli ne laskevat matkan akselin hetkellisestä pisteestä lähtien. Paikkaohjeet annetaan LU:na. Viimeinen lohko ajaa absoluuttiseen nollaan. Nopeusarvo on ilmoitettu Starterissa 1000 LU/min. Valitsin resoluutioksi 0,1 astetta, joten 500 1000 LU/min tarkoittaa, että kierrosnopeus on noin 140 kierrosta minuutissa. Lohkot valitaan kirjoittamalla ohjausyksikölle lohkon numeroa vastaava numero binäärisenä käyttäen neljää bittiä. Tein FC-lohkon, jolla muutetaan int-tyyppinen luku bittejä vastaavaksi. Tällä tavoin ei tarvitse miettiä, miten luvut esitetään binäärisenä, joka kerta kun eri lohkoa halutaan käyttää.




| Index | No. | Job | Parameter | Mode | Position | Velocity | Acceleration | Deceleration | Advance | Hide |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | POSITIONING | 0 | RELATIVE (1) | 200000 | 500 | 100 | 100 | END (0) | ☐ |
| 2 | 2 | POSITIONING | 0 | RELATIVE (1) | 100000 | 500 | 100 | 100 | END (0) | ☐ |
| 3 | 3 | POSITIONING | 0 | ABSOLUTE (( | 0 | 600 | 100 | 100 | END (0) | ☐ |

