Kemi Tornion ammattikorkeakoulu

Logiikkaohjelmien kirjaston luominen Step 7 -ympäristöön

Toni Perälä

Tekniikan opinnäytetyö
Sähkötekniikan koulutusohjelma
Insinööri (AMK)

KEMI 2013

ALKUSANAT

Haluaisin kiittää PLC-Automation Oy:n toimitusjohtajaa Lauri Kerästä mahdollisuudesta tehdä tämä opinnäytetyö. Suuri kiitos kuuluu myös opinnäytetyön ohjaajalle Tapani Ruokaselle.

Oulussa 10.12.2013

Toni Perälä

TIIVISTELMÄ

KEMI-TORNION AMMATTIKORKEAKOULU, Tekniikka

Koulutusohjelma:	Sähkötekniikan koulutusohjelma
Opinnäytetyön tekijä:	Toni Perälä
Opinnäytetyön nimi:	Logiikkaohjelmien kirjaston luominen STEP 7 -ympäristöön
Sivuja (joista liitesivuja):	47 (15)
Päiväys:	11.12.2013
Opinnäytetyön ohjaajat:	DI Tapani Ruokanen, Kemi-Tornion AMK Ins. Aila Petäjäjärvi, Kemi-Tornion AMK Ins. Lauri Keränen, PLC-Automation Oy

Tämän opinnäytetyön toimeksiantajana toimi PLC-Automation Oy. Työn tavoitteena oli luoda yrityksen yhteiseen käyttöön logiikkaohjelmien kirjasto STEP 7 -ympäristöön. Työhön kuului myös käyttöohjeiden kirjoittaminen kerätyille ohjelmille. Päämääränä oli luoda yrityksen käyttöön mahdollisimman kattava logiikkaohjelmien kirjasto. Kirjaston materiaali kerättiin yrityksessä STEP 7:n parissa työskenteleviltä työntekijöiltä sekä itsenäisesti yrityksen projekteista. Kirjaston sisältö rakennettiin yrityksessä yleisimmin käytetyistä logiikkaohjelmista.

Opinnäytetyön teoriaosuudessa käsiteltiin S7-300- ja S7-400 -ohjelmoitavien logiikoiden toimintaa ja rakennetta. Lisäksi perehdyttiin STEP 7 versio 5.5:n eri käyttötarkoituksiin kehiteltyihin ohjelmiin ja ohjelmointikieliin.

Ohjelmoitavat logiikat ovat nykyaikaisen automaation perusta. Siemensillä on laaja valikoima kaikkiin käyttötarkoituksiin olevia ohjelmoitavia logiikoita. Työssä logiikkaohjelmien kirjasto luotiin STEP 7 versio 5.5:llä, joka on pääosin tarkoitettu Simatic S7-300- ja S7-400 -sarjojen ohjelmoitaville logiikoille. Aineistona työssä käytettiin Siemensin teollisuuden teknisen tuen verkkosivuja, Simatic-käsikirjoja ja ohjelmointiin liittyvää kirjallisuutta.

Työssä saatiin luotua hyvä perusta logiikkaohjelmien kirjastolle, jota tulevaisuudessa pystytään helposti täydentämään ja päivittämään.

Asiasanat: automaatio, ohjelmoitava logiikka, kirjasto, ohjelmointikieli.

ABSTRACT

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES, Technology

Degree Programme:	Electrical Engineering
Author:	Toni Perälä
Thesis Title:	Logic programs library in STEP 7 –environment for an automation company
Pages (of which appendixes):	47 (15)
Date:	11 December 2013
Thesis Instructors:	Tapani Ruokanen, MSc. (Tech) Aila Petäjäjärvi, BEnc. (Tech) Lauri Keränen, Engineer, PLC-Automation Oy

This thesis was made for PLC Automation Oy. The purpose of this thesis was to create a comprehensive library for logic programs in to STEP 7 –environment, including program instructions. The material to the library was collected from company's employees and independently from projects of the company. The content of the library was built for most commonly used logic programs.

The theory of this thesis consists of S7-300- and S7-400 -programmable logic controllers functions and structures. The theory also includes different softwares and programming languages of STEP 7 version 5.5.

Programmable logic controllers are the basis of the modern automation. Siemens has a wide selection of programmable logic controllers for all purposes. The logic programs library was made by STEP 7 version 5.5 which is a software for Simatic S7-300- and S7-400 -series. The material for this thesis was collected from Siemens industry online support web pages, Simatic manuals and programming literatures.

In the work, a good foundation to the library was created. The library was made in a way that the company can effortlessly supplement and upgrade it in the future.

Keywords: automation, programmable logic controller, library, programming language.

SISÄLLYS

KÄYTETYT MERKIT JA LYHENTEET

PLC	Ohjelmoitava logiikka, Programmable Logic Controller
CPU	Keskusprosessori, Central Processing Unit

1 JOHDANTO

Tämä työ suoritettiin oululaiselle sähkö- ja automaatioalan yritykselle PLC-Automation Oy:lle. Yritys on erikoistunut erityyppisiin teollisuusalojen teknologiaratkaisuihin, kuten esimerkiksi automaatiojärjestelmien modernisointiin. Yritys on osa PLC-Yhtiöt konsernia. Konserni aloitti toimintansa 1988, jolloin perustettiin insinööritoimisto PLC-Plan Oy ja sähkö- ja automaatioasennuksiin keskittyvä PLC-Sähkö Oy. PLC-Automation Oy:n lisäksi vuosien aikana konserniin on perustettu kaksi muuta yritystä: kappaleen käsittelyyn erikoistunut PLC-Optimi Oy ja rakennusliiketoimintaan erikoistunut PLC-Team Oy. Laajan toimintansa ansiosta yhtiö pystyy tarjoamaan asiakkailleen tarvittaessa koko järjestelmän elinkaaren aikaisen toiminnan. PLC-Yhtiöllä on asiakkaita usealta eri teollisuuden alalta ja se toimii sekä kotimaisilla että ulkomaisilla markkinoilla. Sen suurimpia asiakkaita ovat muun muassa Stora Enso, UPM, Outokumpu, Ruukki ja M-Real. (PLC-Yhtiön www-sivut 2012, hakupäivä 28.10.2013.)

