LAB University of Applied Sciences

Toimilaitelohkojen ohjaus TwinCAT-ympäristössä

LAB-ammattikorkeakoulu
Insinööri (AMK)
2023
Aku Kärkkäinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Aku Kärkkäinen	Opinnäytetyö, AMK	Toukokuu 2023
	Sivumäärä 37	

Työn nimi

Toimilaitelohkojen ohjaus TwinCAT-ympäristössä

Tutkinto ja koulutusala

Insinööri (AMK), konetekniikan koulutus

Toimeksiantajaorganisaatio

Esys Oy

Tiivistelmä

Työn aiheena oli tutkia miten Siemensin TIA Portal -ohjelmointiympäristössä toteutetut, yrityksen sisällä standardoidut, toimilaitelohkot ja niiden ohjaus tulisi toteuttaa Beckhoffin TwinCAT-ohjelmointiympäristössä. Beckhoffin logiikoita sisältävien projektien määrä on kasvussa toimeksiantoyrityksessä ja osaamista niiden ohjelmointiin halutaan kehittää. Toimeksiantaja on Esys, Hollolassa sijaitseva vuonna 2003 perustettu automaatioalan yritys. Työn tavoitteena on löytää hyvä ja toimiva tapa toteuttaa toimilaitelohkot ja niiden ohjaus sekä tehdä niistä esimerkit.

Työ toteutettiin selvittämällä, mitkä osuudet pystyttiin ohjelmoimaan samalla tavalla kuin TIA Portalissa ja toteuttamalla mahdollisimman toimiva ratkaisu niihin osuuksiin, jotka piti toteuttaa eri lailla. Työn teoriaosuuden aineisto koostuu suurelta osin Siemensin ja Beckhoffin omista materiaaleista.

Tuloksena saatiin toimiva ratkaisu toimilaitelohkojen ohjelmointiin ja niiden ohjaukseen. Ratkaisusta tehtiin esimerkki, ja sitä testattiin fyysisellä toimilaitteella onnistuneesti. Työ kehitti myös tekijän ymmärrystä ja osaamista toimilaitteiden ohjauksesta sekä TwinCAT-ohjelmointympäristössä ohjelmoinnista. Tavoitteiden saavuttamista mitattiin säännöllisillä palavereilla opinnäytetyön ohjaajan kanssa sekä suullisilla keskusteluilla toimeksiantajan kanssa.

Toimilaitelohkojen ja niiden ohjauksen standardointi yrityksen sisällä helpottaa työntekijöiden työtaakkaa sekä mahdollistaa tehokkaamman ajankäytön projektien toteutusvaiheessa. Tulevaisuudessa TwinCATille voisi toteuttaa vastaavanlaisen automaattisen projektipohjan luojan, joka Esysillä on käytössä TIA Portalissa.

Asiasanat

TwinCAT, Toimilaitelohko, Beckhoff, PLC-ohjelmointi, Automaatio

Abstract

Author(s)	Type of Publication	Published
Aku Kärkkäinen	Thesis, UAS	May 2023
	Number of Pages 37	

Title of Publication

Control method for device control blocks in TwinCAT platform

Degree, Field of Study

Engineer (UAS), Mechanical Engineering

Organisation of the client

Esys Oy

Abstract

The topic of the thesis was to study how the standardized device control blocks, which were programmed in Siemens TIA Portal, utilized within the company and the method for controlling them could be implemented in the Beckhoff TwinCAT programming platform. The number of projects using Beckhoff hardware is on the rise and the client company wants to develop programming knowledge for Beckhoff software. The client is Esys, an automation company founded in 2003 and based in Hollola. The aim is to find a suitable solution for implementing the device control blocks and the method for controlling them as well as program a working example of both.

The thesis was conducted by determining which parts could be programmed in an equivalent way with TIA Portal and finding a working solution for the parts that needed different solutions. The materials for the theory section were mainly sourced from Siemens and Beckhoff.

The achieved result was a working solution for programming the device control blocks and the method for controlling them. An example was programmed and tested successfully with a physical device. In addition to improving the author's knowledge and understanding about device control, the thesis also developed the author's programming skills in the TwinCAT programming platform. The achievement of the objectives was monitored with regular meetings with the thesis supervisor and discussion with the client.

The standardization of device control blocks and their control method reduces the workload of employees and enables more efficient time management in the implementation phase of projects. In the future, a software solution which automatically creates project templates, similar to what Esys uses with TIA Portal, could be developed for TwinCAT.

Keywords

PLC programming, Automation, Device control block, TwinCAT, Beckhoff

Sisällys

1 Johdanto ... 1
2 Ohjelmointiympäristöt ja -kielet ... 3
2.1 Siemens TIA Portal ... 3
2.2 Beckhoff TwinCAT ... 3
2.3 Ohjelmointikielet ... 4
2.3.1 Instruction list (IL) ... 4
2.3.2 Structured text (ST) ... 5
2.3.3 Ladder diagram (LD) ... 6
2.3.4 Function block diagram (FBD) ... 7
2.3.5 Sequential function chart (SFC) ... 8
3 PLC-ohjelmointi ... 9
3.1 PLC-ohjelmointi TIA Portalilla ... 9
3.2 PLC ohjelmointi TwinCAT 3 -ohjelmistolla ... 11
4 Toimilaitelohkot ... 14
4.1 Toimilaitelohkot yleisesti ... 14
4.2 Toimilaitelohkot TIA Portalissa ... 15
4.3 Toimilaitelohkot TwinCATissa ... 16
4.4 Toimilaitelohkot Esysillä ... 17
5 Toimilaitelohkojen ohjelmointi TwinCATilla ... 21
5.1 Toimilaitelohkot ... 21
5.2 Yhteys HMI:hin ... 25
5.3 Toimilaitelohkokirjasto ... 29
6 Toimilaitelohkojen testaus ... 32
7 Yhteenveto ja pohdinta ... 35
Lähteet ... 36

1 Johdanto

Automaatio on teknologiaa, joka aktiivisesti valitsee dataa, muuntaa tietoa, tekee päätöksiä tai kontrolloi prosesseja (Lee & See 2004, 1). Sen tavoitteena on siis muokata jokin prosessi tai tuotantovaihe toimimaan mahdollisimman itsenäisesti. Automaation etuja voivat olla esimerkiksi alhaisemmat tuotantokustannukset, nopeampi tuotanto ja tarkempi työnjälki (Dey & Sen 2020, 5–6). Automaatio koostuu kolmesta osa-alueesta: käyttövoimasta, jonka avulla prosessia ja automaatiojärjestelmiä ajetaan, ohjelmasta, joka sisältää ohjeet prosessin ohjaamiseen ja ohjausjärjestelmästä, joka toteuttaa nuo ohjeet (Groover 2013, 73). Nykyaikana ohjausjärjestelmänä toimii ohjelmoitavat logiikkatietokoneet eli PLC:t. PLC voidaan määritellä mikrotietokonepohjaisena ohjauslaitteena, joka käyttää ohjelmoitavaan muistiin tallennettuja ohjeita toteuttaakseen logiikkaa, sekvensointia, ajastusta, laskemista ja aritmeettisia funktioita, digitaalisen tai analogisen rajapintamoduulin kautta laitteiden prosessien ohjaamiseen (Groover 2013, 267). PLC:n ohjelmoinnissa sen muistiin tallennetaan ohjeet prosessin suorittamiseen logiikkaohjelmointikielillä. PLC-ohjauksen rinnalle on kuitenkin tietotekniikan kehittyessä noussut myös tavallisiin tietokoneisiin eli PC:ihin perustuva ohjaus (Groover 2013, 274). PC:tä voi ohjelmoida hyvin samaan tapaan kuin PLC:tä, ja lisäksi PC tarjoaa mahdollisuuden käyttää tietokoneohjelmointikieliä ja muita PC-ohjelmistoja, joita PLC:t eivät tue.

Beckhoffin PC-pohjaisen logiikkaohjauksen käyttö toimeksiantajayritys Esysillä kasvaa, ja siihen halutaan panostaa. PLC-pohjainen Siemens on laajemmalti käytössä ja sen ohjelmoinnin kehittämiseen on panostettu huomattavasti enemmän. Yksi näistä panostuksen kohteista on erilaisten toimilaitteiden ohjaukseen tarkoitetut toimilaitelohkot. Siemensin ohjelmistolla ohjelmoidut, ja yrityksen sisällä standardoidut toimilaitelohkot halutaan nyt tuoda Beckhoffin TwinCAT-ohjelmistoon. Tämän opinnäytetyön tavoitteena on tutkia, miten toimilaitelohkojen ohjelmointi tapahtuu TwinCATissa sekä suunnitella ratkaisu toimilaitelohkojen yhdistämiseen human-machine interface- eli HMI-rajapintaan, jonka kautta toimilaitelohkojen manuaaliohjaus tapahtuu. Tavoitteena on myös toteuttaa toimivat esimerkit toimilaitelohkoista sekä niiden ohjauksesta, jotka olisivat mahdollisimman lähellä TIA Portalin toteutustapaa. Työssä myös käydään läpi, miten ohjelmointi tapahtuu Siemensin ja Beckhoffin ohjelmistoilla, ja kuinka Siemensin standardoidut toimilaitelohkot ja niiden ohjaus toimivat.

Työn toimeksiantaja on Hollolassa sijaitseva Esys Oy. Esys on perustettu vuonna 2003 ja se työllistää 45 työntekijää. Esysin toimipiste Hollolassa koostuu toimiston lisäksi asennushalleista ja varastosta. Yrityksen liikevaihto on n. 12–15 miljoonaa euroa. (Esys)

Esys on sähköautomaatioon keskittynyt insinööritoimisto, joka tarjoaa suunnittelu-, asennus- ja käyttöönottopalveluja. Myös sähkökeskusten valmistaminen, kokonaisten automaatioprojektien toteuttaminen ja sähkökomponenttien myynti kuuluvat yrityksen tarjontaan. Esysin asiakkaita ovat lukuisat yritykset eri toimialoilla, kuten elintarvike-, kaivos ja puunjalostusteollisuudessa. Isoja asiakkaita ovat muun muassa Fazer, Raute ja Metso Outotec. Näistä Raute ja Metso Outotec ovat sopimusasiakkaita mikä tarkoittaa muun muassa sitä, että Esysillä on työntekijöitä kyseisten yritysten tiloissa. Esys on myös Siemens-järjestelmäpartneri. Siemens-partnerit ovat alansa huippuosaajia, joiden osaamista Siemens ylläpitää ja kehittää. (Esys)

2 Ohjelmointiympäristöt ja -kielet

2.1 Siemens TIA Portal

Siemens TIA Portal on automaatioympäristö, joka yhdistää useita Siemensin automaatio-ohjelmia yhden ohjelmiston alle. Aikaisemmin esimerkiksi PLC-koodin ohjelmoimiseen oli käytettävä Step 7-ohjelmaa ja käyttöliittymien tuottamiseen WinCC-ohjelmaa. Nykyään molemmat löytyvät TIA Portalin alta. Lisäksi saatavilla on ajureiden ohjauksen mahdollistava SINAMICS Startdrive ja moottoriohjaukseen tarkoitettu SIMOCODE ES. Myös SIMOTION, pakettiratkaisu liikeohjaukseen ja logiikkaan on integroitu TIA Portaliin. TIA Portal on siis suunniteltu sisältämään kaikki tärkeimmät automaatioprojektin osa-alueet. Ohjauksen ja käyttöliittymän lisäksi ohjelmalla voi ohjata hajautettuja kenttälaitteita, säädellä moottoreita sekä hallita liikkeenohjausta ja energiahallintaa. Lähtökohtaisesti koko Siemensin laitekantaa on mahdollista ohjata saman ohjelman sisällä. Ohjelmisto mahdollistaa myös usean käyttäjän samanaikaisen työskentelyn samassa projektissa. TIA Portalista tulee saataville uusia versioita aika ajoin, mutta perustaltaan ne ovat samanlaisia. Isoin uudistus tuli versiossa V16, kun Siemens julkisti SIMATIC WinCC Unified-visualisointijärjestelmän. (Siemens.)

TIA Portalin ohjelmaeditori graafisille kielille on monipuolinen. Kaikkia objekteja pystyy esimerkiksi raahaamaan paikasta toiseen, ja maalaamalla halutut objektit on mahdollista kopioida, ja liittää ne haluttuun paikkaan. Editorissa on suuri määrä erilaisia operaattoreja esimerkiksi laskutoimituksiin. Koodin osuudet eli networkit pystyy sulkemaan niin, että vain niiden otsikot näkyvät, mikä helpottaa halutun koodiosuuden etsimistä isommissakin projekteissa. Editorin yläosaan voi myös lisätä pysyvästi näkyvillä olevan työkalurivin valitsemistaan operaattoreista nopeampaa käyttöä varten. Lisäksi vaikka jokin lohko on tehty graafisella kielellä, pystyy lohkoon kirjoittamaan koodia myös tekstipohjaisella kielellä. Esimerkiksi ladderilla kirjoitettuun lohkoon voi lisätä joko ST- tai IL-kielellä kirjoitettua koodia.

2.2 Beckhoff TwinCAT

Beckhoff TwinCATin uusin versio, TwinCAT 3, pohjautuu Microsoft Visual Studioon, mikä mahdollistaa tavanomaisten logiikkaohjelmointikielten lisäksi C++, C ja MATLAB kielen käytön. Sen rakenne on modulaarinen, eli se koostuu useista eri osista, joita käyttäjä voi ladata tarpeen mukaan. Tärkein niistä on TwinCAT XAE (eXtended Automation Engineering), joka on Windows pohjaisella tietokoneella käytettävä suunnitteluympäristö, joka sisältää työkalut laitteiston ohjelmointiin ja konfigurointiin sekä mahdolliset laajennusosat, kuten HMI-engineering. Suunnitteluympäristön käyttäminen ei vaadi lisenssiä, mikä helpottaa uusien käyttäjien, kuten opiskelijoiden, tutustumista ohjelmistoon. Toinen tärkeä osa TwinCAT

ympäristöä on TwinCAT XAR (eXtended Automation Runtime), joka on teollisuustietokoneella suoritettava reaaliaikainen runtime. Sen käyttöjärjestelmänä voi olla Windows tai Beckhoffin oma TwinCAT/BSD. (Beckhoff 2023)

TwinCATin ohjelmaeditori graafisille kielille on huomattavasti rajoittuneempi verrattuna TIA Portaliin. Objektien raahaus ei ole mahdollista, ja niiden kopioiminen on rajatumpaa. Myöskään koodiosuuksien sulkeminen ei ole graafisten kielien kanssa mahdollista. Operaattoreita on huomattavasti vähemmän, eikä niille ole pysyvästi näkyvissä olevaa suosikkityökaluriviä. Ohjelmaeditorin asetuksista pystyy lisäämään kuitenkin pikanäppäinyhdistelmiä eri operaattoreille, ja operaattorien vähäisen määrän vuoksi varsinaisesta työkalulaatikosta on suhteellisen helppo löytää tarvittavat operaattorit.

2.3 Ohjelmointikielet

Molemmat automaatioympäristöt noudattavat IEC 61131-3- standardia (Siemens 2013; Beckhoff 2023). Standardin mukaan saatavilla pitää olla kaksi tekstipohjaista ja kolme graafista ohjelmointikieltä. Standardi määrittelee myös näiden kielten muotorakennetta ja semantiikkaa. (IEC 61131-3 2013, 14)

TIA Portalissa ohjelmointikielet ovat hieman eri nimisiä kuin standardissa. Alla olevassa taulukossa 1. näkyy miten kielet vastaavat toisiaan.