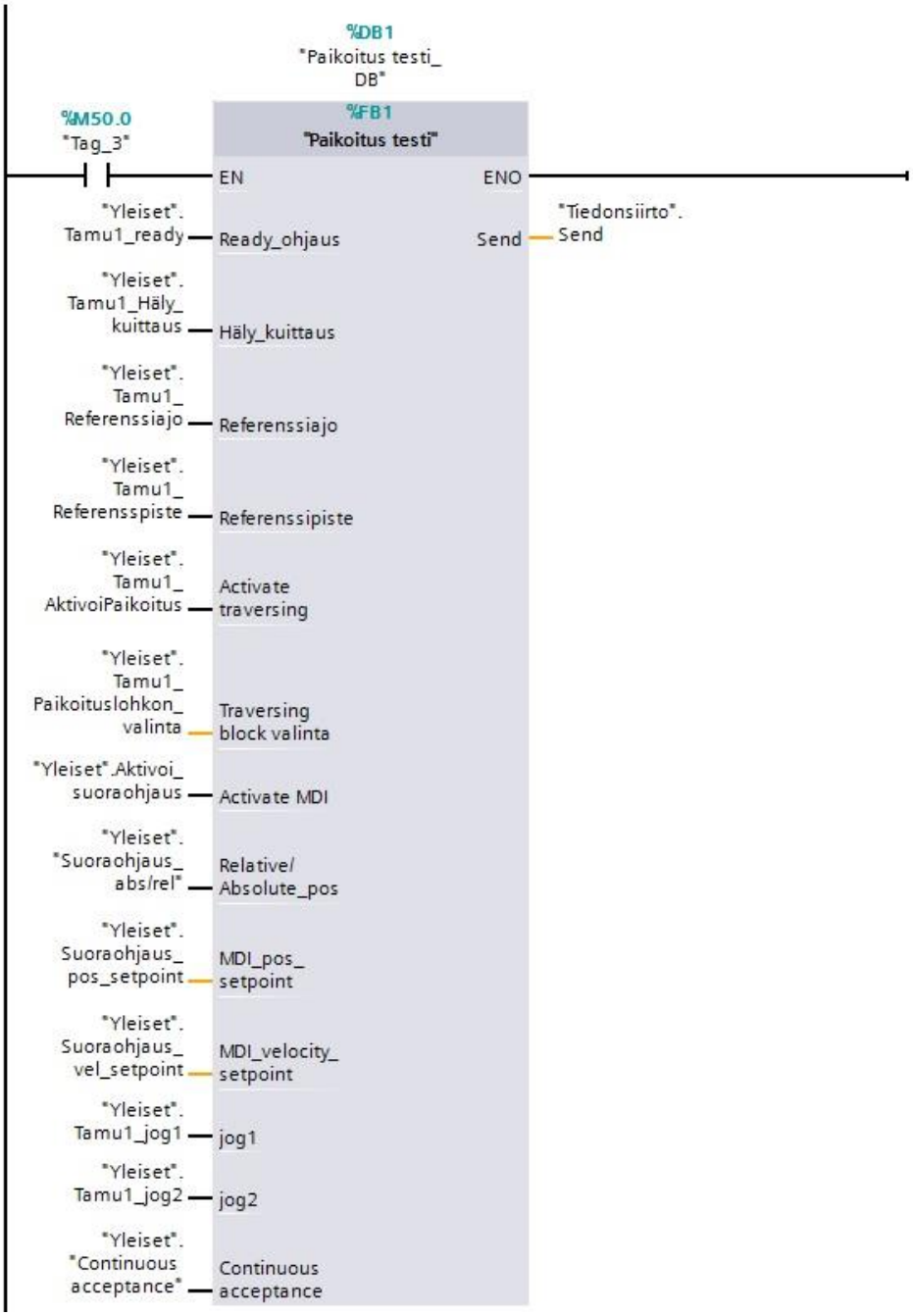
KUVIO 56. Paikoituslohkot

Paikoituksen toimiakseen täytyy ohjausyksikön tietää missä moottorin liikuttama kuorma on. Käytössäni oli inkrementaalianturi, joten sähkökatkon jälkeen pitää hakea referenssipiste. Referenssipisteen simuloimiseen käytin muuttujaa, jota ohjasin TIA Portalin kautta päälle ja pois. Referenssipisteen määrittelystä ja paikoituslohkoista on lisää tietoa luvussa 5.2.

Aluksi en saanut moottoria pyörimään, vaikka kaikki tiedot siirtyivät oikein ohjausyksikköön. Starter-ohjelmiston diagnostiikasta sain selville, että moottori ei ehkä pyöri koska nopeuden ohitus on 0 %. Ohjekirjaa lukemalla sain selville, että override-ohjaussanaa käytetään kertomaan millä prosenttimäärällä nopeuden ohjearvo saa ohittaa määritellyn maksiminopeusmäärän. Käytin 100 % override arvoa testauksessa. Samalla lailla on paikoituslohkon kiihdyttämisen ja jarruttamisen kiihtyvyyden ohitus ilmoitettu paikoituslohkoissa. Suoraohjauksessa käytetään override-ohjaussanan lisäksi MDI_ACC- ja MDI_DEC-ohjaussanoja kiihtyvyyksien ohittamisen ilmoittamiseen.