Opinnäytetyön aiheena oli luoda logiikkaohjelmille kirjasto Siemensin STEP 7 -ympäristöön. Ohjelmakirjasto tuli yrityksen yhteiseen käyttöön. Kirjaston tarkoitus oli koostua yrityksen työntekijöiden yleisimmin käyttämistä Siemens Simatic S7-300- ja S7-400 -ohjelmoitavien logiikoiden ohjelmista. Materiaali kirjastoon kerättiin yrityksen projekteista sekä haastattelemalla työntekijöitä. Kirjastosta pyrittiin luomaan mahdollisimman kattava kokonaisuus. Kerätyille ohjelmille tuli myös kirjoittaa selkeät käyttöohjeet. Kirjaston avulla yrityksen työntekijöiden ajankäyttö tulee tehokkaammaksi. Lisäksi tämän yhtenäisen toimintatavan myötä yrityksen projektit toteutuvat samassa linjassa.

Simatic yhdistää kaikki automaatioratkaisujen osajärjestelmät yhdenmukaiseksi automaatiojärjestelmäksi, aina kenttätasolta prosessiohjaukseen. Tätä kutsutaan TIA:ksi, joka tulee englanninkielen sanoista Totally Integrated Automation. TIA käsittelee yhtenäisen konfiguroinnin, ohjelmoinnin, tiedon hallinnan ja koko automaatiojärjestelmän viestinnän. Opinnäytetyön teoreettinen osa rakentui Simatic S7-300- ja S7-400 -sarjan ohjelmoitavista logiikoista, Simatic STEP 7 V5.5 ohjelmistosta ja sen ohjelmointikielistä.

2 S7-300/400 OHJELMOITAVAT LOGIIKAT

Ohjelmoitavalla logiikalla eli PLC:llä tarkoitetaan teollisuuden tietokonetta, jonka avulla automatisoidaan tiettyjä teollisuuden prosesseja, kuten kuljettimia ja toimilaitteita. PLC:n kautta kulkee kenttälaitteiden ohjaukset, tehtävät ja kentältä tulevat mittaukset.

S7-300 ja S7-400 ovat tällä hetkellä Siemensin yleisimmät käytössä olevat ohjelmoitavat logiikat. Ne ovat rakenteeltaan modulaarisia, joten käyttäjä pystyy itse kokoamaan laajasta yksikkövalikoimasta käyttötarkoituksiinsa tarvittavat yksiköt. (Siemens www-sivut 2013, hakupäivä 23.10.2013.)

2.1 Komponentit

Simatic S7-300- ja S7-400 -PLC:t ovat modulaarisia ja käsittävät seuraavat komponentit:

- Kehikko
  Alumiininen profiilikisko, johon eri yksiköt kiinnitetään.

- Virtalähde (PS)
  Muuntaa verkkojännitteen logiikoiden vaatimaksi 24 VDC käyttöjännitteeksi.

- Prosessori (CPU)
  Säilöö ja suorittaa käyttäjäohjelmaa.

- Liitäntäyksiköt (IM)
  Yhdistävät kehikot keskenään.

- Signaaliyksiköt (SM)
  Sovittavat järjestelmän signaalit prosessisignaaleiksi tai ohjaimien kautta digitaalisiksi ja analogisiksi signaaleiksi.

- Toimintayksiköt (FM)

Suorittavat monimutkaiset tai lyhyttä aikaa vaativat prosessisignaalitoiminnot itsenäisesti.

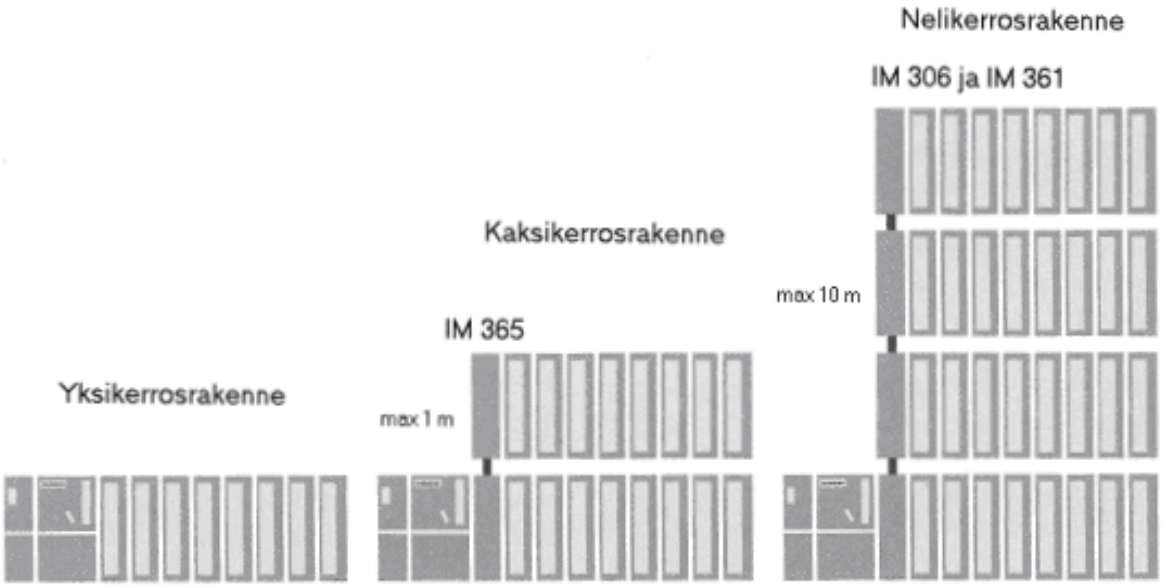
- Kommunikaatioprosessori (CP)
Luo yhteyden aliverkkoihin, toisiin logiikkoihin.