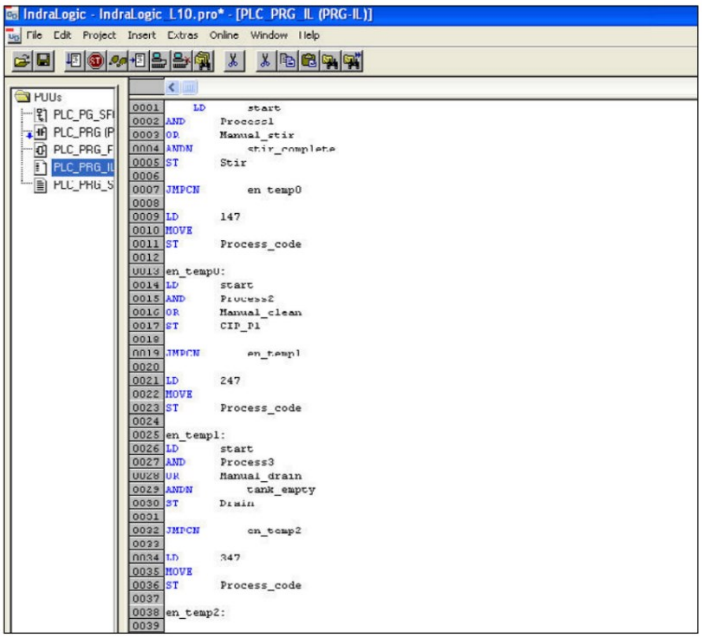
STEP 7 (TIA Portal)	IEC 61131-3 languages
Instruction List (AWL/STL)	Corresponds to IEC 61131-3 language "AWL/IL"
Ladder Logic KOP/LAD	Corresponds to IEC 61131-3 language "KOP/LD"
Function Block Diagram FUP/FBD	Corresponds to IEC 61131-3 language "FUP/FBD"
Structured Control Language SCL	Corresponds to IEC 61131-3 language "ST"*
S7-GRAPH	Corresponds to IEC 61131-3 language "AS/SFC"*

Taulukko 1. IEC 61131-3-standardin kielien vastaavuudet TIA Portalissa (Siemens 2013)

2.3.1 Instruction list (IL)

Instruction list eli IL on matalan tason ohjelmointikieli, ja se on yksi vanhimmista IEC 61131-3 standardin kielistä. Se muistuttaa rakenteeltaan jonkin verran symbolista konekieltä. IL koostuu useista riveistä koodia, joista jokainen edustaa yhtä toimintoa. Sen etuja ovat nopea prosessointi, vähäinen muistin tarve ja kompakti muotorakenne. Lisäksi IEC 61131-3-standardin mukaisten toimintojen avulla tehtyä koodia voi siirtää suoraan sellaisenaan eri laitteistoalustoille. IL on yleisemmin käytössä Euroopassa, toisin kuin Yhdysvalloissa, jossa ohjelmoijat suosivat enemmän graafisia kieliä. (Bosch Rexroth Corporation 2009.)

Tekstipohjaisuus tekee kielestä kuitenkin graafisia kieliä vaikeammin seurattavan erityisesti huoltotilanteissa. Nykyään PLC:t ovat myös varustettuja niin isoilla muisteilla ja nopeilla prosessoreilla, että IL-kielen ohjelmoinnin edut jäävät melko vähäisiksi. Uusin IEC 61131-3 standardin versio, eli 2013 vuonna julkaistu 3. versio merkitsikin IL-kielen vanhentuneeksi eli sitä ei sisällytetä seuraavaan standardin versioon (PLCopen). Esimerkki IL-kielestä on nähtävissä kuvassa 1. (Bosch Rexroth Corporation 2009.)



Kuva 1. Esimerkki IL-kielen rakenteesta (Bosch Rexroth Corporation 2009)

2.3.2 Structured text (ST)

Structured text on IL-kielen tavoin tekstipohjainen. Se muistuttaa jonkin verran esimerkiksi tietokoneohjelmoinnissa käytettävää C-kieltä, joten tietokoneohjelmointitaustasta voi siis olla huomattavasti hyötyä ST-kielen käytössä. Kielen avulla on helppo rakentaa tiivistä koodia monimutkaisiinkin prosesseihin muun muassa toistorakenteiden ja osoittimien avulla. IL-kielen kaltaisesti prosessorit suorittavat ST pohjaisia ohjelmia nopeammin kuin graafisia ohjelmia. Myös koodin siirrettävyys eri laitteistoalustoiden välillä on helppoa. (Bosch Rexroth Corporation 2009.)

ST-kielellä on kuitenkin myös haittapuolia. Virheiden löytäminen tekstipohjaisesta koodista on jo itsessään vaikeampaa kuin graafisista kielistä, ja koodin ollessa mahdollisimman tiivistä se vaikeutuu entisestään. Tekstipohjainen koodi on myös sitä osaamattomille vaikeampi tulkita kuin graafiset kielet. ST-kieltä käytetäänkin usein yhdessä graafisten kielten kanssa. Tällöin esimerkiksi ohjelman sekvenssi on tehty graafisella kielellä, ja erilaisia funktioita ja muita toimintoja on tehty ST-kielellä. Näin ohjelman toimintaa on helpompi seurata,

mutta samalla pystytään hyödyntämään ST-kielen etuja. IEC 61131-3-standardi mahdollistaa myös omien komentojen toteuttamisen yhdellä kielellä, joita voidaan kutsua muilla kielillä. Yleinen tapa on tehdä funktio ST-kielellä, ja kutsua sitä LD-kiellä tehdyssä ohjelmassa. Tällöin pitää kuitenkin olla tarkkana, että funktio on toteutettu oikein, sillä muut kuin sen tekijä eivät pääse sitä muokkaamaan. Esimerkki ST-kielestä on nähtävissä kuvassa 2. (Bosch Rexroth Corporation 2009.)

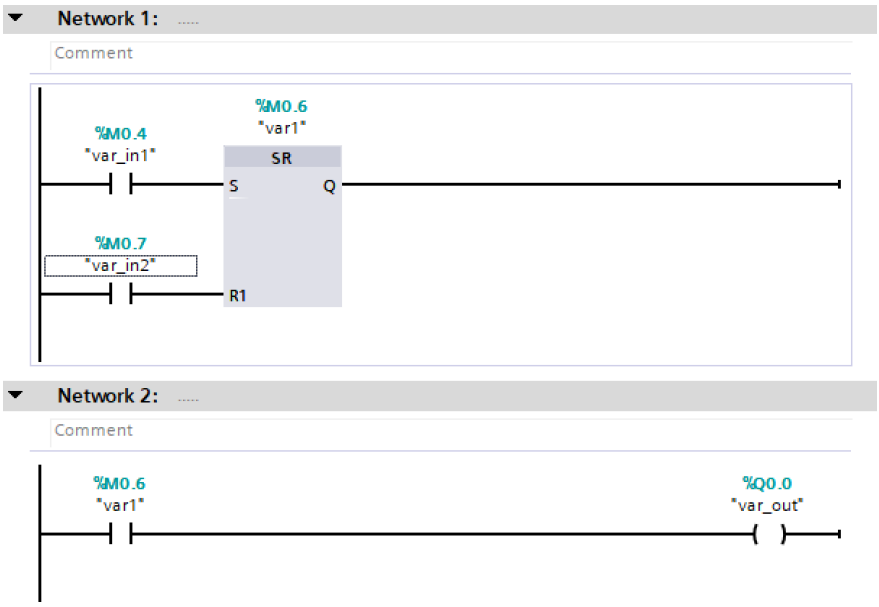
```
Network 3:
Comment
1 IF "var_in" = TRUE THEN
2     "var_out" := TRUE
3     ;
4 ELSIF "var_in" = False THEN
5         "var_out" := FALSE
6         ;
7 END_IF;
8
9
```

Kuva 2. Esimerkki ST-kielen rakenteesta

2.3.3 Ladder diagram (LD)

Ladder diagram, yleisemmin ladder, on graafinen kieli, joka koostuu eräänlaisista ohjausvirtapiireistä eli networkeista. Networkeissa olevilla kytkimillä, keloilla ja muilla operaattoreilla ohjataan lähtöjä tai muita toimintoja. Ladder on PLC-koodauksen kielistä todennäköisesti käytetyin ja IL-kielen ohella yksi vanhimmista. Kieltä osaa siis valtaosa PLC-koodin ohjelmoijista ja jopa henkilöt, joilla ei ole ohjelmointitaustaa voivat ymmärtää sitä jossain määrin, ainakin paremmin kuin tekstipohjaisia kieliä. Helppolukuisuus mahdollistaa myös helpon virheenetsinnän. Ladder sopii parhaiten yksinkertaisen ohjelman kirjoittamiseen yhdestä tuotantosovelluksesta, esimerkiksi tavarankäsittelystä. Ohjelmassa tarvitsee tällöin vain esimerkiksi sensorien tulojen tarkkailua ja lähtöjen ohjaamista tulojen avulla. Seassa voi olla esimerkiksi ajastimia ja laskureita, mutta ohjelma ei vaadi mitään monimutkaisempia laskutoimituksia. (Bosch Rexroth Corporation 2009.)

Ladderin huonot puolet tulevat esiin laajempia ohjelmia tehdessä. Laajat ohjelmat voivat muuttua vaikeasti luettaviksi ja tulkittaviksi, jos niiden toimintaa ei ole kattavasti selitetty koodin edetessä. Suurten prosessikokonaisuuksien toteuttaminen voi olla hankalaa koska useassa eri prosessin vaiheessa käytettäviä lähtöjä ohjaavilla piireillä on taipumus kasvaa todella suuriksi. Myös yleisesti tarvittavia toimintoja, kuten PID, trigonometria ja data-analyysi on vaikea sisällyttää ladder ohjelmaan. Esimerkki ladderista on nähtävissä kuvassa 3. (Bosch Rexroth Corporation 2009.)

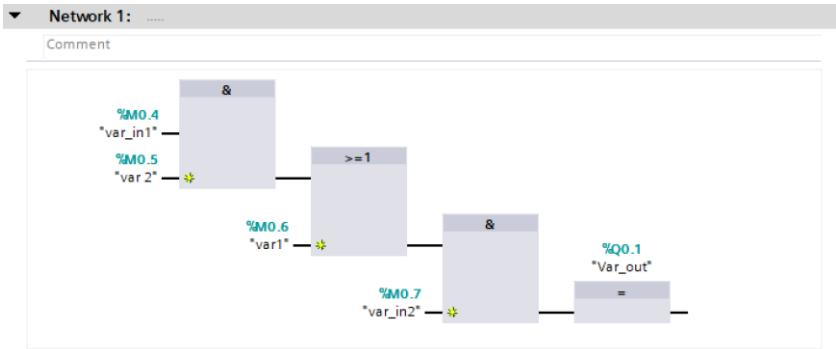


Kuva 3. Esimerkki ladderin rakenteesta

2.3.4 Function block diagram (FBD)

Function block diagram eli FBD, koostuu samantyylisistä virtapiireistä, kuin ladder. Komennot ovat myös samoja, mutta ne toteutetaan erilaisilla toimintolaatikoilla, kuten boolen algebran JA-, TAI- ja EI-operaattoreilla. FBD-kieltä on erittäin helppo seurata ja oppia ilman osaamista relelogiikasta. Kuten ladder se soveltuu parhaiten yksikertaisiin ohjelmiin, joissa esimerkiksi sensoritulot ohjaavat moottorilähtöjä. (Bosch Rexroth Corporation 2009.)

FBD-kielen heikkoudet ovat isoissa ohjelmissa, joissa tarvitaan erikoisempia tuloja, lähtöjä ja funktioita. Toimintolaatikot vievät paljon ruutualaa, ja projektista voi tulla hankalasti käsiteltävä, mikäli se kasvaa liian isoksi. FBD-kielen kanssa pitää myös käyttää enemmän aikaa suunnitteluun ennen ohjelman tekoa, jotta ymmärtää ohjelmaa ja sen kulkua, sillä muutoksien tekeminen jälkikäteen voi olla vaikeaa. Esimerkki FBD-kielestä on nähtävissä kuvassa 4. (Bosch Rexroth Corporation 2009.)

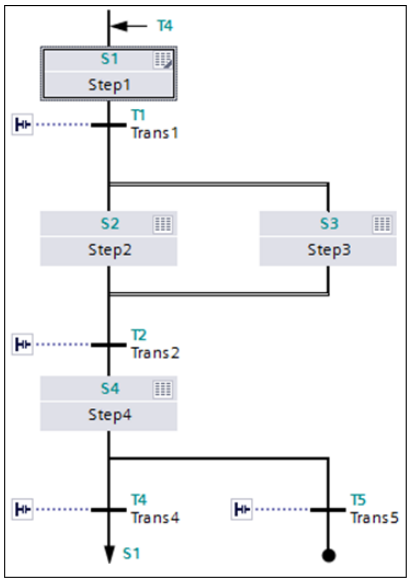


Kuva 4. Esimerkki FBD-kielen rakenteesta

2.3.5 Sequential function chart (SFC)

Sequential function chart eli SFC, koostuu toimintolaatikoista ja siirtymistä. Toimintolaatikoiden toiminnan voi määritellä millä tahansa ohjelmointikielellä. Siirtymäaskeliin asetetaan ehtoja, ja kun ehdot täyttyvät ohjelma siirtyy seuraavaan askeleeseen. SFC on hyvä valinta esimerkiksi paljon toistoa vaativiin poiminta- ja sijoitustehtäviin. Virheenetsintä on erittäin helppoa kielen helpon seurattavuuden ja selkeän askelluksen ansiosta. (Bosch Rexroth Corporation 2009.)

FBD-kielen tapaan SFC vaatii paljon suunnittelutyötä etukäteen, muuten ohjelmasta voi tulla vaikeaselkoinen. SFC ei myöskään sovi kaikkiin sovelluksiin, sillä sen muotorakenne voi tuoda ohjelmaan tarpeetonta monimutkaisuutta. Graafisena kielenä SFC-kielen suoritusnopeus on hidas, ja se on myös graafisista kielistä hitain suuren yleisrasitteen vuoksi. Muita kieliä pystyy suhteellisen helposti kääntämään toiselle kielelle, mutta SFC ei sitä mahdollista. Esimerkki SFC-kielestä on nähtävissä kuvassa 5. (Bosch Rexroth Corporation 2009.)



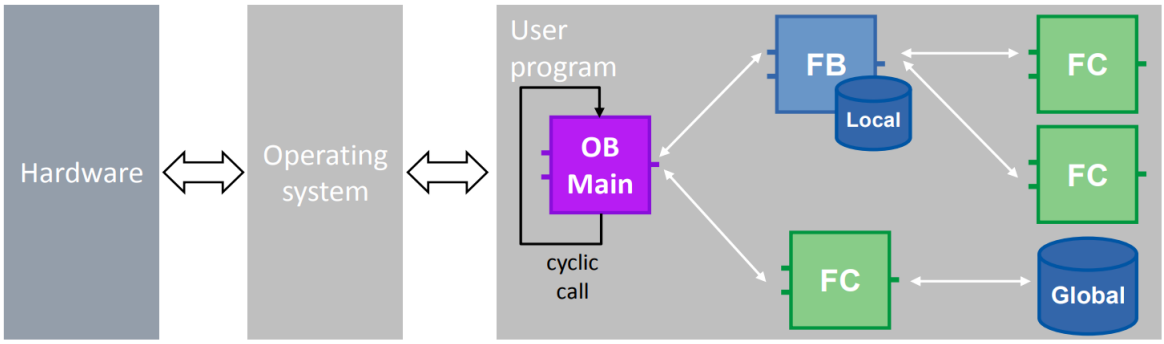
Kuva 5. Esimerkki SFC-kielen rakenteesta (Siemens 2018a)

3 PLC-ohjelmointi

3.1 PLC-ohjelmointi TIA Portalilla

Logiikkaohjaus pohjautuu laitteistoon, jonka resursseja käyttää käyttöjärjestelmä ja sen sisällä ohjelmakoodi. Siemensin laitteistona toimii aina PLC. Varsinaisen ohjelman muodostavat organisaatioyksiköt eli ohjelmalohkot. Näitä lohkoja määritellään IEC 61131-3 standardissa kolme kappaletta: program, function block eli funktiolohko ja function eli funktio (IEC 61131-3 2013, 45).