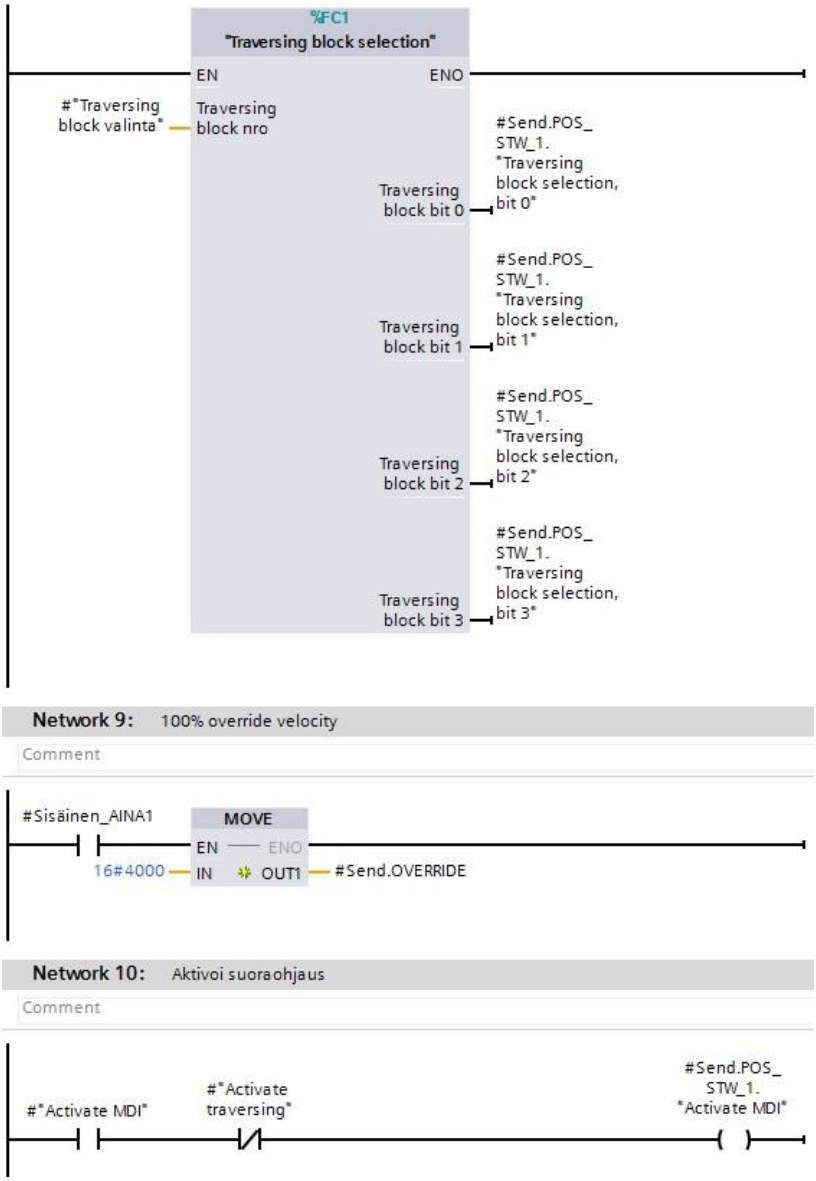
Kuviossa 57 on FB-lohko, johon tein ohjelman taajuusmuuttajan ohjaamiselle. Ulostuloksi riittää vain Tiedonsiirto-datalohkossa käytetty Send-struktuuri. Tämä siirretään Tiedonsiirto-datalohkoon ja sitten se lähetetään taajuusmuuttajalle (KUVIO 55). Kopioin struktuurin PLC-datatyypistä ja liitin sen FB-lohkon instanssidatalohkoon. Sisääntuloilla ohjataan taajuusmuuttaja päälle ja sitten annetaan sille paikoitusohjeita. Referenssiajo-sisääntulolla käynnistetään referenssiajo ja referenssipisteellä kerrotaan milloin kuvitteellinen kuorma on referenssissä. Tässä testissä pulssianturi oli vain suoraan kiinni moottorin akselissa, joten oikeaa referenssipistettä ei ollut fyysisesti olemassa. Activate traversing-sisääntulolla käynnistetään valittu paikoituslohko. Paikoituslohko valitaan Traversing block valinta-sisääntulolla. Siihen tuodaan int-tyyppinen luku 1–16 väliltä. Activate MDI käynnistää suoraohjauksen ja MDI_pos_setpoint-sisääntuloon kirjoitetaan haluttu paikka LU:na. Relative/Absolute_pos kertoo käytetäänkö suhteellista vai absoluuttista paikoitusta suoraohja-

uksessa. En kuitenkaan saanut absoluuttista paikoitusta toimimaan testissä. Sain vika ilmoituksen, että Continuous acceptance ei saa olla päällä kun käytetään absoluuttista paikoitusta mutta sen ottaminen pois päältä ei auttanut. Sisääntuloilla jog1 ja jog2 voidaan moottoria ajaa myötä- tai vastapäivään määritellyn matkan tai niin kauan kuin jog on päällä. Matkat ja nopeudet määrittelin Starter-ohjelmassa.