Ohjelmoitavat logiikat voivat koostua useista kehikoista, jotka ovat yhteyksissä toisiinsa väyläkaapelilla. Virtalähde, CPU ja I/O-moduulit liitetään keskuskehikkoon. I/O-moduuleita on kuitenkin mahdollista erottaa keskuskehikosta liitäntäyksiköitä käyttämällä. Pääyksikköön on myös mahdollista yhdistää hajautettuja I/O:ta. (Berger 2012, 20.)

Kehikot yhdistävät moduulit kahdella väylällä. I/O-väylä on suunniteltu nopeille vaihtuville tulo- ja lähtösignaaleille. Kommunikaatioväylä on taas suunniteltu siirtämään suuria määriä tietoa. Väylä yhdistää prosessorin ja ohjelmoitavan laitteen toimintayksikköön ja kommunikaatioprosessoriin. (Berger 2012, 20.)

2.2 S7-300

S7-300 yhteydessä puhutaan keskitetystä I/O:sta. Tällä tarkoitetaan, että tulo- ja lähtösignaaliyksiköt liitetään logiikkaohjaimen kanssa samaan kehikkoon. Yhteen kehikkoon voidaan lisätä enintään kahdeksan tulo- ja lähtösignaaliyksikköä. S7-300 -logiikkaohjaimessa voi keskuskehikon lisäksi olla enintään kolme laajennuskehikkoa (Kuva 1). Laajennuskehikkoja käytettäessä etäisyys logiikkaohjaimeen on hyvin rajoitettu. Kaksikerrosrakenteisessa järjestelmässä käyttäen IM 365-moduulia etäisyys voi kehikkojen välillä olla enintään metrin. Nelikerrosrakenteisessa järjestelmässä, käyttäen IM 360- ja IM 361-moduulia, logiikkaohjaimen ja viimeisen kehikon välillä voi olla enintään 10 metriä. (Berger 2012, 20.)



Kuva 1. S7-300 modulaarinen rakenne (Berger 2012, 21)

2.2.1 CPU:t

S7-300 prosessorit (CPU:t) jaetaan käyttötarkoituksen mukaan tavallisten prosessorien lisäksi kolmeen eri luokkaan. Turvatekniset CPU:t (Kuva 2) mahdollistavat tavallisen kahden eri järjestelmän integroimisen. Varsinaisesti toimivaan logiikkaan voidaan yhdistää erillinen turvalogiikka. Tämän avulla saadaan järjestelmä jossa on otettu huomioon koneturvallisuuden viranomaismääräykset. (Siemens www-sivut 2013, hakupäivä 23.10.2013.)

Kuva 2. Turvatekninen CPU (Siemens www-sivut 2013, hakupäivä 23.10.2013)

Kompaktit CPU:t (Kuva 3) sisältävät prosessorin lisäksi siihen liitettyjä tulo- ja lähtöpiirejä. Näitä on saatavilla eri malleja sisältäen pelkkiä digitaalisia tulo- ja

lähtösignaaliyksiköitä tai mukana voi olla myös analogisia signaaliyksiköitä. Näiden yksiköiden lisäksi ohjaimeen on lisätty sisäänrakennettuja funktioita. (Siemens www-sivut 2013, hakupäivä 23.10.2013.)

Kuva 3. Kompakti-CPU (Siemens www-sivut 2013, hakupäivä 23.10.2013)

Ulkonäöltään kompaktien prosessorien näköiset teknologia CPU:t on kehitetty etupäässä liikkeenohjaussovelluksiin. Näihin ohjaimiin on lisätty kattavat funktiokomennot sekä niiden toiminta on optimoitu nopeita ohjauksia vaativiin paikoituksiin. (Siemens www-sivut 2013, hakupäivä 23.10.2013.)