TIA Portalissa ohjelmalohkot eroavat hieman ohjelmointikielten kaltaisesti standardin nimeämistyylistä. Ohjelmalohkot TIA Portalissa ovat: organization block eli organisointilohko, funktio, data block eli datalohko sekä funktiolohko. Organisointilohko on ohjelman päälohko, joka toimii rajapintana ohjelman ja käyttöjärjestelmän välillä. Se myös kutsuu kaikki muut lohkot, jotka halutaan ohjelmakiertoon mukaan. Funktiot ovat lohkoja, joilla ei ole syklistä datavarastoa, eli niiden sisältämä tieto ei säily ohjelmakierrosta toiseen. Funktiolohko sen sijaan säilyttää tietonsa ohjelmakierrosta toiseen instanssidatalohkossa. Instanssidatalohkon sisältö voidaan määritellä ainoastaan funktiolohkossa, ja se luodaan automaattisesti funktiolohkon yhteydessä. Funktiolohkolle on myös mahdollista tehdä moni-instanssinen tietovarasto. Tällöin Instanssidatalohkoa ei luoda, vaan funktiolohko liitetään johonkin olemassa olevaan muistiin, kuten funktiolohkoa kutsuvan toisen funktiolohkon instanssidatalohkoon. Datalohko on varasto muuttujille, joka voi olla yleinen datalohko, jossa muuttujat ovat koko ohjelman käytössä tai funktiolohkon instanssidatalohko. Lohkojen ja muiden järjestelmän osien välisiä yhteyksiä kuvataan kuvassa 6. (Siemens 2018b, 42–47.)



Kuva 6. Logiikkaohjelmoinnin osien välisiä yhteyksiä (Siemens 2018b, 47)

TIA Portalissa on suuri määrä erilaisia tietotyyppejä, joista muutamia yleisimpiä ovat esimerkiksi Bool, Int, Real ja Word. Bool on yhden bitin kokoinen ja sillä on kaksi tilaa: true ja false. Int tietotyyppiä käytetään kokonaislukujen ilmoittamiseen ja se on kooltaan 16 bittiä eli 2 tavua. Real on liukulukujen tietotyyppi ja se vie tilaa 32 bittiä eli 4 tavua. Word koostuu

sarjasta bittejä, ja se vie 16 bittiä eli kaksi tavua tilaa. Monista tietotyypeistä löytyy kooltaan suurempi tai pienempi versio erilaisilla etuliitteillä, joita ovat esimerkiksi short, double ja long. Double kaksinkertaistaa tietotyyppien, kuten Word ja Int, koon neljään tavuun. Short sen sijaan puolittaa Int tietotyypin yhden tavun kokoiseksi. Long etuliitettä käytettäessä Real tietotyypin koko tuplaantuu kahdeksaan tavuun ja Int tietotyyppi kasvaa nelinkertaiseksi kahdeksaan tavuun.

Ohjelmalohkot TIA Portalissa voivat olla optimoituja tai optimoimattomia. Suurin ero käyttäjän kannalta on se, että optimoidun lohkon muuttujiin viittaaminen on symbolista, eli muuttujiin viitataan niiden nimillä. Optimoimattoman lohkon muuttujiin viitataan absoluuttisesti, eli niiden osoitteen perusteella. Tarkka osoite näkyy lohkoissa nimellä offset. Offset koostuu kahdesta pisteen erottamasta luvusta, joista ensimmäinen kuvaa tavunumeroa ja toinen numero tavun yksittäistä bittiä. Offset alkaa luvusta 0.0 ensimmäisen lohkon muuttujan kohdalla, ja kasvaa sen mukaan kuinka suuri muuttujan tietotyyppi on. Bittien kohdalla offset kasvaa tasaisesti bitti kerrallaan, mutta esimerkiksi kuvassa 10 näkyvän D2_Service_Switch-muuttujan jälkeen offset hyppää seuraavaan kokonaiseen tavuun, koska seuraava muuttuja on tietotyyppiä int, joka vie kaksi tavua tilaa, eikä se voi alkaa edellisen tavun sisältä. Väliaikaisilla muuttujilla offset arvo alkaa nollasta ja vakiomuuttujilla offset arvoa ei ole ollenkaan. (Siemens 2018b, 17.)

Valve

	Name	Data type	Offset
1	Input		
2	D1_Feedback	Bool	0.0
3	D1_Feedback_InUse	Bool	0.1
4	D2_Feedback	Bool	0.2
5	D2_Feedback_InUse	Bool	0.3
6	D1_Function_Allowed	Bool	0.4
7	D2_Function_Allowed	Bool	0.5
8	D1_Request	Bool	0.6
9	D2_Request	Bool	0.7
10	NOT_Allowed_Close	Bool	1.0
11	Impulse_Control	Bool	1.1
12	Service_Switch_InUse	Bool	1.2
13	D1_Service_Switch	Bool	1.3
14	D2_Service_Switch	Bool	1.4
15	Valve_Type	Int	2.0
16	Feedback_Delay	Real	4.0
17	Master_Reset	Bool	8.0
18	100 ms Pulse	Bool	8.1

Kuva 7. Offsetin määräytyminen

Lisäksi yksittäisten muuttujien muuttaminen retain-tyyppisiksi ei ole optimoimattomassa lohkossa mahdollista. Eli jos yhdenkin muuttujan haluaa asettaa retain-muuttujaksi, niin kaikki lohkon muuttujat on tällöin muutettava retain-muuttujiksi, mikä johtaa turhaan pysyvän muistin käyttöön (Siemens 2018b, 13–16). Optimoidussa lohkossa samaa ongelmaa ei ole. Retain-muuttujia käytetään silloin kun halutaan muuttujan arvon säilyvän, vaikka järjestelmä

sammuu. Virtakatkoksen tapahtuessa retain-muuttujat kirjoitetaan non-volatile muistiin eli pysyvään muistiin (Siemens 2018b, 64). Muistialueen koko vaihtelee prosessorin mukaan niin, että uudemmissa prosessoreissa on pääsääntöisesti enemmän pysyvää muistia. Esimerkiksi S7-1200-sarjan prosessoreissa pysyvää muistia on vain 10 kilotavua, kun uudemmissa S7-1500-sarjan prosessoreissa muistin koko vaihtelee 88 kilotavusta 768 kilotavuun (Siemens 2018b, 64). Optimoidun datalohkon sisältö järjestellään niin, että suuremmat muuttujat, kuten sanat, ovat lohkon muistissa alkupuolella ja yksittäiset bitit lopussa. Lohkon tavujärjestys kääntyy optimoidessa big endian -järjestyksestä little endian -järjestykseen. Big endian tarkoittaa sitä, että ensimmäiset tavut ovat merkitsevimpiä ja little endian päinvastaista, eli merkitsevimmät tavut ovat viimeisenä. Optimoitujen ja optimoimattomien lohkojen tavujärjestyksen ero johtuu siitä, että vanhemmat logiikat, kuten S7-300/400, lukivat muistia big endian -järjestyksellä, kun taas uudemmat logiikat S7-1500-sarjasta käyttävät little endian -järjestystä. Uusien logiikoiden on siis helpompi ja nopeampi lukea optimoitua lohkoa sen oikean tavujärjestyksen ja järjestellyn sisällön ansiosta. (Siemens 2018b, 13–16.)

Tuotannonohjauksessa laitteiden käsiajot toteutetaan usein HMI-paneelien avulla. Paneelien graafinen käyttöliittymä yhdistetään ohjattavien laitteiden ohjelmalohkoihin HMI-tagien eli paneelin omien muuttujien avulla. Paneelissa ei voi siis suoraan käyttää muuttujia ohjelmalohkoista. HMI-tageja on kahden tyyppisiä: sisäisiä ja ulkoisia. Sisäisiä muuttujia voi käyttää ainoastaan paneelin käyttöliittymässä eikä niillä ole yhteyttä PLC:hen. Ulkoiset muuttujat taas yhdistetään vastaavanlaisiin muuttujiin PLC:n puolella ja ne voivat viitata PLC-muuttujiin absoluuttisesti tai symbolisesti. Ulkoisille HMI-tageille on valittavissa kolme erilaista päivitystilaa, jotka voi valita jokaiselle muuttujalle erikseen. Ensimmäinen tila päivittää HMI-tagia syklisesti tagin ollessa näkyvillä näytöllä ja toinen riippumatta siitä onko tagi näkyvillä vai ei. Kolmas tila ei päivitä HMI-tagia syklisesti, vaan ainoastaan manuaalisesti tagien päivittämiseen tarkoitetulla funktiolla tai skriptillä. Kahden ensimmäisen tilan päivityssykli voidaan myös määritellä jokaisen muuttujan kohdalla erikseen, ja se siis määrittää kuinka kauan HMI-tagin päivittymisessä kestää, kun siihen liitetyn PLC-tagin arvo muuttuu. Syklin pituudelle on ennalta määritellyt arvot välillä 100 millisekuntia ja yksi tunti.

3.2 PLC ohjelmointi TwinCAT 3 -ohjelmistolla

TwinCATin logiikkaohjaus koostuu samoista peruselementeistä kuin Siemensillä, missä laitteisto suorittaa logiikkaohjelmaa, joka koostuu ohjelmalohkoista. Beckhoffin laitteisto on kuitenkin PC-pohjaista, eli logiikkaohjelmaa suorittaa esimerkiksi Windows käyttöjärjestelmällä varustettu teollisuustietokone. Pääohjelmalohkot TwinCATissa ovat IEC IEC 61131-3 standardissa mainitut funktio, funktiolohko ja ohjelma. Niiden lisäksi TwinCATista löytyy

viisi muuta ohjelmointiobjektia: action ja transition sekä olio-ohjelmointiin tarkoitetut method, property ja interface. Funktiolohko ja funktio ovat verrattavissa TIA Portalin vastaaviin lohkoihin. Ohjelma-lohko on toimintaperiaatteeltaan samankaltainen kuin funktiolohko, sillä se palauttaa yhden tai useamman arvon suorituskerran jälkeen ja säilyttää arvonsa seuraavaan ohjelmakiertoon asti. Sillä ei kuitenkaan ole instansseja niin kuin funktiolohkoilla, eikä sitä käytettäessä tarvitse välttää ulkoisia muuttujia. Action-toiminnolla voi toteuttaa ylimääräistä koodia funktiolohkoon tai ohjelmaan. Toiminnolle ei voi kuitenkaan määritellä uusia muuttujia, vaan se käyttää muuttujia lohkosta, johon se on liitetty. Toiminnon ohjelmointikieli voi olla myös eri kuin päälohkossa. Transition-objektilla taas voi määrätä ehtoja, joiden täyttyessä siirrytään koodissa seuraavaan askeleeseen. (Beckhoff.)

TwinCATilla ohjelmoidessa tärkeä osa alue on projektipohjaan luotava System-konfiguraatio, joka sisältää projektin kannalta välttämättömiä asetuksia, kuten Lisenssihallinnan, real time -asetukset ja taskien hallinnan. Real time -asetuksissa määritellään reaaliaikaisen runtimen asetuksia, kuten muistinkäyttöä, prosessoriydinten allokointia, jatkuvan keskeytyksen aikasyklin määrittelyä ja taskien prioriteetteja. Taskien tehtävä on kutsua pääohjelmalohkoa syklisesti ja niille voidaan määritellä prioriteettien lisäksi keskeytysten määrä, joka vastaavasti määrittää ohjelman suoritussyklin pituuden. Taskien prioriteettien avulla on mahdollista suorittaa useampaa ohjelmaa samanaikaisesti, silloin kun ohjelmien suoritussyklien on oltava eripituisia. (Beckhoff.)

Yksi isoimmista eroista TIA Portaliin on datalohkojen puuttuminen. TwinCATissa muuttujia voi luoda ohjelmalohkojen sisällä tai GVL (global variable list) editorissa. Ohjelma suosittelee tähtimerkin käyttöä tarkan osoitteen sijasta, kun esimerkiksi lähtö- ja tulomuuttujille annetaan osoitteita. Tällöin ohjelma määrittelee muuttujalle optimaalisen osoitteen. TwinCATissa on kahdentyyppisiä muuttujia, jotka eivät menetä arvoaan PLC:n sammuessa. Molemmat muuttujatyypit vaativat pysyvää muistia toimiakseen. Persistent muuttujat kirjoitetaan pysyvälle muistialueelle vasta kun ohjelma sammuu, joten ne vaativat yleensä sopivan UPS (Uninterruptible Power Supply) -laitteen, joka takaa virransaannin ainakin hetkeksi virran katkeamisen jälkeen. Muuttujat voidaan myös kirjoittaa muistialueelle siihen tarkoitetulla funktiolohkolla, jolloin UPS-laitetta ei välttämättä tarvita. Retain-muuttujat sen sijaan kirjoitetaan pysyvälle muistialueelle joka ohjelmakierron lopussa. Persistent ja retain-muuttujat toimivat siis käytännössä samalla tavalla, mutta niiden kirjoitustavat pysyvälle muistialueelle eroavat hieman toisistaan. Molemmat muuttujatyypit voi tyhjentää komennolla reset origin, joka palauttaa kaikki PLC projektin muuttujat alkuarvoihinsa. (Beckhoff.)

TwinCATissa on mahdollista luoda käyttäjän määrittelemiä tietotyyppejä, joita käytetään monen toimilaitteen ohjauksessa. Ominaisuuden nimi on data unit type eli DUT. DUT

tietotyyppejä on neljä erilaista: structure, enumeration, alias, ja union. Structure eli struktuuri yhdistelee useita eri tietotyypin muuttujia yhdeksi rakenteeksi. Enumeration eli enumeraatio koostuu enumeraatiosuureista, joilla tarkoitetaan usein merkkijonoa, joka vastaa jotakin kokonaislukua. Koodiin ei siis kirjoiteta lukua, vaan sitä vastaava merkkijono, jolloin lukua voidaan muuttaa tarvittaessa muokkaamalla merkkijono vastaamaan jotain toista lukua. Aliaksen avulla voidaan luoda vaihtoehtoinen nimi jollekin muuttujalle. Union muistuttaa paljon struktuuria, mutta erona on se, että unionissa kaikki tietotyypit vievät saman verran muistia kuin sen isoin tietotyyppi. (Beckhoff.)