KUVIO 57. FB-lohko paikoituksen testaamiseen

FB-lohkon sisällä periaatteessa siirretään vain sisääntulojen ohjaamat bitit Send-struktuurin bitteihin. Lisäksi siellä puretaan paikoituslohkon numero neljään bittiin, joilla kerrotaan mitä paikoituslohkoa halutaan käyttää. Lisäksi siellä on määritelty nopeuksien ja kiihtyvyyksien ohitus arvot. Kuviossa 58 on paikoituslohkon valintaan käytetty FC-lohko, nopeuden ohitus arvon siirtäminen ja suoraohjauksen käynnistäminen. Käytin FC-lohkoa koska tällainen toimenpide ei tarvitse syklistä muistia. Kaikki taajuusmuuttajan ohjaamiseen käytetyt muuttujat käyttävät samoja nimiä kuin PLC-tietotyypissä. Tällä tavoin ohjelma on yhtenäisempi ja selkeämpi.




KUVIO 58. FB-lohkon sisältö

Tämän ohjelman tekeminen oli kaikkein eniten aikaa vievä ja vaativin, koska se vaati niin paljon tiedon hakemista. Taajuusmuuttajan ja paikoittajan määrittelyn tekeminen oikein vaati monta yritystä ja moottorin ja pulssianturin hajoaminen aiheuttivat ylimääräisiä ongelmia. Ohjelman tekeminen oli kuitenkin mielenkiintoista ja käytännönläheistä. Eniten aikaa vei taajuusmuuttajan ilmoittamien vikojen korjaaminen etsimällä ratkaisua niihin ohjekirjoista ja internetistä. PLC-tietotyyppien käyttäminen kommunikoinnissa oli uusi asia minulle. Tietotyypit aiheuttivat ongelmia mutta ongelmat olivat helposti korjattavissa. Aikaa säästyy vaikka ongelmia syntyisikin.

# 7 POHDINTA

Aloittaessani opinnäytetyön tekemisen olin huolissani, että työstä paisuisi liian laaja enkä ehtisi saada sitä valmiiksi määräaikaan mennessä. Toimeksiantajan puolelta tuli paljon ideoita ja asioita, joita he halusivat tutkia, ja selvittää ja osa niistä asioista ei loppujen lopuksi ehtinytkään tähän opinnäytetyöhön. Tulevaisuudessa kuitenkin näitä pois jääneitä asioita on tarkoitus selvittää. Opinnäytetyötä tehdessä en kuitenkaan koskaan tuntenut, että minulla on kiire, koska sain keskittyä siihen rauhassa. Opinnäytetyön aihe ja asiat, joita siinä käsiteltiin, olivat minusta mielenkiintoisia ja se auttoi motivaatiotani. Työkavereiltani saamat kehut ja kommentit olivat myös apuna.

Alun perin opinnäytetyöhön olisi pitänyt kuulua standardiohjelmarungon sisältävä esimerkkiohjelma, joka tulisi Apexin sisäiseen käyttöön. Tällaisella ohjelmalla olisi haluttu luoda yhtenäiset ohjelmointitavat firman sisälle. Ohjelmaa ei koskaan kuitenkaan ehditty suunnitella muiden kiireiden takia ja opinnäytetyön ollessa jo valmiiksi aika laaja. TIA Portalin sisältämät teknologiakirjastot oli myös tarkoitus tutkia ja testata opinnäytetyöhöni liittyen mutta ajan puutteen takia nekin jäivät pois.

Asiat, jotka ehdin opinnäytetyössäni käsitellä, olivat kuitenkin tarpeeksi monimutkaisia ja kiinnostavia, että asioiden pois jääminen ei jäänyt harmittamaan. Mielenkiintoisimmat aiheet, joita käsittelin, olivat paikoitussovelluksen tekeminen, HTML-sivujen käyttäminen valvomona ja IWLAN-verkon ongelmien selvittäminen. Kaikki kolme aihetta olivat täysin uusia minulle ja sen takia mielenkiintoisia. Lisäksi kaikissa kolmessa aiheessa oli asioita, joita toivotettavasti pystytään tulevaisuudessa hyödyntämään. IWLAN-verkon rakentaminen on varmasti yksi asia, jota tullaan tulevaisuudessa tekemään enemmän.