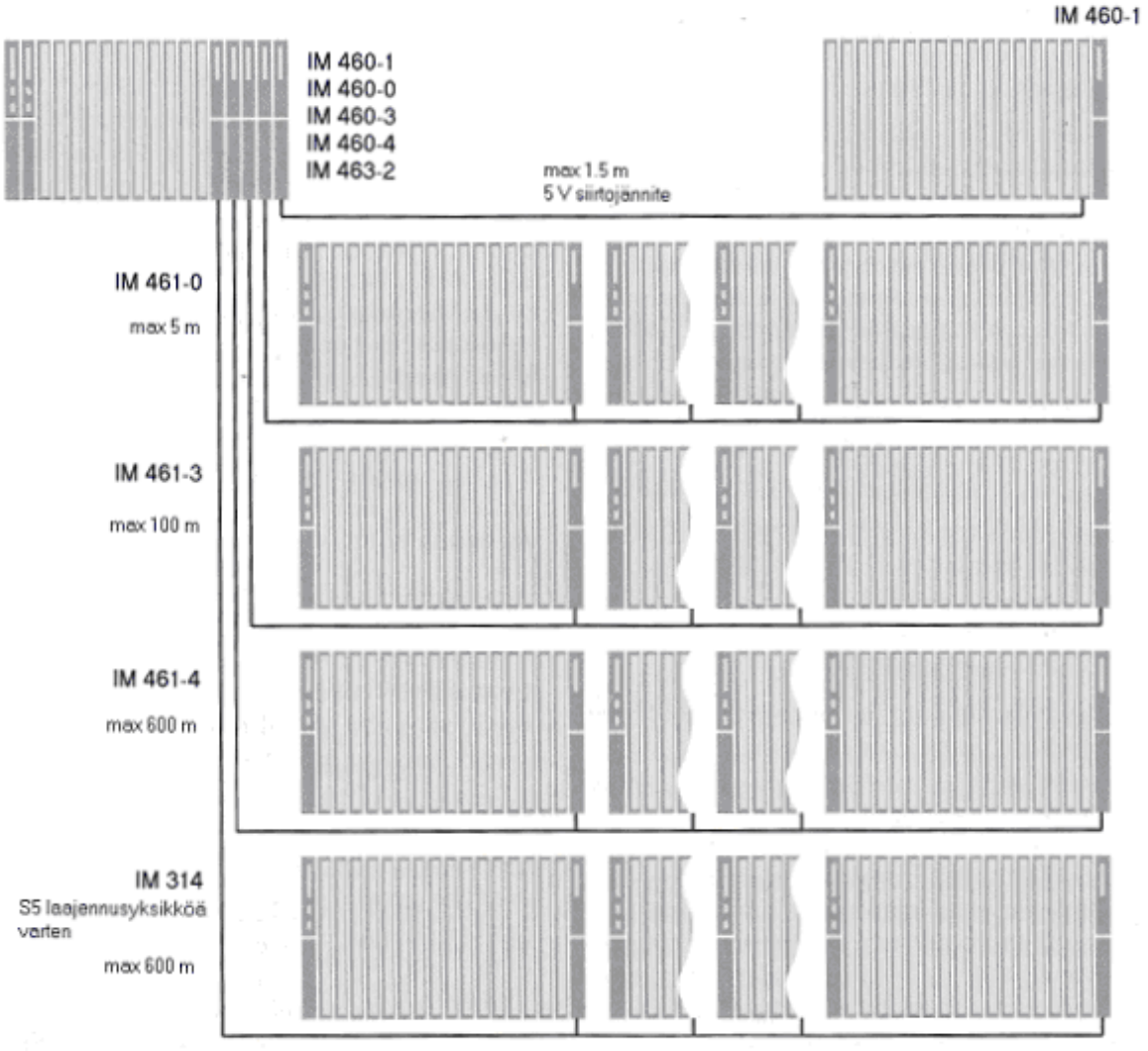
2.3 S7-400

S7-400 -logiikkaohjaimet ovat rakenteeltaan huomattavasti isokokoisempia verrattuna S7-300 -sarjaan. Ne ovat myös Siemensin TIA-konseptin tehokkaimmat ohjaimet. S7-400 -sarjaan on saatavilla kolmea erikokoista ohjainkehikkoa: UR1 sisältää 18 korttipaikkaa, UR2 sisältää 9 korttipaikkaa ja CR3 neljä korttipaikkaa. UR1:stä ja UR2:sta voidaan käyttää myös laajennuskehikoissa. Tulo- ja lähtösignaalikorttien lisäksi virtalähde ja prosessori vaativat myös korttipaikkansa. Ne voivat tyypistä riippuen viedä useamman kuin yhden korttipaikan. (Siemens www-sivut 2013, hakupäivä 28.10.2013.)

Käyttämällä IM 460-1 ja IM 461-1 liitäntämoduuleita voidaan laajennuskehikko viedä 1,5 metrin päähän keskuskehikosta. Yhteen liitäntämoduuliin voidaan liittää yksi laajennuskehikko, jossa kulkee myös 5 V siirtojännite.

IM 460-0 ja IM 461-0 käyttämällä voidaan laajennuskehikot viedä viiden metrin päähän keskuskehikosta. Yhteen liitäntämoduuliin voidaan lisätä peräti neljä laajennuskehikkoa. (Berger 2012, 22 - 23.)

Liitäntämoduuleilla IM 460-3 ja 461-3 tai IM 460-4 ja 461-4 voidaan laajennuskehikko viedä 100 metriin tai jopa 600 metriin asti. Laajennuskehikkoja voidaan lisätä neljä yhteen liitäntämoduuliin (Kuva 4). Liitäntämoduuleita käyttämällä keskuskehikkoon on mahdollista liittää enintään 21 laajennuskehikkoa. (Berger 2012, 22 - 23.)



Kuva 4. S7-400 modulaarinen rakenne (Berger 2012, 21)

3 STEP 7 V5.5

STEP 7 V5.5 on Simatic S7-300/400, Simatic ET200 CPU ja Simatic WinAC ohjelmoitavien logiikoiden ohjelmoimiseen tarkoitettu ohjelmistopaketti. Se mahdollistaa käyttäjän käyttää näiden järjestelmien suorituskykyä helposti ja kätevästi. Ohjelmisto sisältää kaikki toiminnot automaatioprojektin luomista varten: laitteiston konfiguroinnin ja parametroinnin, kommunikaation määrittämisen, ohjelmoinnin, testauksen, dokumentoinnin ja käyttö- ja diagnostiikkatoiminnot. STEP 7 V5.5 sisältää Simatic Manager- , S7 Graph-, S7 SCL- ja S7 PLCSIM -ohjelmistot. (Siemens www-sivut 2013, hakupäivä 28.10.2013.)