TwinCATissa on TIA Portalin lailla useita eri tietotyyppejä ja pääosin ne ovat samanlaisia. Eroja on lähinnä tietyyppien ko'oissa. Esimerkiksi BOOL tietotyyppi vie TwinCATissa 8 bittiä eli yhden tavun tilaa. Ainoa yhden bitin kokoinen tietyyppi on BIT, jota voi käyttää ainoastaan struktuureissa tai funktiolohkoissa, kun halutaan osoittaa yksittäisiin bitteihin. BIT tietotyypin käyttäminen säästää siis muistia verrattuna BOOL tietotyyppiin, mutta yksittäisten bittien käsittely on huomattavasti hitaampaa. (Beckhoff.)

TwinCATissa yhteys HMI-paneeliin tapahtuu Automation Device Specification (ADS) -protokollan kautta. Kun halutaan viedä muuttujia PLC:stä HMI-kehitysympäristöön, täytyy niiden välille luoda yhteys ADS:n avulla. Halutut PLC-muuttujat linkitetään map-komennolla, joka lisää muuttujat omaan kansioonsa HMI kehitysympäristössä.

4 Toimilaitelohkot

4.1 Toimilaitelohkot yleisesti

Toimilaitelohko tai laitelohko on ohjelmalohko, joka ohjaa erilaisia toimilaitteita. Toimilaite tarkoittaa jotakin yksittäistä liikkuvaa osaa, kuten hihnakuljettimen moottoria, tuotantolinjastolla. Laitteita on usein useita samanlaisia, jolloin jokaisen ohjaamiseen käy samanlainen laitelohko. Kun laitetta halutaan liikuttaa, annetaan laitelohkolle lupa käskeä laitetta suorittamaan haluttu liike sille määritettyjen parametrien mukaan. Laitelohkolle siis syötetään tuloarvoja, ja laitelohkon sisällä näitä arvoja käytetään laitteen ohjaukseen laitelohkon sisältämän koodin mukaisesti. Tuloarvoja voivat olla esimerkiksi sallinta, nopeus ja nollaus. Laitelohkosta annetaan ulos myös lähtöarvoja, joita voidaan vastaavasti lukea sekvenssissä esimerkiksi sen tarkistukseen, että laite liikkuu halutulla tavalla. Myös laitteen vikailmoituksia ilmaistaan usein laitelohkon lähtöarvoilla.

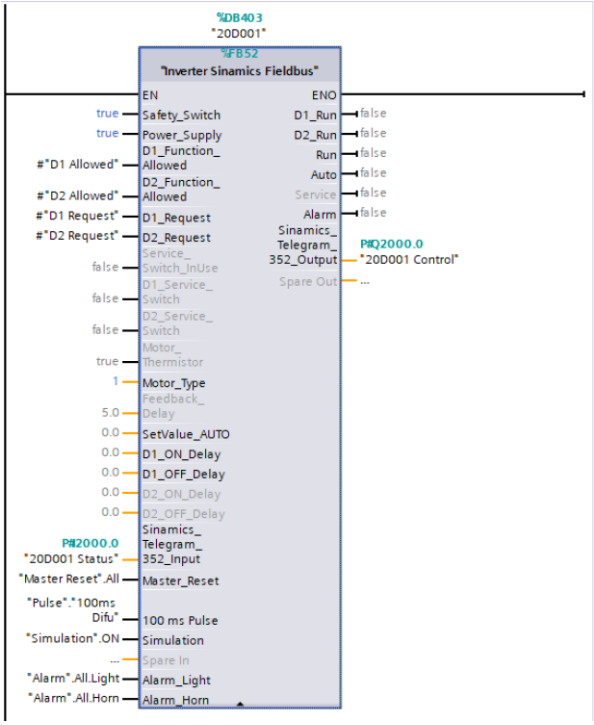
Laitelohkot toteutetaan yleensä funktiolohkoilla. Funktiolohkoja on kahdentyyppisiä: standardifunktiolohkot ja käyttäjän luomat funktiolohkot (Lydon 2018). Standardifunktiolohkot ovat ohjelmaeditoreista löytyviä valmiita lohkoja, jotka toteuttava esimerkiksi erilaisia laskutoimituksia. Käyttäjän luomat funktiolohkot sen sijaan ovat tyhjiä, ja käyttäjä voi itse ohjelmoida niiden toiminnallisuuden. Jatkossa nimellä funktiolohko tarkoitetaan yksinomaan käyttäjän luomia funktiolohkoja. Funktiolohkojen isoimpia etuja on uudelleen käytettävyys, eli kyky luoda useampia instansseja, jolloin samoille toimilaitteille voi tehdä useamman instanssin samasta toimilaitelohkosta. Funktiolohkon toimintaa määritellään tarkemmin IEC 61131-3 -standardissa (IEC 61131-3 2013, 66). Sen mukaan funktiolohko tuottaa suoritettaessa yhden tai useamman arvon. Funktiolohkoja voi olla useita instansseja eri nimellä. Jokaisella funktiolohkolla on oltava oma tunniste eli instanssin nimi ja muistialueensa, johon tallennetaan sen lähdöt ja sisäiset arvot. Tuloarvojen ja tarvittavien sisäisten arvojen tulee säilyä ohjelmakierron ajan, ja ainoastaan funktiolohkon tulo- ja lähtöarvot tulee olla saatavilla lohkon ulkopuolelta. Funktiolohkon muistiin ei saa viitata lohkon ulkopuolelta eikä lohkon sisältä saa vastaavasti viitata lohkon ulkopuoliseen muistiin.

Laitelohkojen käyttö voi lyhentää automaatioprojektin ohjelmointiin kuluvaa aikaa merkittävästi, varsinkin jos projektissa on suuri määrä samanlaisia laitteita. Ilman laitelohkoja esimerkiksi tuotantolinjastolla, jossa on lukuisia sylintereitä lajittelemassa tuotteita, jokaisen sylinterin toiminta pitäisi ohjelmoida erikseen sekvenssin sisällä. Funktiolohkon avulla jokaisen sylinterin ohjauksen pystyy toteuttamaan yhdellä laitelohkolla, josta tehdään vain useampi instanssi. Samoja laitelohkoja voi siis käyttää jatkossa kaikissa projekteissa, joissa on samanlaisia laitteita, mikä helpottaa myös laitekokoonpanon muokkaamista projektin eri

vaiheissa. Laitelohkojen avulla ohjelman rakenne pysyy myös siistinä, mistä on hyötyä varsinkin käyttöönotossa ja vianetsinnässä. Myös virheiden määrä vähenee, sillä laitelohkot ovat jo valmiiksi testattuja ja toimivia. (Lydon 2018.)

4.2 Toimilaitelohkot TIA Portalissa

Funktiolohkolla toteutetun toimilaitelohkon rakenne koostuu IO-rajapinnasta, ohjelmakoodista sekä sen instanssidatalohkosta tai toisen funktiolohkon datalohkosta, johon funktiolohko on liitetty. IO-rajapinnan tehtävänä on toteuttaa vuorovaikutus lohkon ja muun ohjelman välillä. Rajapinta sisältää tulojen ja lähtöjen lisäksi tulolähtöjä. Staattiset muuttujat, väliaikaiset muuttujat ja vakiomuuttujat sijaitsevat lohkon datalohkossa. Datalohkossa muuttujia on myös tarvittaessa mahdollista määritellä retain-muuttujiksi. Tulojen kautta lohkolle syötetään tietoa ohjattavan laitteen tilasta, ja lähtöjen avulla laitetta ohjataan. Tulolähtöjä voidaan nimensä mukaisesti sekä lukea että kirjoittaa. Tulolähdöt myös viittaavat suoraan niihin liitetyn muuttujan osoitteeseen funktiolohkoa kutsuttaessa edellyttäen, että molemmat lohkot, joissa muuttujat sijaitsevat, ovat optimoituja (Siemens 2018b, 19). Tuloihin ja lähtöihin sen sijaan ainoastaan kopioidaan niihin liitetyn muuttujan arvo, mikä vaatii lisämuistin käyttöä kutsutussa lohkossa (Siemens 2018b, 59–60). Staattista muistialuetta käytetään lohkon sisällä sellaisille muuttujille, joita ei tarvitse tuoda lohkon ulkopuolelta eikä viedä ulos lohkosta. Väliaikaset muuttujat alustetaan joka ohjelmakierrolla, ja vakiomuuttujia käytetään esimerkiksi juuri vakioiden säilyttämiseen. Itse lohkoon ohjelmoidaan laitteen toiminta. Ohjelmoinnin suurimpana erona muihin lohkoihin on se, että ulkoisia muuttujia ei saa käyttää. TIA Portal ei anna virhettä ulkoisten muuttujien käyttämisestä, eli ohjelmoijan tulee olla tarkkaavainen, ettei käytä vääränlaisia muuttujia. Ulkoisten muuttujien käytön välttäminen on tärkeää siksi, että niiden käyttö poistaa mahdollisuuden lohkon modulaariseen käyttöön. Funktiolohkon kutsu, joka on nähtävissä kuvassa 8, muodostuu laatikosta, jonka vasemmalla puolella sijaitsevat lohkon rajapinnan tulomuuttujat ja oikealla puolella lähtömuuttujat. Jokaiseen rajapinnan muuttujaan voi yhdistää vastaavantyyppisen muuttujan tai kirjoittaa siihen halutun arvon. Tyhjäksi jätetty kenttä saa automaattisesti tietotyyppiä vastaavan oletusarvon, joka on esimerkiksi BOOL-tietotyypin muuttujalla false. Laatikon yläpuolella näkyy myös lainausmerkkien sisällä, mihin datalohkoon funktiolohko on liitetty.



Kuva 8. Esimerkki funktiolohkon kutsusta TIA Portalissa

Tiedonsiirtoa toimilaitelohkojen ja ohjattavien laitteiden välillä toteutetaan usein tavallisten muuttujien lisäksi käyttäjän luomilla tietotyypeillä. TIA Portalissa käyttäjän määrittelemä tietotyyppi on nimeltään PLC data type. Muuttujan sisällön voi määritellä vapaasti, ja se on käytettävissä koko projektissa. Esimerkiksi taajuusmuuttajat, ja niiden ohjausjärjestelmät pakkaavat niiden välillä siirrettävät tiedot telegrammeilla, jotka luodaan PLC data type-muuttujina. Telegrammi muodostuu otsikosta, käyttäjädatasta ja loppuosasta. Otsikko ja loppuosa muodostavat protokollan rungon, jonka sisällä käyttäjädata kulkee. PROFIdrive profiilissa on määritelty useampia tyypillisiin käyttötarkoituksiin soveltuvia telegrammeja, ja annettu niille tunnistenumero. PROFIdrive profiili on lisäys PROFIBUS ja PROFINET standardeihin. Se määrittelee yhtenäisen laitteiden toiminnan ja pääsyn ajureiden dataan (PROFIBUS Nutzerorganisation e.V. 2022). Esimerkiksi telegrammi 352:n tulokomponentti sisältää tilasanan lisäksi nopeus-, virta- ja vääntölukemia sekä varoitus- ja vikailmoitukset. Lähtökomponentti sisältää ohjaussanan lisäksi nopeuden asetusarvon ja jonkin verran vapaasti määriteltävää prosessidataa. (Siemens 2020b, 223–225.)

4.3 Toimilaitelohkot TwinCATissa

Toimilaitelohkot TwinCATissa toteutetaan hyvin samantyylisesti TIA Portaliin verrattuna. Datalohkojen puuttuminen vaikuttaa isolle erolle, mutta käytännössä se ei juurikaan vaikuta itse lohkon sisäiseen ohjelmointiin, sillä funktiolohkolla on edelleen oma muistinsa, sitä ei vain esitetä datalohkon muodossa. Eroavaisuuksia löytyy esimerkiksi lohkon muuttujiin

viitatessa. TIA Portalissa muuttujiin pystyy viittamaan lohkon numeron perusteella ja absoluuttisella osoitteella, kun TwinCATissa suositaan symbolista viittausta. Lohkot toteutetaan siis edelleen funktiolohkoilla, ja niiden rakenne koostuu IO-rajapinnan ja ohjelmakoodin lisäksi muuttujaeditorista, johon luodaan halutut muuttujat. Funktiolohkon muuttujatyyppejä ovat: lokaali, tulo, lähtö, tulolähtö, staattinen ja väliaikainen. Lokaalimuuttujat ovat paikallisia muuttujia ilman mitään erikoisominaisuuksia. Tulolähdöt toimivat samalla periaatteella kuin TIA Portalissa, eli ne viittaavat suoraan niihin linkitetyn muuttujan osoitteeseen. Staattiset muuttujat alustetaan, kun lohkoa kutsutaan ensimmäisen kerran. (Beckhoff.) Lisäksi muuttujia on mahdollista merkitä vakiomuuttujiksi sekä retain ja persistent muuttujiksi. TwinCATin funktiolohkon kutsu, joka on nähtävissä kuvassa 9, on datalohkon puutumista ja graafista tyyliä lukuun ottamatta hyvin samannäköinen TIA Portaliin verrattuna. Erona on kuitenkin esimerkiksi se, että tuloihin ja lähtöihin ei linkity automaattisesti tietotyypin oletusarvoa kuten TIA Portalissa. Tulolähtömuuttujiin on myös liitettävä jokin muuttuja tai arvo, muuten TwinCAT antaa virheilmoituksen.

Kuva 9. Esimerkki funktiolohkon kutsusta TwinCATissa

4.4 Toimilaitelohkot Esysillä

Esysillä on käytössä standardisoidut laitelohkot, joita käytetään kaikissa projekteissa. Laitelohkot on toteutettu TIA Portalissa funktiolohkoilla, joiden ohjelmointikielenä on Ladder. Monet samanlaiset laitteet toimivat usein samanlaisilla lohkoilla. Esimerkiksi moottoreilla

lohkot kuitenkin vaihtelevat sen mukaan, miten moottoria ohjataan. Suoralähdölle, taajuusmuuttajalle ja startterille on jokaiselle oma laitelohkonsa. Myös venttiileille ja PID-säätimille on erilaisia lohkoja. Laitelohkoja lisätään projekteihin TIA Portalin Global library -toiminnon avulla. Global library on eräänlainen kirjasto, joka mahdollistaa erilaisten lohkojen ja muiden ohjelmointiobjektien, kuten PLC data type tietotyyppien, säilyttämisen ja versioinnin. Kun ohjelmointiobjektia muokataan se saa aina uuden versionumeron, jolloin on helppo tarkistaa, onko käytettävä ohjelmointiobjekti ajan tasalla. Kirjasto sijaitsee Esysin verkkoasemalla, josta työntekijät pääsevät helposti lisäämään kirjaston omiin projekteihinsa ja korjaamaan mahdollisia virheitä sen sisällössä.

Esysillä on käytössä ohjelma nimeltään Hurauttaja, jonka avulla osa uuden projektin koodista kirjoitetaan automaattisesti. Ohjelmaa varten laaditaan Excel-taulukko, johon kootaan kaikki logiikassa olevat toimilaitteet. Taulukon ja muiden aputiedostojen avulla projektiin luodaan I/O-listat, laitteiden funktiolohkot ja instanssidatalohkot sekä luodaan tekstitiedosto, joka sisältää funktiolohkojen laitekutsut ja symbolitietojen kirjotuksen datalohkoihin. Ohjelmalla voi myös luoda HMI-puolelle tagilistat eli muuttujalistat, tekstilistat sekä hälytyslistat. Ohjelman käyttö vähentää toistuvaa työtä ja jättää enemmän aikaa tärkeiden toiminnallisuuksien ohjelmointiin. Lisäksi ohjelma vähentää käyttäjän tekemiä kirjoitusvirheitä, koska suuri osa koodista on automaattisesti luotua. (Roininen 2020, 2.)