TIA Portalin ja uusien logiikoiden käyttäminen opinnäytetyössäni olivat myös asioita, joiden takia opinnäytetyö oli mielenkiintoinen. Uusien ohjelmointitapojen selvittäminen ja niiden käyttäminen ohjelmissa opettivat minua luomaan selkeämpiä ohjelmia ja käytän varmasti monia oppimiani asioita tulevaisuudessakin. Opinnäytetyön kirjallista osuutta tehdessäni opin myös paljon asioita ja päädyin korjaamaan ohjelmia, kun sain uusia ideoita. Kirjallinen osuus opinnäytetyöstäni onnistua mielestäni hyvin ja sain siinä hyvin käsiteltyä asioita, jotka liittyivät keskeisesti opinnäytetyöhön.

**LÄHTEET**

Apex Automation Oy. 2011. Www-sivu. Saatavissa: http://www.apexautomation.fi/fi. Luettu 5.4.2014.

Automation.com. 2013. Www-sivu. Saatavissa: http://www.automation.com/product-showcase/siemens-announces-Sinamics-cu250s-2-control-unit. Luettu 13.5.2014

Basic Positioner. 2013. Basic Positioner Function Manual. PDF-dokumentti. Saatavissa: http://cache.automation.siemens.com/dnl/TQ/TQ1NjYyOQAA_72918700_HB/Function_Manual_Basic_Positioner_en-US.pdf. Luettu 15.5.2014.

Cycle and response times. 2013. S7-1500, ET 200MP, ET200SEP Cycle and response times. PDF-dokumentti. Saatavissa: http://cache.automation.siemens.com/dnl/zE/zE1Mjg1AAAA_59193558_HB/s71500_cycle_and_reaction_times_function_manual_en-US_en-US.pdf. Luettu 15.4.2014.

CU250S-2 Control Unit. 2013. SINAMICS G120 Converter with the CU250S-2 Control Unit. PDF-dokumentti. Saatavissa: http://cache.automation.siemens.com/dnl_iis/Tc/TcyODc2OQAA_81164948_HB/Control_Units_CU250S-2_Servo_en-US.pdf. Luettu 13.5.2014.

Function manual. 2013. S7-1500 Structure and Use of the CPU Memory. PDF-dokumentti. Saatavissa: http://cache.automation.siemens.com/dnl/jE/jExMzU2NQAA_59193101_HB/s71500_structure_and_use_of_the_PLC_memory_function_manual_en-US_en-US.pdf. Luettu 15.4.2014.

Leine & Linde. 2012. DRIVE-CLiQ-anturit raskaisiin käyttöolosuhteisiin. Www-dokumentti. Saatavissa: http://www.leinelinde.fi/uutiset/lehdistotiedote/drive-cliq/. Luettu 13.5.2014.

Niiranen, J. 1999. Sähkömoottorikäytön digitaalinen ohjaus. 2., korjattu painos Helsinki: Otatieto Oy. Luettu 30.5.2014.

Pigan, R. & Matter, M. 2008. Automating with PROFINET. 2., uudistettu painos. Erlangen: Publicis Publishing. Luettu 12.5.2014.

PLCTutor. 2013. What is a PLC? Www-sivu. Saatavissa: http://www.plctutor.com. Luettu 15.4.2014.

Profinet. 2014. Profinet. Www-sivu. Saatavissa: http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/teollinen_tiedonsiirto_esim_profinet/profinet.htm. Luettu 19.5.2014.

Programming guideline. 2014. Programming Guideline for S7-1200/1500. PDF-dokumentti. Saatavissa: http://cache.automation.siemens.com/dnl/zE/zExMzE4MQAA_81318674_Tools/81318674_Programming_guideline_DOKU_v12_en.pdf. Luettu 14.4.2014.

Puska, M. 2005. Langattomat lähiverkot. Jyväskylä: Talentum Media Oy. Luettu 19.5.2014

S7-1500. 2014. Tehokasta automaatio-ohjelmointia S7-1500 -logiikalla. Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_Simatic/s7_1500.php. Luettu 15.4.2014.

S7-1500 Web server. 2013. Simatic S7-1500 Web server Function Manual. PDF-dokumentti. Saatavissa:
https://a248.e.akamai.net/cache.automation.siemens.com/dnl/zM/zM0MzkyOQAA_59193560_HB/s71500_webserver_function_manual_en-US_en-US.pdf. Luettu 28.4.2014.