STEP 7 V5.5 käyttökieliin kuuluu englanti, saksa, ranska, italia ja espanja Lisäksi ohjelmistosta on myös japanin- ja kiinankieliset versiot. Versio 5.5 toimii seuraavilla käyttöjärjestelmillä, MS Windows XP Professional SP2 tai SP3, MS Windows Server 2003 SP2 standard edition, MS Windows 7 32/64-bit Ultimate/Professional/Enterprise tai MS Windows Server 2008 R2 64-bit. (Berger 2013, 62 - 63.)

3.1 Simatic Manager

Keskeisimpänä ja tärkeimpänä työkaluna STEP 7 V5.5:ssä toimii Simatic Manager. Se toimii ohjelmoitavien logiikoiden käyttöliittymänä. Simatic Manageriin luodaan oikeaa laitteistoa vastaavat objektit. Ohjelmalla luotu projekti tulee sisältää kaikki laitteiston osat, PLC:n, CPU:n, DI/DO ja AI/AO-kortit, väyläkaapelit ja mahdolliset hajautetut I/O:t. Ohjelmaan lisättyjen laitteiston osien tunnukset on vastattava oikeiden osien tunnuksia. Huolimatta kohdejärjestelmästä Simatic Manager hallitsee kaiken automaatioprojektiin kuuluvat tiedot. Kohdejärjestelmiä ovat Simatic S7, Simatic C7 ja Simatic WinAC. (Berger 2013, 67 - 68.)

3.1.1 Organisaatiolohko (OB)

OB on ohjelmatason korkein lohko ja sellainen on oltava aina ohjelmassa. Se on keskusprosessorin käyttöjärjestelmän ja käyttäjän välinen liitoskohta. Organisaatiolohkoja käytetään suorittamaan tiettyjä ohjelmaosia. Niihin kirjoitetaan niin sanottuja kutsuja, joita keskusprosessorin käyttöjärjestelmä kutsuu tiettyjen ehtojen

täyttyessä. Niitä ovat esimerkiksi CPU:n käynnistys ja pysäytys sekä virheen esiintyminen.

Tietyillä organisaatiolohkoilla on omat toiminnot ja oletusprioriteetit. Esimerkiksi OB 1 pitää sisällään pääohjelman, OB 81 virtalähdehäiriön ja OB 84 keskusprosessorin kovalevyhäiriön. Liitteessä 1 on esitetty kaikki valmiit toiminnot sisältävät organisaatiolohkot. (Siemens 2006, 13.)

3.1.2 Toimintayksikkölohko (FB)

Toimintayksikkölohkot ovat käyttäjäohjelman parametroitava osa, joka pystyy varastoimaan tietoa sille vakinaisesti osoitetulle tiedostoyksikölle. Tiedon tallentamisella ohjelmaa ei tarvitse kirjoittaa useaan kertaan. Toimintayksikkölohkoon kirjoitetaan yleensä sellaiset ohjelman osat, joita kutsutaan useasti ohjelmakierron aikana. (Berger 2013, 193.)

3.1.3 Toimintalohko (FC)

Toimintalohkoja käytetään ohjelmoitaessa ohjelmia, joissa on usein toistuvia automaatiotehtäviä. Niitä voivat olla esimerkiksi parametrointi ja paluuarvon toimittaminen takaisin toiselle kutsuvalle lohkolle. Toimintalohkon arvo on vapaasti valittavissa. Lisäksi toimintalohkolla voi myös olla lähtöparametreja. Toimintalohkot eivät varastoi informaatiota eikä niillä ole määritettyä tiedostoyksikköä. (Berger 2013, 193.)

3.1.4 Tiedostoyksikkö (DB)

Tiedostoyksiköt sisältävät kaiken tiedon käyttäjäohjelmasta. Tiedostoyksiköstä voi luoda global-tiedostoyksikön, instance-tiedostoyksikön tai type-tiedostoyksikön. Global-tiedostoyksikköön voidaan vapaasti tuoda eri lohkojen tietoa, sekä jokainen lohko voi lukea tämän tiedostoyksikön tietoa, eli sen rakenne on vapaasti valittavissa. Instance-tiedostoyksikön sisältämää tietoa voi käyttää vain sille määrätyt toimintayksikkölohkot. Type-tiedostoyksikön rakenne perustuu käyttäjän määrittelemään tiedostotyyppiin. (Berger 2013, 193.)

3.2 Simatic S7-SCL

SCL on STEP 7:n S7-300/400 logiikkaohjainsarjoille tarkoitettu ohjelmointiohjelma. SCL tulee englannin kielen sanoista Structured Control Language. Ohjelman ohjelmointikieli on tekstipohjainen ja se soveltuu erityisesti ohjelmiin, jotka sisältävät monimutkaisia algoritmisia ja aritmeettisia ohjelmatehtäviä. S7-SCL on standardin IEC 61131-3 mukainen ja se vastaa standardin määrittelemää ST ohjelmointikieltä. (Siemens www-sivut 2013, hakupäivä 12.11.2013.)

3.3 Simatic S7-GRAPH

S7-GRAPH on Simaticin sekvenssiohjelmointia varten kehitetty ohjelmointiohjelma. Ohjelma jaetaan rinnakkaisiin ja peräkkäisiin askeleisiin, joita yhdistää ohjelmapolku. Ohjelma etenee polkua pitkin askeleeseen ja täytettyään askeleen ehdon se etenee seuraavaan askeleeseen. Askeleita ovat erilaiset ohjelman tehtävät. Ohjelmointitapa on selkeä ja helpottaa prosessiohjausten diagnostiikan rakentamista. Ohjelma on myös standardin IEC 61131-3 mukainen. (Siemens www-sivut 2013, hakupäivä 12.11.2013.)