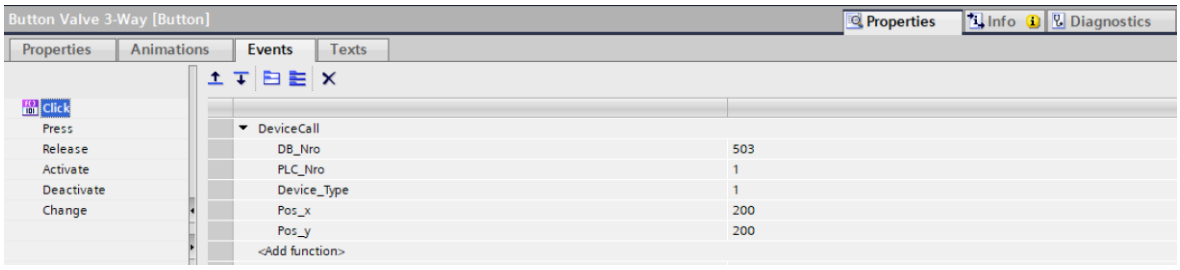
Esysin laitelohkot ovat opinnäytetyön kirjoituksen aikana pienessä murroksessa, jossa niitä muokataan yhteensopivaksi uuteen WinCC unifed -visualisointijärjestelmään. Laitelohkojen rakenne ja toiminta saattaa siis muuttua. Muutostöiden keskeneräisyyden vuoksi Esysin laitelohkojen toiminta- ja käyttöperiaate esitetään tällä hetkellä käytössä olevien lohkojen mukaan.

Esysin laitelohkot yhdistetään ohjelmasekvenssin lisäksi HMI-paneeliin, jossa laitteiden käsiajot on toteutettu. Käsiajoja käytettäessä valitaan laite, jota halutaan ohjata, jolloin HMI on yhteydessä valitun laitteen laitelohkoon. Esysillä laitteen valinta on toteutettu Visual Basic-skriptillä eli VB-skriptillä. VB-skriptin koodi kirjoitetaan TIA Portalissa omaan kansioonsa, josta skriptiä sitten kutsutaan halutussa paikassa. Laitteen valintaan käytettävä DeviceCall -skripti sisältää 5 parametriä:

- DB_Nro eli datalohkon numero, jonka arvoksi annetaan halutun laitelohkon instanssidatalohko.
- PLC_Nro, jonka avulla voidaan ohjata laitetta, joka on yhdistettynä eri PLC:hen kuin edellinen ohjattu laite.

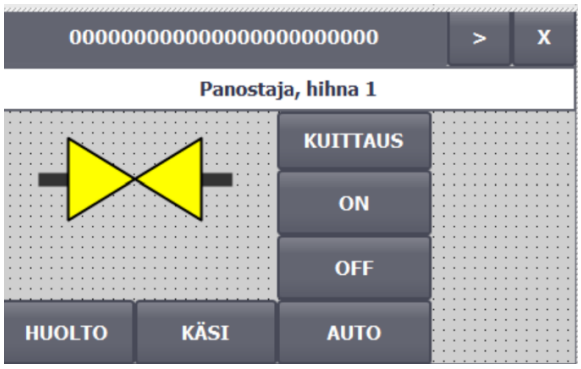
- Device_Type eli laitetyyppi, jonka arvoksi valitaan numero, joka avaa numeroa vastaavan laitetyypin ponnahdusikkunan. Ponnahdusikkuna sisältää laitteen ohjaukset.
- Pos_x, jonka arvoksi annetaan x-koordinaatti HMI-paneelin näytöllä, johon ponnahdusikkunan halutaan avautuvan.
- Pos_y, jonka arvoksi annetaan vastaavasti y-koordinaatti.

Parametrien arvot annetaan kutsun yhteydessä. Kutsu tapahtuu siten että näytöllä painetaan nappia, johon kutsutoiminto parametreineen on liitetty. Esimerkki tällaisen napin sisällöstä on nähtävissä kuvassa 10.



Kuva 10. Venttiilin ohjausikkunan avausnapin toiminnallisuus

Esysin laitelohkot ovat optimoimattomia, sillä muuttujan tarkka osoite koostuu datalohkon numerosta ja lohkon sisäisestä offset arvosta. Näin ollen lohkon jokaisen instanssin muuttujiin voidaan viitata vaihtamalla osoitteesta pelkkä lohkonumero, sillä muuttujan offset arvo on kaikissa lohkoissa sama. Kutsu tapahtuu painamalla näytölle sijoitetusta napista, jonka toimintoparametreihin on syötetty halutut arvot, kuten datalohkon numero. Napin kutsuttua DeviceCall-skriptin, avautuu näytölle valitun laitteen ponnahdusikkuna. Ponnahdusikkunassa on käsiajojen lisäksi kyseisen laitteen nimi ja datalohkon numero. Esimerkki tällaisesta ikkunasta on nähtävissä kuvassa 11.



Kuva 11. Venttiilin ohjausikkuna

Ohjausikkunoissa on myös lisävalikko, jonka saa avattua ikkunan yläreunassa olevasta nuolinäppäimestä. Lisävalikko, joka on nähtävissä kuvassa 12, sisältää parametreja, hälytyksiä sekä muuta lisätietoa.

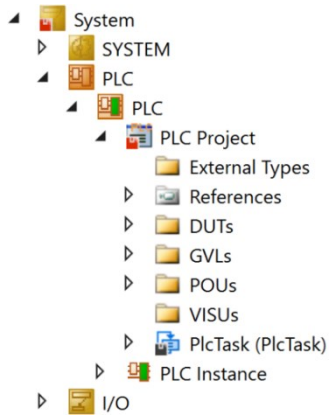
Kuva 12. Venttiilin ohjausikkunan lisävalikko

5 Toimilaitelohkojen ohjelmointi TwinCATilla

5.1 Toimilaitelohkot

Laitelohkojen ohjelmointi aloitetaan luomalla TwinCAT XAE -projektitiedosto, joka sisältää pohjan projektin rakenteelle. Se luo oletuksena projektipuun ja konfiguraatiot, joilla varsinaiset projektin osat ohjelmoidaan. Konfiguraatiot ovat nimeltään System, Motion, PLC, Safety, C++, Analytics ja I/O. Yleisesti ottaen System-, I/O-, PLC- ja Safety-konfiguraatiot muodostavat tavallisen automaatioprojektin ytimen, ja Motion, C++ sekä Analytics ovat esimerkkejä lisäosista, joita projekteihin on mahdollista lisätä. PLC-konfiguraatio sisältää itse PLC-projektin eli logiikkakoodin, ja Safety-konfiguraatiossa määritellään projektin turvalaitteiston toimintaa. I/O konfiguraatiossa määritellään yhdistetyn I/O-kortin tuloja ja lähtöjä.

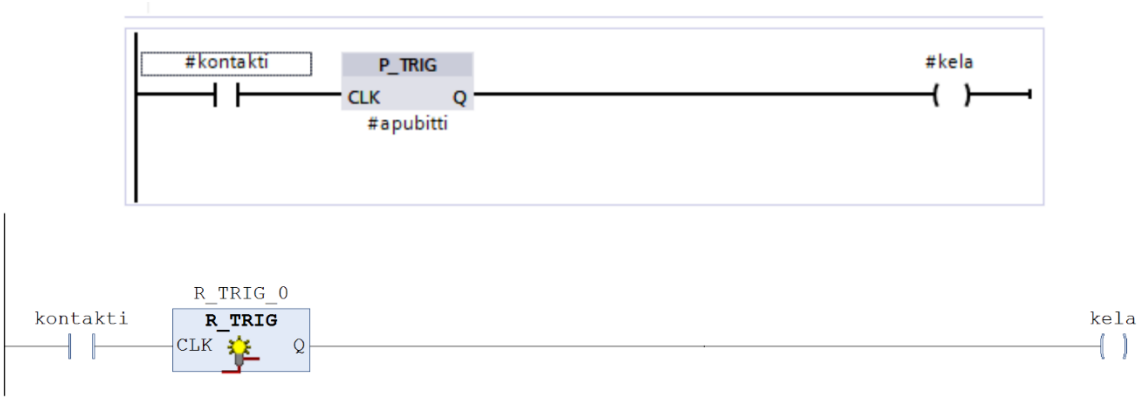
Toimilaitelohkojen ohjelmointia varten tarpeellisia konfiguraatioita ovat System ja PLC. I/O-konfiguraatiota ei voi piilottaa, joten se jää tyhjäksi, sillä testiprojektista ei ole tarvetta yhdistää fyysisiin toimilaitteisiin. Fyysisten toimilaitteiden testausta käsitellään kappaleessa 6. Muut konfiguraattorit voi joko piilottaa tai jättää tyhjiksi. Kuvassa 13 on nähtävissä TwinCAT 3 projektipuu, johon on lisätty PLC projekti ja josta on piilotettu tarpeettomat konfiguraattorit.



Kuva 13. TwinCAT 3 projektipuu

Ohjelmointi aloitetaan luomalla POUs-kansioon funktiolohko. Lohkon muuttujaeditorissa määritellään tarpeelliset muuttujat ja niiden tyypit. Muuttujien nimet ja tietotyypit on helppo kopioida TIA Portalista ja lisätä niihin TwinCATin syntaksin vaatimat välimerkit Excelissä. Myös kommentit lisätään Exceliin, josta koko muuttujalista kommentteineen voidaan sitten kopioida Valve-lohkon muuttujaeditoriin. Muuttujaeditorissa on kaksi erilaista tilaa: tekstipohjainen ja taulukkopohjainen. Tekstipohjaiseen editoriin pystyy suoraan kopioimaan esimerkiksi juuri Excelistä tekstiä, mutta taulukkopohjaisessa editorissa jokainen rivi on luotava erikseen. Muuttujiin on kuitenkin tehtävä joitakin muutoksia. Esimerkiksi tyhjät, lohkon rakennetta ylläpitävät muuttujat, voi jättää tuomatta TIA Portalista, sillä TwinCATissa

käytetään symbolista viittaamista, joten niille ei ole enää tarvetta. Myös positiivisen ja negatiivisen reunan tunnistukseen käytettävä Bool-tyypin muuttujiia sisältävä taulukko ei ole enää tarpeellinen, sillä TwinCATissa reunan tunnistus tapahtuu valmiilla standardifunktiolohkoilla, joille luodaan omat instanssinsa. Tia Portalissa olisi mahdollista käyttää standardifunktiolohkoa reunantunnistuksessa, mutta Esysin lohkoissa sitä ei ole käytetty. Eroa reunantunnistuksien toteutuksessa on havainnollistettu kuvassa 14.



Kuva 14. TIA Portalin reunantunnistus apubitin avulla ja TwinCATin standardifunktiolohko reunantunnistukseen

Kun muuttujat on saatu tuotua, edetään itse ohjelmointivaiheeseen. Ohjelmointikielenä lohkojen ohjelmoinnissa toimii Ladder. Lohkojen osuudet, kuten hälytyskäsittely, saattavat muuttua, kun lohkoja otetaan käyttöön oikeissa projekteissa. Lohkot toteutetaan kuitenkin alustavasti samalla tavalla kuin TIA Portalissa. TwinCATin ladder-editorin kömpelyys tulee hyvin esille, kun yrittää tehdä samanlaisia networkkeja kuin TIA Portalissa, jossa samaan aiheeseen liittyvät ohjaukset on usein haluttu samaan networkiin, jolloin niiden pitää olla omilla riveillään, eikä niiden tule olla yhteydessä toisiinsa. TwinCATissa tämän toteuttaminen on kuitenkin hieman hankalampaa. Vaikka ohjelmointikieli on sama, on TwinCATin ladder-ohjelmaeditori huomattavasti rajoittuneempi kuin TIA Portalin, ja vaikuttaakin siltä, että TwinCATissa on panostettu huomattavasti enemmän tekstipohjaisten kielten ohjelmaeditoreihin. Esimerkiksi tilanteen, joka näkyy kuvassa 15, missä kaksi erillistä kelaa on aseteltu allekkain samaan networkiin ilman, että ne ovat yhteydessä toisiinsa, onnistuu vain kiertoreitin kautta. Kontaktin voi kopioida, ja sen jälkeen valita koko network ja liittää kontakti ctrl+v komennolla, jolloin kontakti ilmestyy omalle rivilleen samaan networkiin.

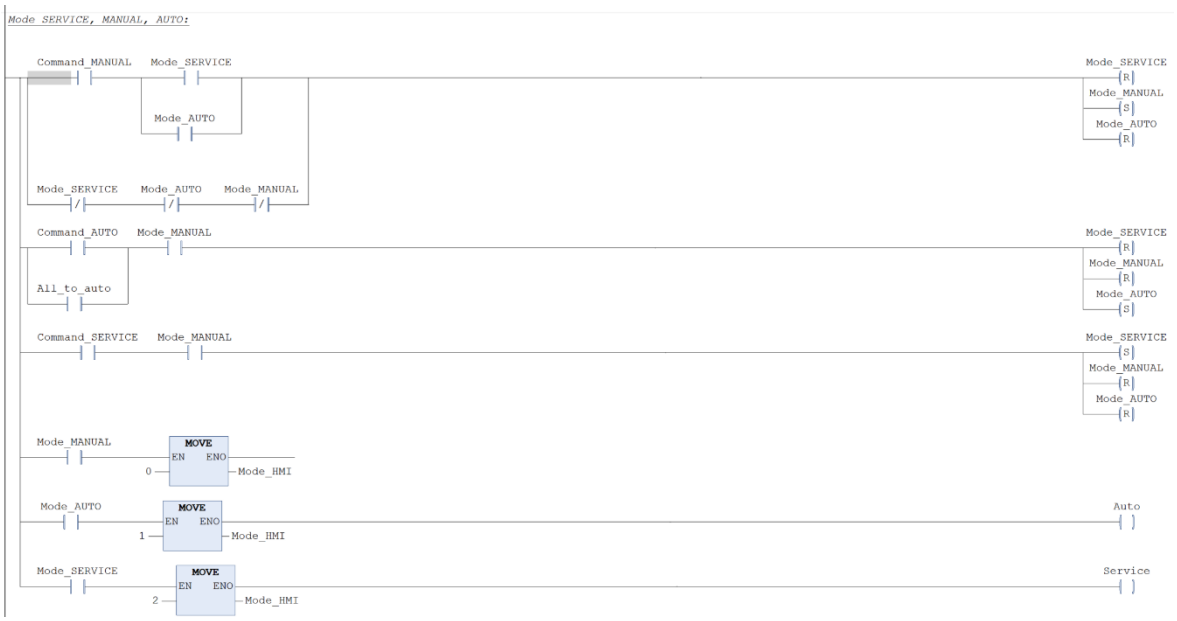
Kuva 15. Kaksi kelaa omilla riveillänsä

Toinen vaihtoehto, joka näkyy kuvassa 16, on aloittaa ensimmäisen kelan edestä haara, johon sijoitetaan toinen kela ja kontakti. Näin saadaan samalla lailla toimiva mutta hieman erinäköinen ratkaisu.