SIMATIC STEP 7. 2014. TIA Portal (SIMATIC STEP 7). Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_Simatic/ohjelmistot/tia_portal_Step7.htm. Luettu 11.4.2014.

Sinamics G. 2014. Sinamics G. Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/kayttotekniikka_ja_liikkeenohjaus/sahkokaytot/taajuusmuuttajat/Sinamics_g.htm. Luettu 24.6.2014

SINAMICS StartDrive. 2014. SINAMICS StartDrive-ohjelmisto. Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/kayttotekniikka_ja_liikkeenohjaus/liikkeenohjausjarjestelmat/Sinamics_drive_solutions/Sinamics_startdrive.htm. Luettu 11.4.2014.

STARTER. 2014. STARTER Commissioning Software. Www-sivu. Saatavissa:
http://www.automation.siemens.com/mcms/mc-solutions/en/engineering-software/starter-commissioning-tool/Pages/starter-commissioning-tool.aspx. Luettu 15.5.2014.

STEP 7. 2013. STEP 7 Professional V12.0 SP1. PDF-dokumentti. Saatavissa:
http://cache.automation.siemens.com/dnl/Dg/Dg0NDk0NQAA_77991795_HB/STEP_7_Professional_V12_SP1_enUS_en-US.pdf. Luettu 22.4.2014.

System manual. 2014. SIMATIC S7-1500 System Manual. PDF-dokumentti. Saatavissa:
http://cache.automation.siemens.com/dnl/DQ/DQ3MTQ3NTEA_59191792_HB/s71500_system_manual_en-US_en-US.pdf. Luettu: 15.4.2014.

TIA Portal. 2014. TIA Portal – teollisuusautomaation ohjelmistoalusta. Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/tia_portal.php. Luettu 11.4.2014.

United States Patent. 2005. United States Patent US6977449. PDF-dokumentti. Saatavissa:
http://www.freepatentsonline.com/6977449.pdf. Luettu 30.5.2014.

WinCC V11. 2014. TIA Portal (SIMATIC WinCC V11). Www-sivu. Saatavissa:
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/kayttoliittymat/ohjelmistot/tia_portal_wincc.php. Luettu 11.4.2014.

See also: Static parameters (Page 2516)

## See also

Specifying the interface parameter set (Page 2526)
Specifying the memory space model of the data block (Page 2527)
GRAPH programming language (Page 2486)
Sequencers (Page 2488)
GRAPH-specific operands (Page 2495)
Actions (Page 2496)
Steps and transitions (Page 2492)
Permanent instructions (Page 2507)
Conditions (Page 2508)
Interlock and supervision alarms (Page 2509)

## Input parameters

### Overview of the input parameters of a GRAPH function block

The following table provides an overview of the input parameters of a GRAPH function block:

| Parameters | Data type | Description | Standard parameter set | Maximum parameter set |
|---|---|---|---|---|
| OFF_SQ | BOOL | OFF_SEQUENCE:<br>Switch off sequencer, i.e., deactivate all steps | x | x |
| INIT_SQ | BOOL | INIT_SEQUENCE:<br>Activate initial steps, reset sequencer | x | x |
| ACK_EF | BOOL | ACKNOWLEDGE_ERROR_FAULT:<br>Acknowledge a fault, force advance to next step | x | x |
| REG_EF | BOOL | REGISTRATE_ERROR_FAULT:<br>Register all errors and faults | - | x |
| ACK_S | BOOL | ACKNOWLEDGE_STEP:<br>Acknowledge step indicated in "S_NO" output parameter | - | x |
| REG_S | BOOL | REGISTRATE_STEP:<br>Register step indicated in "S_NO" output parameter | - | x |
| HALT_SQ | BOOL | HALT_SEQUENCE:<br>Stop/reactivate sequencer | - | x |
| HALT_TM | BOOL | HALT_TIMES:<br>Stop/reactivate all step activation times and time-dependent operations (L and D) of the sequencer | - | x |