3.4 Simatic S7-PLCSIM

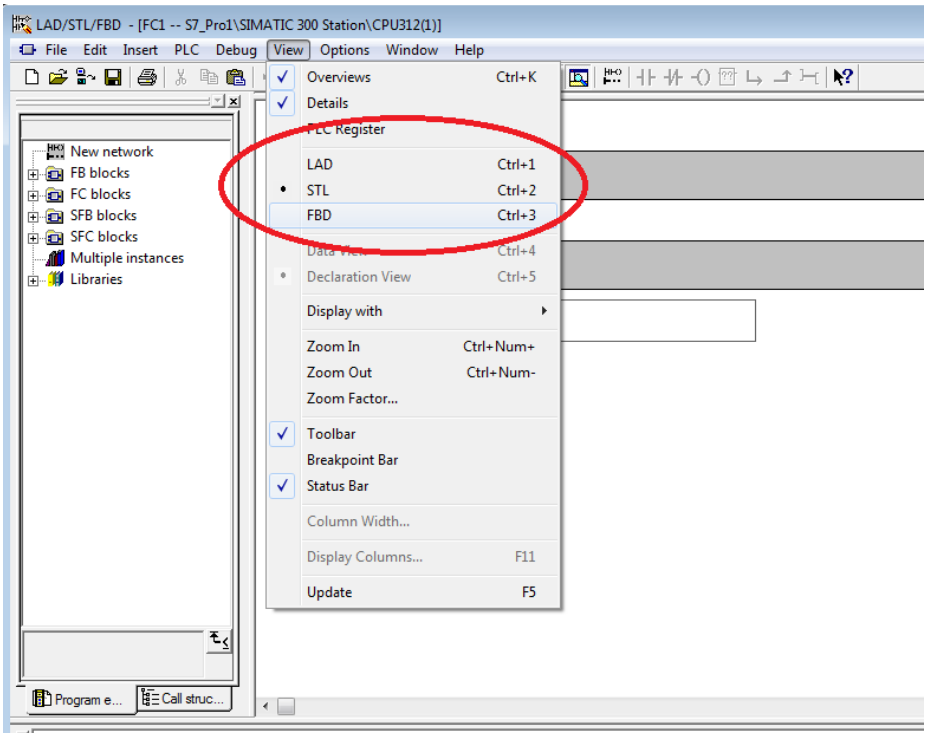
PLCSIM on Simatic Step 7:n simulointityökalu, joka on luotu tilanteita varten, jossa prosessiohjelmia ei pystytä testaamaan paikan päällä. Lisäksi testaaminen ohjelmointivaiheessa nopeuttaa kehitystyötä ja helpottaa mahdollisten ohjelmointivirheiden havaitsemista. Simulointiohjelman avulla voidaan myös optimoida ohjelmien suoritusnopeutta. S7-PLCSIM-ohjelmalla pystytään simuloimaan S7-300- ja S7-400 -sarjan logiikkaohjaimia sekä WinAC:n logiikkaa. Ohjelmalla pystyy simuloimaan montaa ohjainta yhtä aikaa, joka mahdollistaa isojenkin projektien testaamisen. (Siemens www-sivut 2013, hakupäivä 12.11.2013.)

3.5 STEP 7:n integroidut ohjelmointikielet

Simatic Managerissa on mahdollisuus luoda ohjelmia kolmella eri ohjelmointikielellä, LAD:llä, FBD:llä tai STL:llä. Ohjelmointikieli valitaan joko projektia luodessa tai lohkoa ohjelmoitaessa. Tämä ei kuitenkaan tarkoita, että projekti ohjelmoidaan yhdellä

kielellä, vaan projektissa on mahdollista käyttää kaikkia ohjelmointikieliä. Ohjelmointikieli vaihdetaan lohkoa ohjelmoitaessa view-vetovalikosta (Kuva 5).

Myös ohjelmointikielten muuntaminen on osittain mahdollista. Yleensä LAD- tai FBD-ohjelmointikielellä kirjoitetut ohjelmat voidaan muuntaa ongelmitta STL kieleksi. Muuntaessa LAD-kielellä tehtyjä ohjelmia FBD:ksi ja päinvastoin, tulee STL:n näyttöön ohjelmaelementtejä, jotka eivät ole esitettävissä kohdekielellä. (Siemens 1996, 269.)