Kuva 16. Kaksi kelaa omilla riveillänsä haaran avulla

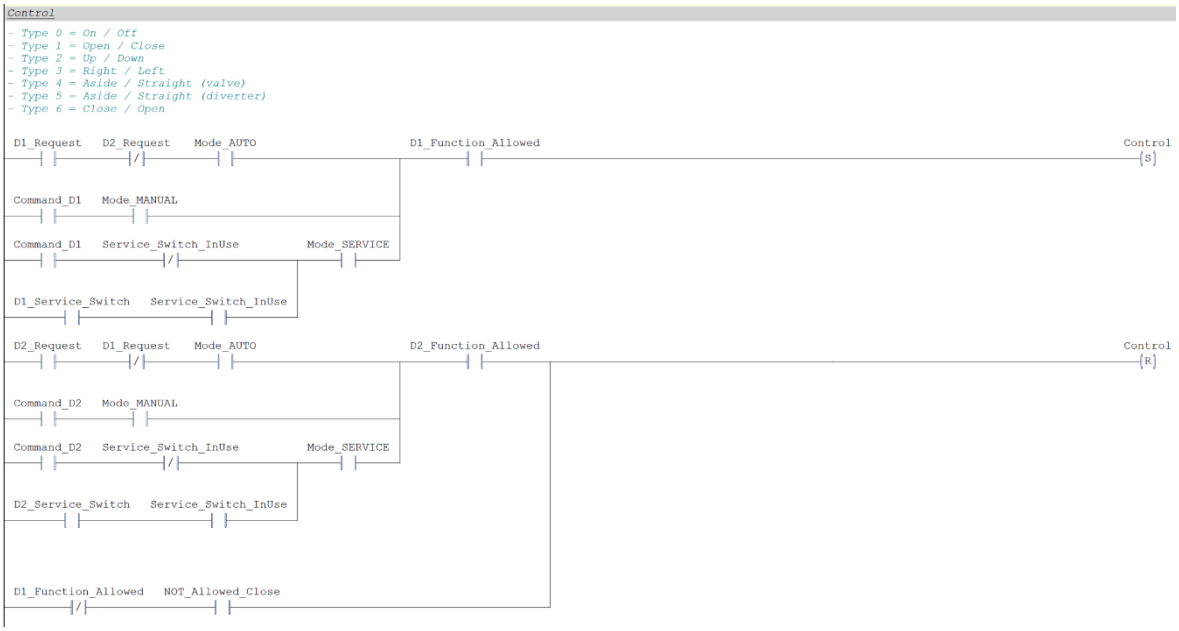
Haaran avulla toteutettu ratkaisu vaikuttaa olevan parempi, sillä kopiointi ja liittämistekniikalla rivien väliin ei pysty lisäämään uutta riviä, vaan se pitää tehdä networkin loppuun. Haaroja taas pystyy lisäämään mihin tahansa väliin, mutta vain jos networkissa ei ole käytetty kopiointi ja liittämistekniikkaa, sillä se estä uusien haarojen tekemisen networkiin. Networkit voisi siis toteuttaa samalla tavalla kuin TIA Portalissa, mutta se ei olisi järkevää. Käyttökelpoisimpana ratkaisuna on siis haarojen käyttö.

Esimerkkinä esitetään pätkä Valve-nimisen venttiililohkon koodia ja toimintaperiaatetta. Lohko on ohjelmoitu vastaavan nimisen TIA Portalissa ohjelmoidun lohkon pohjalta. Lohkon ensimmäinen network käsittelee yhteyttä HMI:hin, ja siihen palataan osiossa 5.2. Toinen network, joka on nähtävissä kuvassa 17, sisältää manuaaliohjauksen venttiilin ohjaustilalle, joka koostuu kolmesta tilasta: Auto, Manual ja Service. Networkin lopussa ohjataan myös ohjaustilan tilatiedon välittämistä HMI:in ohjausikkunalle. Networkkien otsikko- ja kommenttikentät piti käydä asettamassa näkyviin Ladder-editorin asetuksista. Myös muita editorin visuaalisia ominaisuuksia pystyy muokkaamaan, kuten operaattoreiden ja muuttujien kommenttien sekä muuttujien tarkkojen osoitteiden näkyvyyttä. Osoitteiden näkyvyydellä ei laitelohkoissa ole juurikaan merkitystä, sillä kaikki käytettävät muuttujat ovat paikallisia, eli niillä ei ole osoitetta.



Kuva 17. Valve-lohkon toinen network

Kolmannessa networkissa, joka on nähtävissä kuvassa 18, kuunnellaan eri muuttujien arvojen yhdistelmiä, joiden mukaan Bool-tyyppinen ohjausmuuttuja nimeltä Control, asetetaan päälle tai pois. Ohjausmuuttuja ohjaa lohkon lähtöjä, mutta ei suoraan. Sen toimintaan vaikuttaa, onko lohkossa päällä pulssiohjaus. Pulssiohjauksella lähtö on päällä vain tietyn ajan ja ilman pulssiohjausta lähtö jää päälle pysyvästi. Lohko oli TIA Portalilla toteutettu SR-standardifunktiolohkolla, mutta TwinCATissa SR-lohkot vaativat oman muuttujatyyppinsä, eivätkä hyväksy Bool-tyyppistä muuttujaa. Näin ollen network on toteutettu set- ja reset-keloilla.



Kuva 18. Valve-lohkon kolmas network

5.2 Yhteys HMI:hin

Yhteys HMI:hin on muodostettava TwinCATissa huomattavasti erilaisella tavalla kuin TIA Portalissa. Datalohkojen puuttumisen vuoksi numeromuuttujalla haluttuun datalohkoon osoittaminen ei onnistu, joten sen tilalle piti kehittää uusi ratkaisu. Esysin nykyisissä TwinCAT-projekteissa käsiajosivulta ohjataan suoralla osoituksella laitelohkojen muuttujia, jolloin jokaisen napin komento pitää linkittää jokaisen uuden laitelohkon kohdalla uudelleen, eikä se ole järkevää. Esysillä ei lisäksi ole TIA Portalin kaltaisia ponnahdusikkunoita käytössä TwinCATissa, vaan useat eri laitteidenkin napit ovat samalla käsiajosivulla. Tämän ratkaisun tilalle otetaan käyttöön ponnahdusikkunana aukeavat ohjausikkunat samaan tapaan kuin TIA Portalissa. Samalla luovutaan myös suorasta osoituksesta muuttujiin. Avatun ohjausikkunan mukana muuttuva suora osoitus voisi todennäköisesti olla mahdollista toteuttaa esimerkiksi JavaScriptillä, mutta yhteyden ymmärrettävyyden ja selkeyden vuoksi ratkaisu pyrittiin löytämään ohjelmasta ilman monimutkaisen koodin apua. Erityisempää ohjelmointikieliosaamista ei siis tarvita, jolloin useimmat henkilöt pystyvät tarvittaessa nopeasti tutustumaan yhteyden muodostamisessa käytettyyn ratkaisuun ja käyttämään sekä tarvittaessa korjaamaan tai muokkaamaan sitä.

Ratkaisuksi yhteyden muodostamiseen valikoitui DUT-struktuuri, joka sisältää muuttujat, joita halutaan ohjata ohjausikkunan kautta. Struktuurista luodaan array eli taulukko, jossa on niin monta alkiota kuin ohjattavia laitteita on ohjelmassa. Näin struktuurista on tavallaan luotu instansseja niin kuin itse laitelohkoistakin. Kuvassa 19 on nähtävissä ohjausikkunoiden toiminnan kannalta tarpeelliset, logiikkaohjelmoinnin puolella olevat objektit, jotka ovat HMIValve-nimisen struktuurin sisältö ja aValve-niminen taulukko, jonka alkioina on nValves-nimisen tasalukumuuttujan määrittelemä määrä struktuureja.

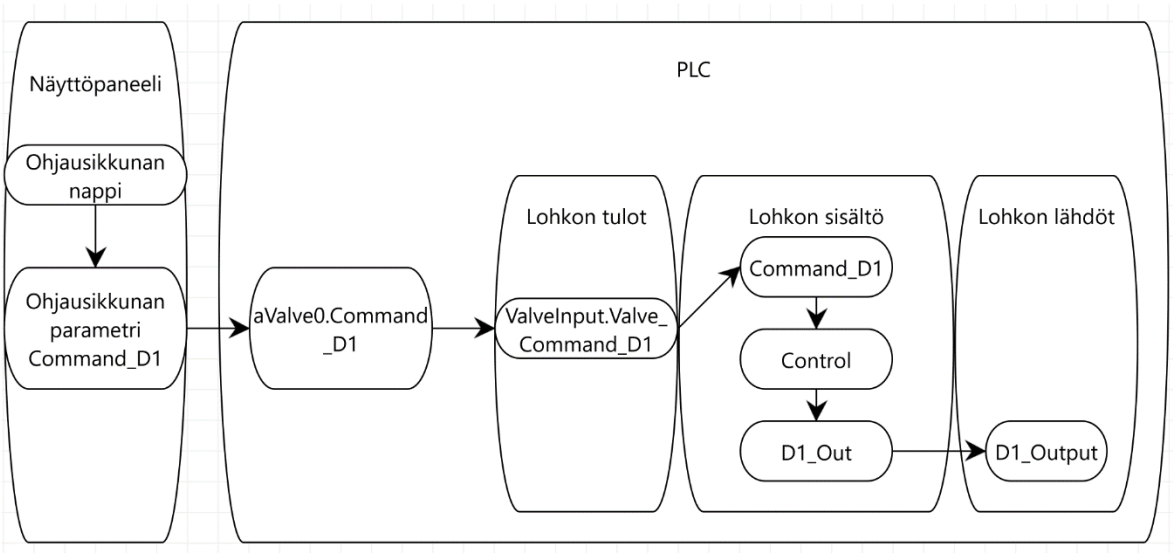
```
HMIValve
TYPE HMIValve :
STRUCT
    Valve_D1_Feedback_InUse          : BOOL;
    Valve_Feedback_Delay             : REAL;
    Valve_D2_Service_Switch          : BOOL;
    Valve_D1_Function_Allowed        : BOOL;
    Valve_D1_Service_Switch          : BOOL;
    Valve_Command_D1                 : BOOL;
    Valve_Symbol                     : STRING;
    Valve_Command_RESET              : BOOL;
    Valve_D1_Output                  : BOOL;
    Valve_Service_Switch_InUse       : BOOL;
    Valve_D2_Feedback_InUse          : BOOL;
    Valve_DB_Number                  : INT;
    Valve_D1_Feedback                : BOOL;
    Valve_Mode_HMI                   : INT;
    Valve_D2_Function_Allowed        : BOOL;
    Valve_Function_Counter           : DINT;
    Valve_Command_MANUAL             : BOOL;
    Valve_Command_SERVICE            : BOOL;
    Valve_D1_Request                 : BOOL;
    Valve_Visible_ControlButtons     : BOOL;
    Valve_D2_Output                  : BOOL;
    Valve_Command_D2                 : BOOL;
    Valve_Type                       : INT;
    Valve_D2_Request                 : BOOL;
    Valve_Alarm_Counter              : DINT;
    Valve_State_HMI                  : INT;
    Valve_D2_Feedback                : BOOL;
    Valve_Command_AUTO               : BOOL;
    Valve_Feedback_Counter           : REAL;
    Valve_Alarm_Auto                 : BOOL;
    Valve_Alarm_D1_Feedback          : BOOL;
    Valve_Alarm_D2_Feedback          : BOOL;
END_STRUCT
END_TYPE

Settings
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
    nValves     : UINT  := 20;  // Amount of Sinamics inverters
END_VAR

HMI
{attribute 'qualified_only'}
VAR_GLOBAL
    aValve      : ARRAY [0..Settings.nValves] OF HMIValve;
END_VAR
```

Kuva 19. Struktuurin ja taulukon sisältö

Jokaiseen laitelohkon instanssiin tuodaan yksi taulukon alkio tulolähtömuuttujana, ja alkion sisältämän struktuurin sisältö yhdistetään lohkon sisällä varsinaisiin lohkon muuttujiin ST-kielen sijoituskomennoilla, kuten nähdään kuvassa 20. Yhdistäminen tapahtuu siis niin, että esimerkiksi Command_D1-muuttujan arvo, jota käytetään lohkon sisällä ohjaamaan lohkon lähtö D1_Output päälle, on yhdistettynä ValveInput.Valve_Command_D1-muuttujan arvoon. Jälkimmäisen muuttujan arvo on linkitetty aValve-taulukon ensimmäisen alkion sisältämän struktuurin muuttujaan, jonka nimi on myös Command_D1. Tähän muuttujaan on yhteydessä nappi HMI-paneelin ohjausikkunassa, jota painamalla muuttujan arvo muuttuu, jolloin myös lohkon sisäisen Command_D1-muuttujan arvo muuttuu. Näin saadaan siis ohjattua lohkon lähtöä D1_Output. Selvennys esimerkistä näkyy kuviossa 1, jossa havainnollistetaan yhteyttä lohkon lähdön ja HMI-paneelin napin välillä.

```
                                EXECUTE
EN      Command_D1      :=    ValveInput.Valve_Command_D1;          ENO-
        Command_RESET   :=    ValveInput.Valve_Command_RESET ;
        Function_Counter :=   ValveInput.Valve_Function_Counter ;
        Command_MANUAL  :=    ValveInput.Valve_Command_MANUAL;
        Command_SERVICE :=    ValveInput.Valve_Command_SERVICE;
        Command_D2      :=    ValveInput.Valve_Command_D2;
        Valve_Type      :=    ValveInput.Valve_Type ;
        Alarm_Counter   :=    ValveInput.Valve_Alarm_Counter ;
        State_HMI       :=    ValveInput.Valve_State_HMI ;
        Command_AUTO    :=    ValveInput.Valve_Command_AUTO ;
        Feedback_Counter :=   ValveInput.Valve_Feedback_Counter ;
        Alarm_Auto      :=    ValveInput.Valve_Alarm_Auto;
        Alarm_D1_Feedback :=  ValveInput.Valve_Alarm_D1_Feedback;
        Alarm_D2_Feedback :=  ValveInput.Valve_Alarm_D2_Feedback;
```

Kuva 20. Muuttujien arvojen yhdistämistä ST-kielellä



Kuvio 1. Havainnollistus HMI-paneelin ohjausikkunan yhteydestä toimilaitelohkoon

Ohjausikkuna ohjelmoidaan user control -objektilla, joka kutsutaan näytölle ponnahdusikkunana. Ponnahdusikkunan toiminnallisuus luodaan TcHMI-Popups-nimisellä lisäosalla. Lisäosia TwinCATin HMI engineering alustalla hallinnoidaan NuGet Package Managerilla. Sieltä löytyy virallisia Beckhoffin lisäosia mutta myös nuget.org sivustolta löytyviä lisäosia. Nuget.org on .NET-pakettien jakamiseen tarkoitettu alusta. Ohjausikkuna avataan näytölle komennolla, jonka toimintoparametrejä ovat seuraavat:

- Bindings valikko, jossa valitaan esimerkiksi mihin logiikan puolella olevan taulukon alkioon ohjausikkuna on yhteydessä.
- ContentFace, joka määrittää mikä user control näytölle kutsutaan.
- Destination, joka määrittää mille ikkunalle ponnahdususikkuna tulee näkyviin.
- Dragging, joka määrittää voiko ponnahdusikkunaa raahata.
- InsertWherePressed, joka määrittää avautuuko ponnahdusikkuna siihen kohtaan missä ikkunan avaavaa nappia on painettu.
- Modal, joka määrittää pystyykö ponnahdusikkunan taustalla olevia objekteja käyttämään, kun ponnahdusikkuna on auki.
- Positioning, joka määrittelee ponnahdusikkunan paikkaa ja kokoa näytöllä.