| Parameters | Data type | Description | Standard parameter set | Maximum parameter set |
|---|---|---|---|---|
| ZERO_OP | BOOL | ZERO_OPERANDS:<br>Reset to zero all operands with identifier N, D, L in active steps and do not execute CALL instructions in actions/ reactivate operands and CALL instructions. | - | x |
| EN_IL | BOOL | ENABLE_INTERLOCKS:<br>Disable interlock (sequencer behavior is same as when interlock condition is satisfied)/re-enable | - | x |
| EN_SV | BOOL | ENABLE_SUPERVISIONS:<br>Disable supervision (sequencer behavior is same as when supervision condition is not satisfied)/re-enable | - | x |
| EN_ACKREQ | BOOL | ENABLE_ACKNOWLEDGE_REQUIRED:<br>Enable mandatory acknowledgement | - | x |
| EN_SSKIP | BOOL | ENABLE_STEP_SKIPPING:<br>Enable step skipping | - | x |
| DISP_SACT | BOOL | DISPLAY_ACTIVE_STEPS:<br>Display only active steps | - | x |
| DISP_SEF | BOOL | DISPLAY_STEPS_WITH_ERROR_OR_FAULT:<br>Display only steps with errors and faults | - | x |
| DISP_SALL | BOOL | DISPLAY_ALL_STEPS:<br>Display all steps | - | x |
| S_PREV | BOOL | PREVIOUS_STEP:Automatic mode: Page up through the currently active steps, display of step number in "S_NO" parameter<br>Manual mode: Display preceding step in "S_NO" (smaller number) | x | x |
| S_NEXT | BOOL | NEXT_STEP:<br>Automatic mode: Page down through the currently active steps, display of step number in "S_NO" parameter<br>Manual mode: Display next step in S_NO (larger number) | x | x |
| SW_AUTO | BOOL | SWITCH_MODE_AUTOMATIC:<br>Operating mode switchover: Automatic mode | x | x |
| SW_TAP | BOOL | SWITCH_MODE_TRANSITION:<br>Operating mode switchover: Semi-automatic mode | x | x |
| SW_TOP | BOOL | SWITCH_MODE_TRANSITION_OR_PUSH:<br>Operating mode switchover: Automatic or semi-automatic mode | x | x |
| SW_MAN | BOOL | SWITCH_MODE_MANUAL:<br>Operating mode switchover: Manual mode, an independent sequence is not initiated | x | x |
| S_SEL | INT | STEP_SELECT:<br>Select step number for "S_NO" output parameter in manual mode, enable/disable with "S_ON", "S_OFF" | x | x |
| S_SELOK | BOOL | STEP_SELECT_OK:<br>Apply value in "S_SEL" for "S_NO" output parameter | - | x |

| Parameters | Data type | Description | Standard parameter set | Maximum parameter set |
|---|---|---|---|---|
| S_ON | BOOL | STEP_ON:<br>Manual mode: Activate displayed step | x | x |
| S_OFF | BOOL | STEP_OFF:<br>Manual mode: Deactivate displayed step | x | x |
| T_PREV | BOOL | PREVIOUS_TRANSITION:<br>Show previous valid transition in "T_NO" output parameter | - | x |
| T_NEXT | BOOL | NEXT_TRANSITION:<br>Show next valid transition in "T_NO" output parameter | - | x |
| T_PUSH | BOOL | PUSH_TRANSITION:<br>Transition advances to next step, when condition is satisfied and "T_PUSH" (edge)<br>Requirement: Automatic mode or manual mode | x | x |

See also

Basics on the block interface of a GRAPH function block (Page 2510)
Output parameters (Page 2514)
Static parameters (Page 2516)
Specifying the interface parameter set (Page 2526)
Specifying the memory space model of the data block (Page 2527)
GRAPH programming language (Page 2486)
Sequencers (Page 2488)
Steps and transitions (Page 2492)
GRAPH-specific operands (Page 2495)
Actions (Page 2496)
Permanent instructions (Page 2507)
Conditions (Page 2508)
Interlock and supervision alarms (Page 2509)