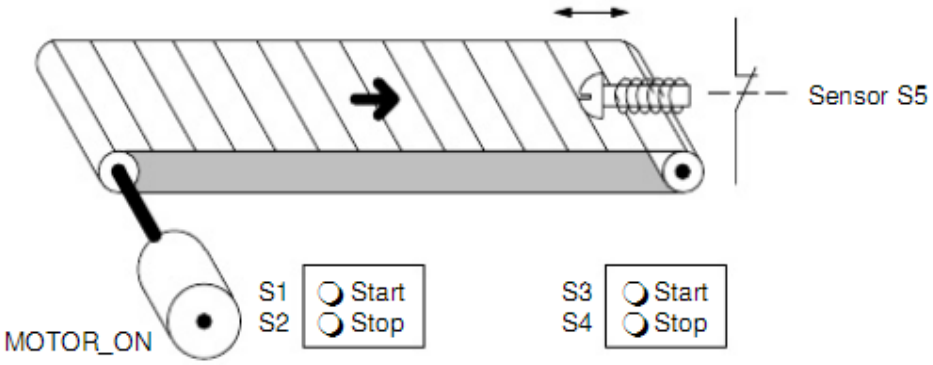
Kuva 5. Ohjelmointikielen valinta

3.5.1 LAD

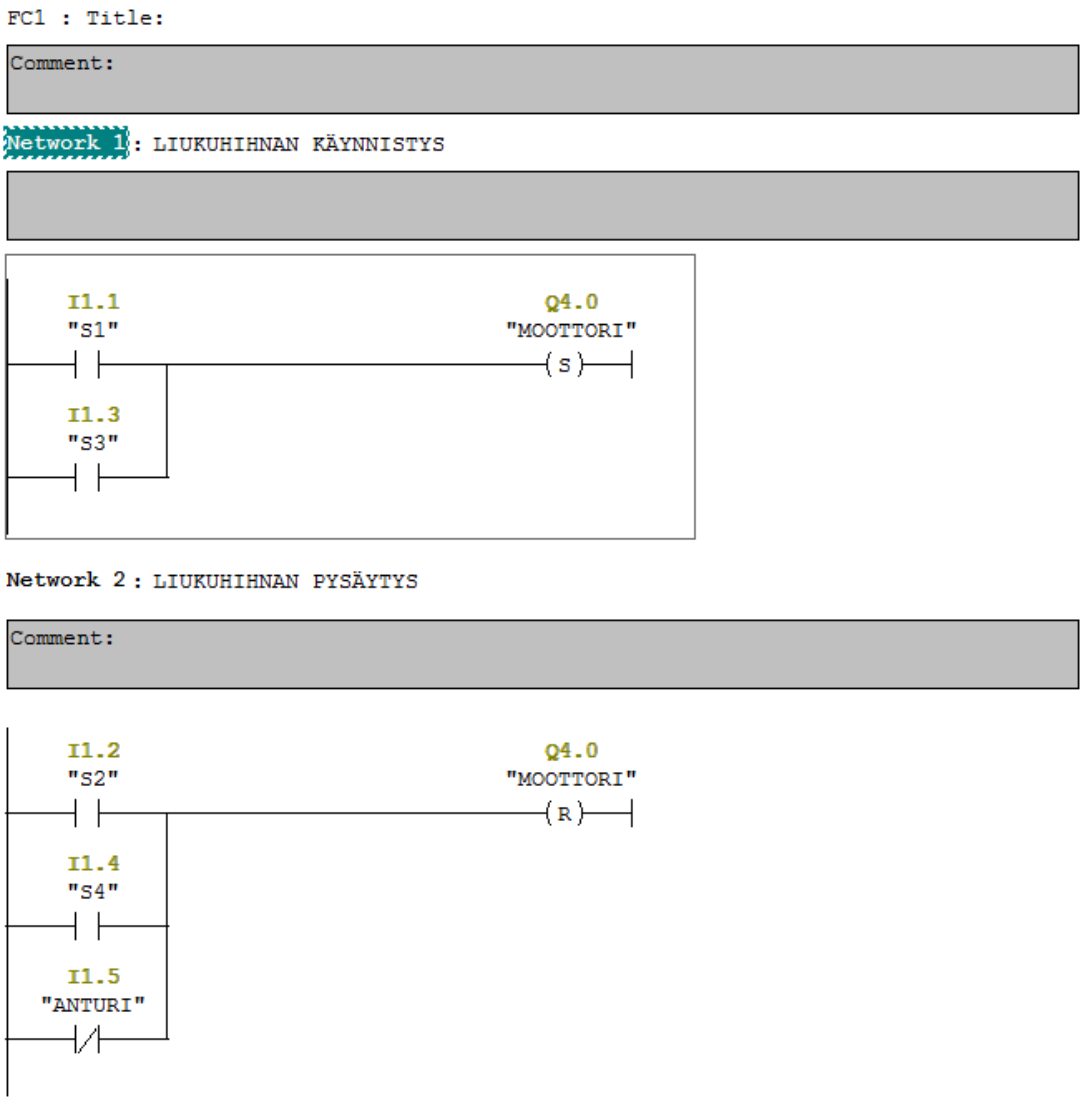
Ulkonäkönsä vuoksi ohjelmointikieltä kutsutaan tikapuukaavioksi ja se muistuttaa paljon ennen käytettyä relekaaviota. Se on graafinen ohjelmointikieli, jossa signaali etenee syklisesti vasemmalta oikealle ja ylhäältä alas. Liitteessä 2 on esitetty kaikki käskyt LAD-ohjelmointikielellä.

Kuva 7 esittää millaiselta näyttää Simatic Managerilla tehty liukuhihnan ohjaus (Kuva 6) käyttäen ohjelmointikielenä LAD:tä. Liukuhihnan molemmissa päissä on ohjauskytkimet, josta liukuhihnan moottori voidaan käynnistää ja pysäyttää. Start-

painikkeiden tulopisteet I1.1 ja I1.3 on nimetty symboleilla S1 ja S3. Stop-painikkeiden tulopisteet I1.2 ja I1.4 on nim etty symboleilla S2 ja S4. Lisäksi kuljettimen päähän on asennettu anturi, jonka tulopiste on I1.5. Anturin kosketin avautuu tavaran kuljettua liukuhihnan päähän. (Siemens 2010, 200.)



Kuva 6. Liukuhihnan ohjaus (Siemens 2010, 200)



Kuva 7. Liukuhihnan ohjaus LAD-ohjelmointikielellä

Ohjelma on jaettu kahteen osioon, joista ensimmäisessä on liukuhihnan käynnistysehdot ja toisessa hihnan pysäytysehdot. Käynnistysosiossa käynnistyspainikkeille on asetettu sulkeutuvat koskettimet. Kun toinen koskettimista sulkeutuu, eli painiketta S1 tai S3 painetaan, asettaa se SET-operandin aktiiviseksi. SET-operandi pitää lähdön Q4.0:n päällä niin kauan kuin RESET-operandi tai sähkökatko sen pysäyttää. RESET-operandi saa signaalin, kun toista pysäytyspainikkeista painetaan tai anturin kosketin avautuu.