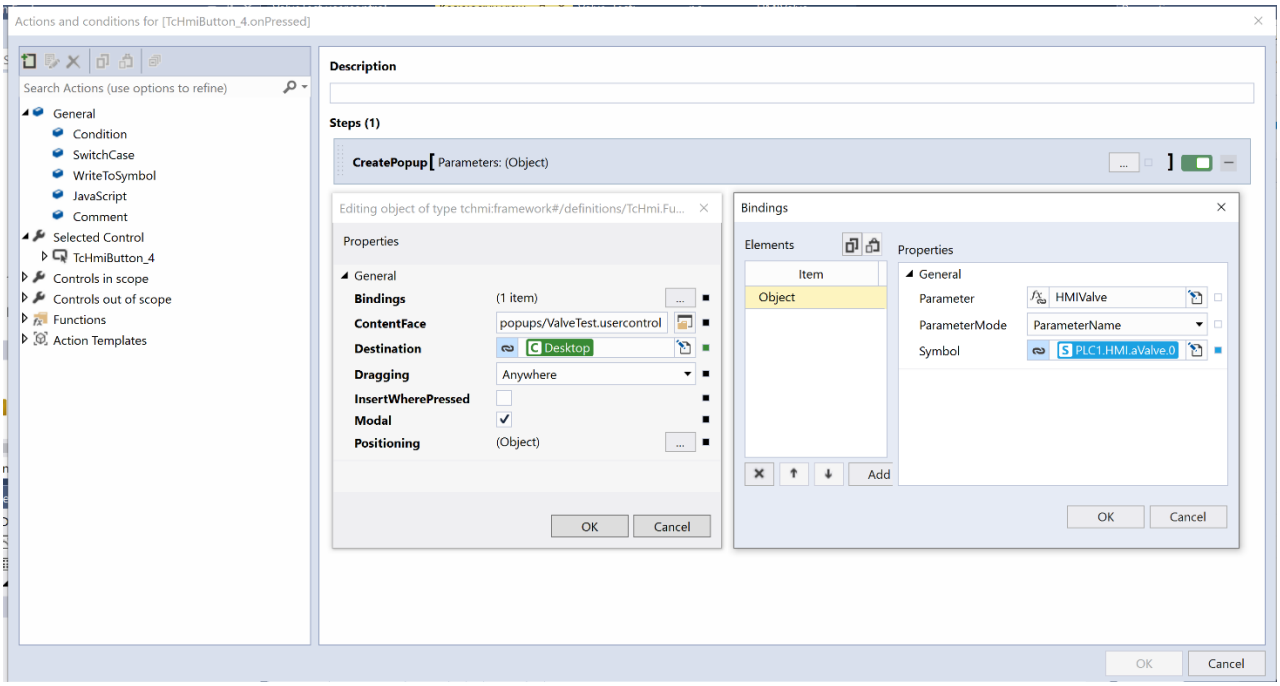
Ohjausikkunalle tehdään kaksi parametrimuuttuja. jotka ovat nähtävissä kuvassa 21. HMI-Valve vastaa sisällöltään laitelohkon ohjaukseen käytettävää struktuuria, ja DetailMode liittyy ohjausikkunan lisävalikkoon.

Parameters:

Name	Datatype	(Drop) Default Value	DefaultValueInternal	Bindable	Visibility
DetailMode	G Boolean	False	False	☑	☑
HMIValve	P PLC1.HMIValve	(Object) ...	(Object) ...	☑	☑

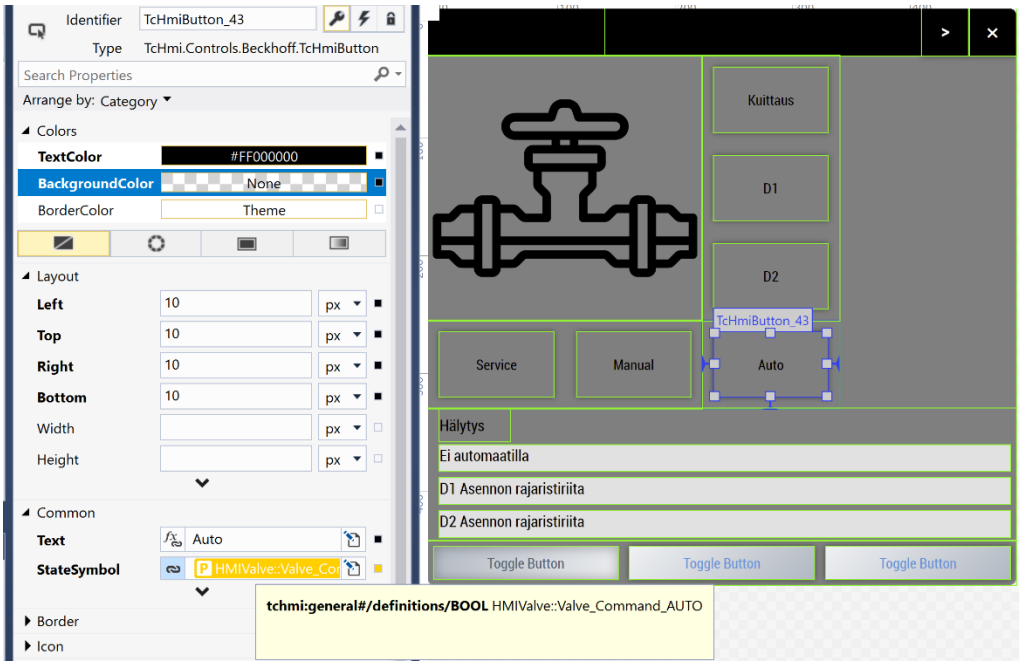
Kuva 21. Ohjausikkunan parametrit ja niiden tietotyypit

Ohjausikkunan avausnapin Bindings-valikossa määritellään objekti, joka linkitetään taulukon alkioon. Tällöin nappi yhdistää valitun alkion sisällön ohjausikkunan HMIValve-parametrimuuttujaan. Ohjausikkunaan ei siis missään vaiheessa tule muutoksia, ja ikkunan sisällä viitataan ainoastaan sen parametrimuuttujan sisältöön. Näin ollen yhdenlaiselle laitteelle tarvitsee tehdä ainoastaan yksi user control -objekti, mikä helpottaa muutosten tekemistä, koska niitä ei tarvitse tehdä useampaan paikkaan. Kuvassa 22 näkyy ohjausikkunan napin sisältämä komento ja sen sisältö.



Kuva 22. Ohjausikkunan napin toiminnallisuus

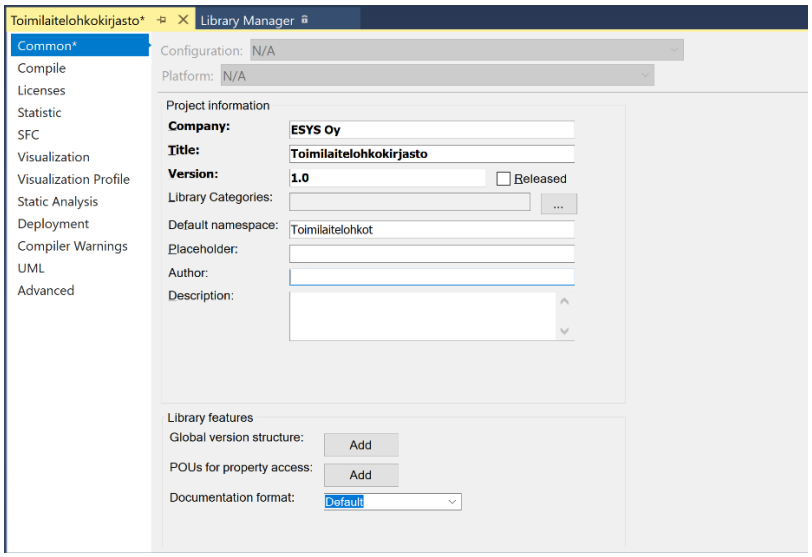
Ohjausikkunan sisällä napit yhdistetään HMIValve-muuttujan sisältöön kuvan 23. tavalla. Siinä napin ominaisuuksissa olevaan StateSymbol-kenttään yhdistetään HMIValve-muuttujan sisältämä COMMAND_AUTO-muuttuja. StateSymbol-kenttään syötetty muuttuja seuraa napin toimintaa niin, että kun nappi on painettuna, muuttuu kenttään syötetyn Bool-tyypin muuttujan arvoksi yksi, ja kun napista päästetään irti, se palautuu nollaksi.



Kuva 23. Automaattitilan napin toiminnallisuus

5.3 Toimilaitelohkokirjasto

TwinCAT sisältää versionhallinnan mahdollistavan kirjastotyökalun, jonka avulla on helppo toteuttaa juuri esimerkiksi toimilaitelohkoille oma kirjastonsa. Kirjaston voi asentaa mihin tahansa projektiin ja käyttää esimerkiksi sen sisältämiä lohkoja ja DUT-objekteja. Toimilaitelohkokirjastoon on järkevää lisätä itse lohkojen lisäksi ainakin niiden DUT-struktuuri. Kirjaston luominen tapahtuu luomalla uusi tyhjä PLC-projekti, johon lisätään halutut lohkot ja objektit. Projektin ominaisuuksiin, jotka näkyvät kuvassa 24, kirjoitetaan yrityksen nimi, otsikko ja versionumero. Kohta ”Released” määrittelee voiko kirjastoprojektiin tehdä muutoksia. Jos kohta on valittuna, niin muutoksia tehdessä kirjaston versionumeroa pitää kasvattaa, mikä ilmoittaa kirjaston käyttäjille, että kirjastoa on muokattu. ”Default namespace”-kenttä mahdollistaa erittelyn kahden samannimisen objektin, kuten laitelohkon välillä, joista toinen on kirjastosta peräisin ja toinen itse projektissa. Esimerkki namespacen käytöstä näkyy kuvassa 25, jossa ylempi taulukkomuuttuja käyttää lokaalia projektissa luotua struktuuria nimeltä HMIValve, ja alempi viittaa kirjastosta löytävään samannimiseen struktuuriin käyttämällä kirjastolle luotua namespacea Toimilaitelohkot.

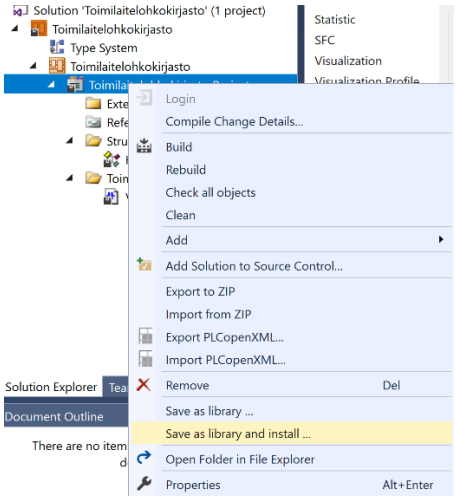


Kuva 24. Projektin ominaisuuksia.

```
VAR_GLOBAL
    aValve        : ARRAY [0..Settings.nValves] OF HMIValve;
    aValve2           : ARRAY [0..Settings.nValves] OF Toimilaitelohkot.HMIValve;
END_VAR
```

Kuva 25. Namespacen käyttö

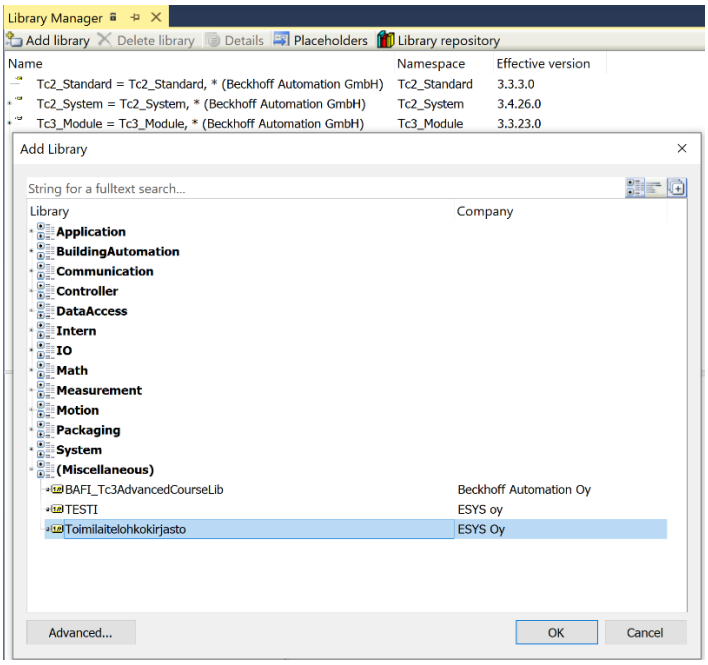
PLC-projekti tallennetaan kirjastoksi valitsemalla se projektipuusta ja painamalla hiiren oikealla näppäimellä avautuvasta valikosta kohtaa ”save as library and install”, kuten nähdään kuvassa 26.



Kuva 26. Projektin tallentaminen kirjastona

Tallennussijainnilla ei ole väliä, sillä TwinCAT kopioi kirjastotiedoston omaan kirjastokansioonsa, joka on yleensä: ”C:\TwinCAT\3.1\Components\Plc\Managed Libraries”. TwinCAT luo sijaintiin kirjastolle oman syntaksinsa, johon ei ole tarvetta koskea resurssienhallinnan

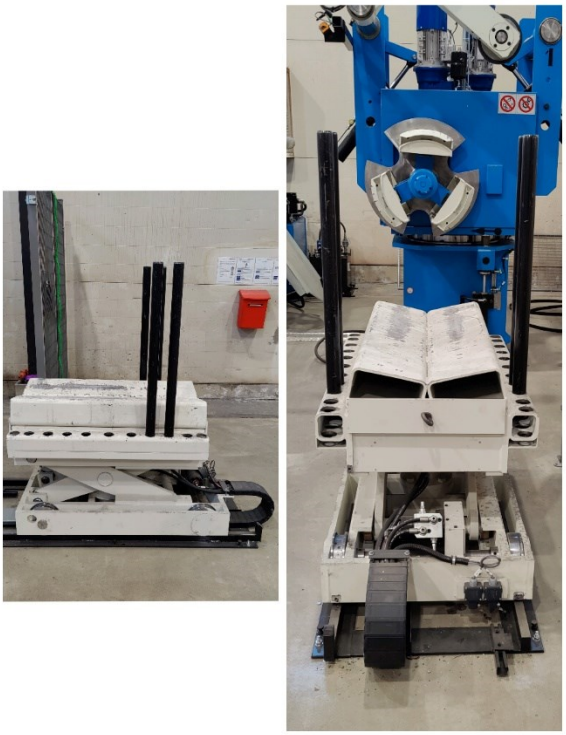
kautta. Kuvassa 27 näkyy, miten kirjaston lisääminen projektiin tapahtuu Library managerissa eli kirjastonhallinnassa. Kirjaston lisäämisen jälkeen sen objekteja pystyy käyttämään projektissa lokaalien objektien tapaan.