## Output parameters

### Overview of the output parameters of a GRAPH function block

The following table provides an overview of the output parameters of a GRAPH function block:

| Parameters | Data type | Description | Standard parameter set | Maximum parameter set |
|---|---|---|---|---|
| S_NO | INT | STEP_NUMBER:<br>Display of step number | x | x |
| S_MORE | BOOL | MORE_STEPS:<br>Additional steps are active | x | x |
| S_ACTIVE | BOOL | STEP_ACTIVE:<br>Displayed step is active | x | x |
| S_TIME | TIME | STEP_TIME:<br>Step activation time | - | x |
| S_TIMEOK | TIME | STEP_TIME_OK:<br>Step activation time is error free | - | x |
| S_CRITLOC | DWORD | STEP_CRITERIA_INTERLOCK:<br>Interlock criteria bits | - | x |
| S_CRITLOCERR | DWORD | S_CRITERIA_IL_LAST_ERROR:<br>Interlock criteria bits for event L1 | - | x |
| S_CRITSUP | DWORD | STEP_CRITERIA_SUPERVISION:<br>Supervision criteria bits | - | x |
| S_STATE | WORD | STEP_STATE:<br>Step status bits | - | x |
| T_NO | INT | TRANSITION:<br>Valid transition number | - | x |
| T_MORE | BOOL | MORE_TRANSITIONS:<br>Additional valid transitions available for display | - | x |
| T_CRIT | DWORD | TRANSITION_CRITERIA:<br>Transition criteria bits | - | x |
| T_CRITOLD | DWORD | T_CRITERIA_LAST_CYCLE:<br>Transition criteria bits from the last cycle | - | x |
| T_CRITFLT | DWORD | T_CRITERIA_LAST_FAULT:<br>Transition criteria bits for event V1 | - | x |
| ERROR | BOOL | INTERLOCK_ERROR:<br>Interlock error (any step) | - | x |
| FAULT | BOOL | SUPERVISION_FAULT:<br>Supervision error (any step) | - | x |
| ERR_FLT | BOOL | IL_ERROR_OR_SV_FAULT:<br>General fault | x | x |

| Parameters | Data type | Description | Standard parameter set | Maximum parameter set |
|---|---|---|---|---|
| SQ_ISOFF | BOOL | SEQUENCE_IS_OFF:<br>Sequencer is switched off (no step is active) | - | x |
| SQ_HALTED | BOOL | SEQUENCE_IS_HALTED:<br>Sequencer is stopped | - | x |
| TM_HALTED | BOOL | TIMES_ARE_HALTED:<br>Timers are stopped | - | x |
| OP_ZEROED | BOOL | OPERANDS_ARE_ZEROED:<br>Operands are reset | - | x |
| IL_ENABLED | BOOL | INTERLOCK_IS_ENABLED:<br>Interlock is taken into consideration | - | x |
| SV_ENABLED | BOOL | SUPERVISION_IS_ENABLED:<br>Supervision is taken into consideration | - | x |
| ACKREQ_ENABLED | BOOL | ACKNOWLEDGE_REQUIRED_IS_ENABLED:<br>Mandatory acknowledgement is enabled | - | x |
| SSKIP_ENABLED | BOOL | STEP_SKIPPING_IS_ENABLED:<br>Step skipping is enabled | - | x |
| SACT_DISP | BOOL | ACTIVE_STEPS_WERE_DISPLAYED:<br>Only active steps are displayed in "S_NO" | - | x |
| SEF_DISP | BOOL | STEPS_WITH_ERROR_FAULT_WERE_DISPLAYED:<br>Only steps with errors and faults are displayed in "S_NO" | - | x |
| SALL_DISP | BOOL | ALL_STEPS_WERE_DISPLAYED:<br>All steps are displayed in "S_NO" | - | x |
| AUTO_ON | BOOL | AUTOMATIC_IS_ON:<br>Display of automatic mode | x | x |
| TAP_ON | BOOL | T_AND_PUSH_IS_ON:<br>Display of semi-automatic mode | x | x |
| TOP_ON | BOOL | T_OR_PUSH_IS_ON:<br>Display of semi-automatic mode | x | x |
| MAN_ON | BOOL | MANUAL_IS_ON:<br>Display of manual mode | x | x |