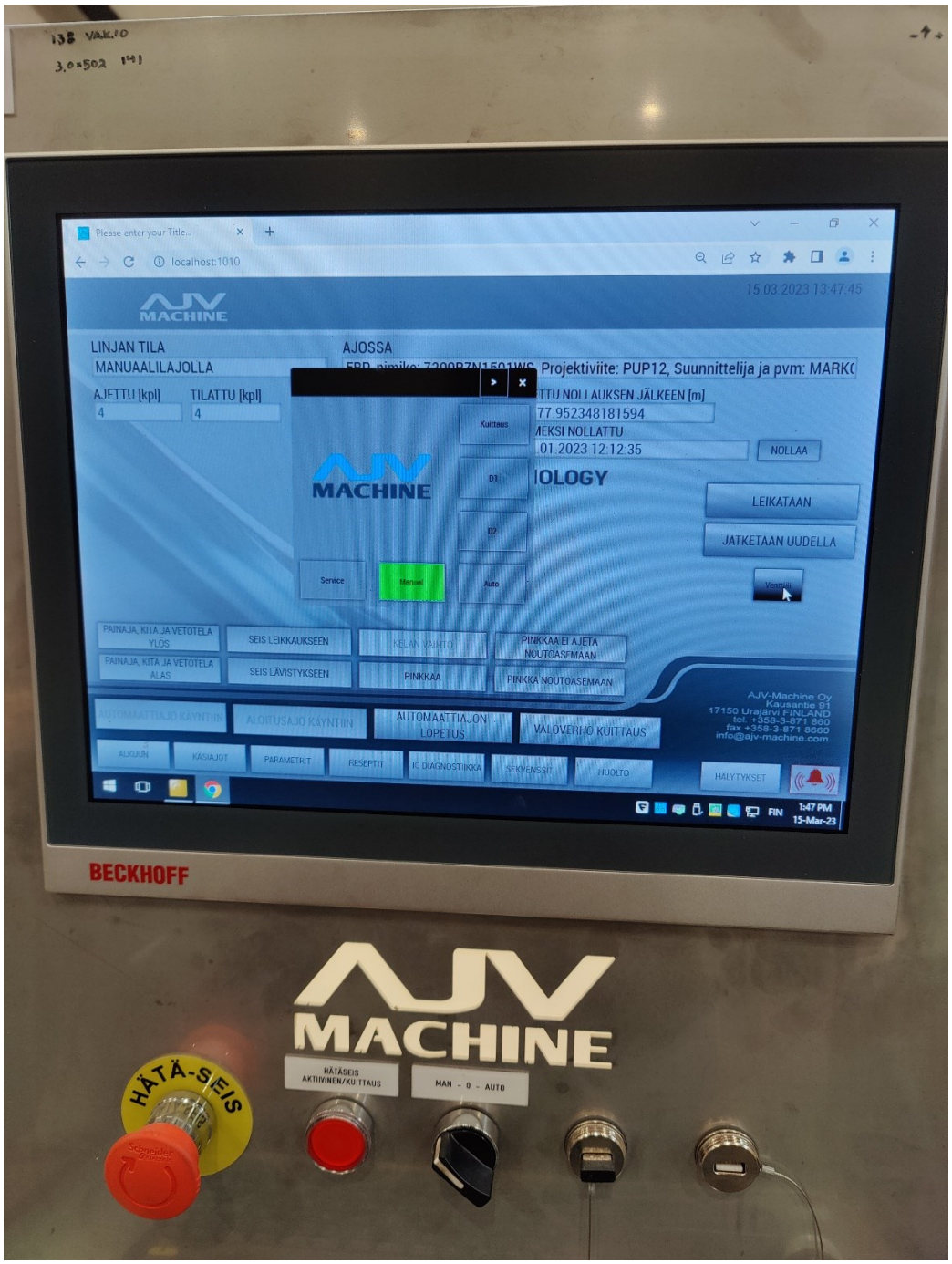
Kuva 27. Kirjaston lisääminen kirjastonhallinnassa

6 Toimilaitelohkojen testaus

Toimilaitelohkon testauksessa tavoitteena oli todeta lohkon toimivuuden lisäksi näyttöpaneelin ja sen ohjausikkunan yhteyden toimivuus logiikkaohjelmaan. Testilaitteena toimi kolmitoimisella hydrauliventtiilillä ohjattava haspelin nostopöytä. Laite oli osa linjastoa, joka sijaitsee Weckman Steel Oy:n Vierumäen tehtaalla. Laite on nähtävissä kuvassa 28 ja laitteen ohjaukseen käytetty teollisuustietokone kuvassa 29. Laitteen ohjaamisen käytettiin Valve-nimistä laitelohkoa, jota jouduttiin hieman muokkaamaan, sillä sen toimintatiloista ei löytynyt kolmitoimisen venttiilin ohjaukseen tarkoitettua tilaa. Testiä varten luotiin myös ohjausikkuna, josta pystyi vaihtamaan laitelohkon toimintatilaa käsiajon, automaatin ja huoltotilan välillä sekä ohjamaan sitä ylös ja alas. Laite liikkuisi myös eteen ja taakse mutta eri venttiilillä, jota ei testissä ohjattu.

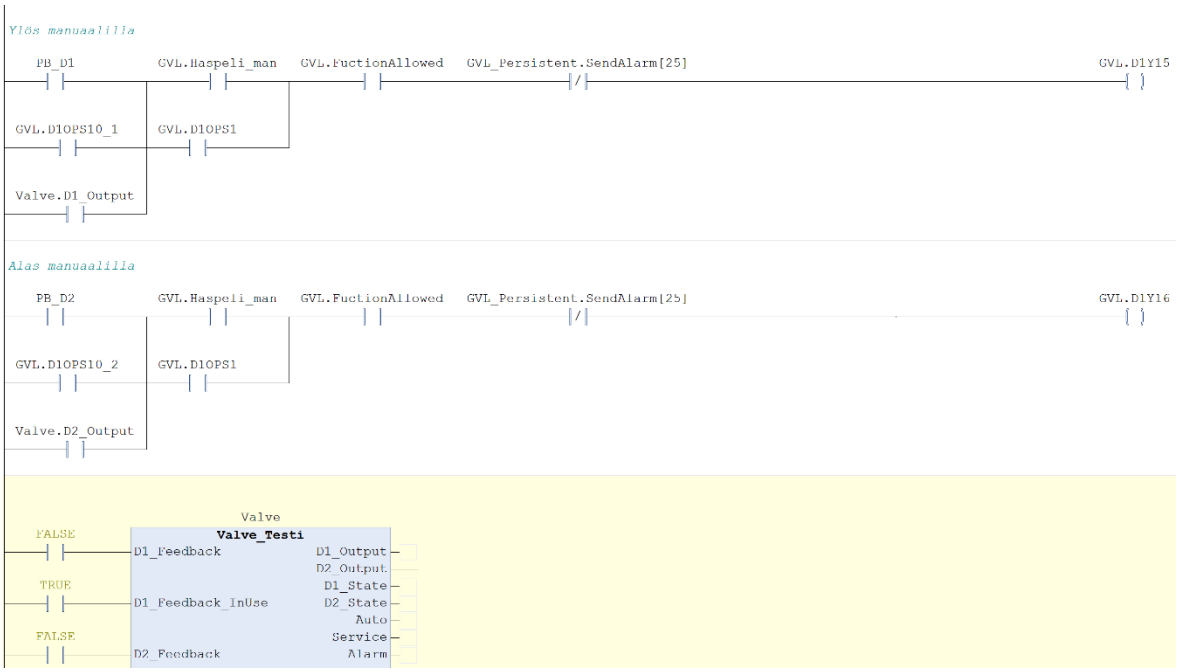


Kuva 28. Haspelin nostopöytä



Kuva 29. Testauksessa käytetty teollisuustietokone

Testaus aloitettiin syöttämällä teollisuustietokoneeseen logiikkaohjelma, joka sisälsi testattavan toimilaitelohkon. Kuten kuvassa 30 näkyy, laitelohko oli integroitu nostopöydän manuaaliohjaukseen lisäämällä siihen kontaktit, jotka kuuntelivat lohkon lähtöjä nimeltä D1_Output ja D2_Output.



Kuva 30. Valve venttiililohko yhdistettynä nostopöydän manuaaliohjaukseen.

Logiikkaohjelman syöttämisen jälkeen teollisuustietokoneen HMI-serverille julkaistiin HMI-projekti, joka sisälsi venttiilin ohjaukseen tarkoitetun ohjausikkunan. Ohjausikkuna on nähtävissä kuvassa 29 näkyvän teollisuustietokoneen näytöllä. HMI-projektin julkaisua ei saatu toimimaan ulkoiselta koneelta, joten linjan teollisuustietokoneeseen asennettiin TwinCAT HMI-engineering. Lisäksi ADS asetuksia jouduttiin muokkaamaan kohtaa AmsNetId, jotta yhteys HMI-serveriin saatiin toimimaan. AmsNetId on paikallisen tietokoneen osoite TwinCAT-verkossa, ja se muodostuu oletuksena järjestelmän, tässä tapauksessa on teollisuustietokoneen, IP-osoitteesta lisäämällä osoitteen loppuun ”1.1”. Tämän jälkeen linjastoon kytkettiin virta ja sen turvatoiminnot kuitattiin, jolloin laitteisto oli valmis testausta varten. Ohjausikkuna avattiin, painettiin ohjausikkunan Manual-nappia, joka ohjasi laitteen manuaaliajolle, jonka jälkeen nostopöytää ohjattiin ylös ja alas napeilla D1 ja D2. Nostopöytä liikkui, kuten oli odotettu, eli ohjaus toimi suunnitellusti.

7 Yhteenveto ja pohdinta

Opinnäytetyön tarkoituksena oli helpottaa yrityksessä yleistyvien Beckhoff projektien ohjelmointia luomalla pohja standardoiduille toimilaitelohkoille sekä niiden ohjaukselle. Työ alkoi aluksi pelkkien toimilaitelohkojen kääntämisellä, mutta se laajeni koskemaan myös niiden ohjausta koska yhteyttä HMI:n ja logiikan välillä ei ollut mahdollista toteuttaa samalla tavalla kuin TIA Portalissa. Ilman yhteyttä HMI:n ja logiikan välillä, ei laitteita pystyisi ohjaamaan manuaalisesti. Lisäksi ohjelmointia TwinCATilla pyrittiin yhtenäistämään TIA Portalin kanssa, jotta siirtyminen ohjelmointialustojen välillä olisi helpompaa. Tuloksena saatiin konkreettinen esimerkki, jonka pohjalta on mahdollista ohjelmoida standardisoituja toimilaitelohkoja sekä niiden ohjausta. Myös manuaaliohjauksen toteutusta HMI-paneelissa saatiin tuotua lähemmäs TIA Portalin mallia ottamalla käyttöön ponnahdusikkunat. Ponnahdusikkunat selkeyttävät käsiajosivujen sisältöä ja lisäksi ne mahdollistivat helpommin konfiguroitavan tavan yhdistää toimilaitelohkot ja niiden ohjausikkunat toisiinsa.

Opinnäytetyön aihe vaati laajaa perehtymistä TwinCAT ohjelmointialustaan, mitä vaikeutti aikaisemman kokemuksen ja kattavan dokumentaation puute. Yrityksen sisällä osaamista TwinCATin käyttöön oli vain vähän. Erilaisia toteutustapoja esimerkiksi toimilaitelohkojen ohjaamiseen joutuikin kokeilemaan hyvin pitkälle, ennen kuin lopullinen ratkaisu varmistui. Arvokasta apua ja osaamista saatiin Beckhoffin tukipalvelusta, jonka järjestämät koulutukset ja tukikäynti olivat ainoat opastetut tutustumiset TwinCATilla ohjelmoimiseen.

Työn aikana käännettiin vain yksi esimerkkilohko TIA Portalista TwinCATille, joten jatkokehitysaiheena olisi koko toimilaitelohkokirjaston kääntäminen TwinCATille. Myös kaikkien lohkojen HMI-rajapintaan yhdistettävien muuttujien struktuurit olisi hyvä toteuttaa ja lisätä kirjastomuotoon. HMI-paneelin käsiohjauspohjan muutos olisi aiheellinen ponnahdusikkunoiden käyttöönoton vuoksi. Lopuksi Esysillä käytössä olevan Hurauttajan integrointi TwinCATille tai uuden vastaavan ohjelman ohjelmointi voisi myös nopeuttaa ja helpottaa TwinCATilla ohjelmointia huomattavasti.

Lähteet

Beckhoff. 2023. TwinCAT automation software. Tuotesivu. Viitattu 16.1.2023. Saatavissa https://www.beckhoff.com/en-en/products/automation/twincat/#stage-special-item-s320986-1_t0

Beckhoff. Beckhoff Information System. Käyttöohje. Viitattu 13.2.2023. Saatavissa https://infosys.beckhoff.com/index_en.htm

Bosch Rexroth Corporation. 2009. Understanding the IEC61131-3 Programming Languages. Nettiartikkeli. Viitattu 16.12.2022. Saatavissa https://www.boschrexroth.com/fi/fi/tuotteet/tuoteryhmaet/saehkoekaeytoet-ja-ohjausjaerjestelmaet/aihealueet/ohjelmoitava-logiikka-plc/

Esys. 2020. Yrityksen verkkosivut. Viitattu 5.1.2023. Saatavissa https://www.esys.fi/

Groover, M.K. 2013. Automation, production systems, and computer-integrated manufacturing. Pearson uusi kansainvälinen painos. Lontoo: Pearson Education.

IEC 61131-3. 2013. Programmable controllers – Part 3: Programming languages. Geneve: International Electrotechnical Commission.

Lydon, B. 2018. User-created function blocks increase productivity. International Society of Automation. Nettiartikkeli. Viitattu 17.2.2023. Saatavissa https://www.isa.org/intech-home/2018/september-october/departments/user-created-function-blocks-increase-productivity

Lee, J.D. & See, K.A. 2004. Trust in Automation: Designing for Appropriate Reliance. Human Factors 46, 50–80.

PLCopen. Status IEC 61131-3 standard. Verkkodokumentti. Viitattu 14.4.2023. Saatavissa https://plcopen.org/status-iec-61131-3-standard

PROFIBUS Nutzerorganisation e.V. 2022. Viitattu 24.1.2023. Saatavissa https://www.profibus.com/download/profidrive-profile-drive-technology

Roininen, J. 2020. TIA HURAUTTAJA. Esys intra. Viitattu 24.3.2023.

Sey, D. & Sen, S.K. 2020. Industrial Automation Technologies. Boca Raton: CRC Press.

Siemens. 2013. How do you program the PLC with STEP 7 (TIA Portal) in compliance with the IEC 61131-3 standard? Sovellusesimerkki. Viitattu 4.1.2023. Saatavissa https://support.industry.siemens.com/cs/document/50204938/how-do-you-program-the-plc-with-step-7-(tia-portal)-in-compliance-with-the-iec-61131-3-standard-?dti=0&lc=en-US

Siemens. Kohti digitaalista tehdasta. Yrityksen verkkosivut. Viitattu 9.1.2023. Saatavissa https://www.siemens.com/fi/fi/tuotteet/teollisuus/tia-portal.html

Siemens. 2018a. Automation of Sequential Processes with GRAPH in the TIA Portal for S7-1500. Käyttöohje. Viitattu 5.1.2023. Saatavissa https://support.industry.siemens.com/cs/document/109759822/automation-of-sequential-processes-with-graph-in-the-tia-portal-for-s7-1500?dti=0&lc=en-US

Siemens. 2018b. Programming Guideline for S7-1200/1500. Käyttöohje. Viitattu 23.1.2023. Saatavissa https://support.industry.siemens.com/cs/document/90885040/programming-guideline-for-s7-1200-s7-1500?dti=0&lc=en-US

Siemens. 2020. SINAMICS G120C Converters. Käyttöohje. Viitattu 24.1.2023. Saatavissa https://support.industry.siemens.com/cs/document/109782995/sinamics-g120c-converters?dti=0&lc=en-FI

Siemens. 2022. SIMOTION Motion Control System. Tuotesivu. Viitattu 9.1.2023. Saatavilla https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10000433