3.5.2 STL

STL on Simatic STEP 7:n tekstiohjelmointikieli, jota voidaan käyttää logiikkalohkojen koodiosan luomiseen. Kaikista STEP 7:n ohjelmointikielistä STL on lähimpänä S7-prosessorien käyttämää konekieltä. Kaikki käskyt STL-ohjelmointikielellä on esitetty

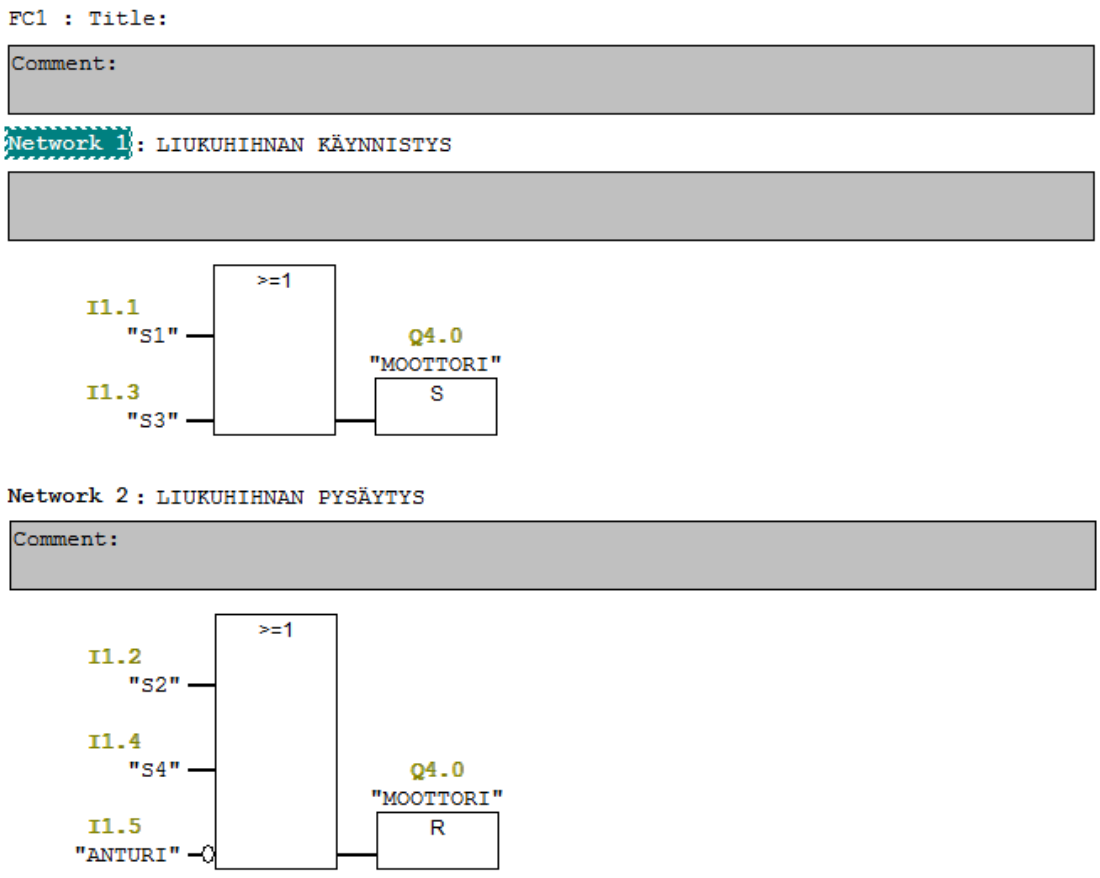
liitteessä 3. Käyttämällä STL-ohjelmointikieltä ohjelmoitaessa S7-logiikkaa saadaan optimaalisin ajoaika ja muistinkäyttö. Ohjelmointikieli koostuu kaiken kaikkiaan 130 eri peruskäskystä, joiden avulla ohjelman luominen onnistuu selkeästi. Seuraavassa kuvassa (Kuva 8) on esitetty kuinka liukuhihnan ohjaus (Kuva 6) tapahtuu STL-ohjelmointikielellä. (Siemens www-sivut 2013, hakupäivä 24.11.2013.)

```
FC1 : Title:
Comment:

Network 1: LIUKUHIHNAN KÄYNNISTYS

      O     "S1"                    I1.1
      O     "S3"                    I1.3
      S     "MOOTTORI"              Q4.0

Network 2 : LIUKUHIHNAN PYSÄYTYS
Comment:

      O     "S2"                    I1.2
      O     "S4"                    I1.4
      ON    "ANTURI"                I1.5
      R     "MOOTTORI"              Q4.0
```

Kuva 8. Liukuhihnan ohjaus STL-ohjelmointikielellä

3.5.3 FBD

FBD eli toimintakaaviodiagrammi on Simatic STEP 7:n graafinen ohjelmointikieli, jossa ohjelmat luodaan Booleaan algebrasta tuttujen loogisten lohkojen muodossa. Se sisältää perusoperaatioiden lisäksi laajan valikoiman parametroituja operandeja. Kaikki käskyt FBD-ohjelmointikielellä on esitetty liitteessä 4. Seuraavassa kuvassa (Kuva 9) on esitettynä liukuhihnan ohjaus (Kuva 6) FBD-ohjelmointikielellä. (Siemens www-sivut 2013, hakupäivä 24.11.2013.)



Kuva 9. Liukuhihnan ohjaus FBD-ohjelmointikielellä

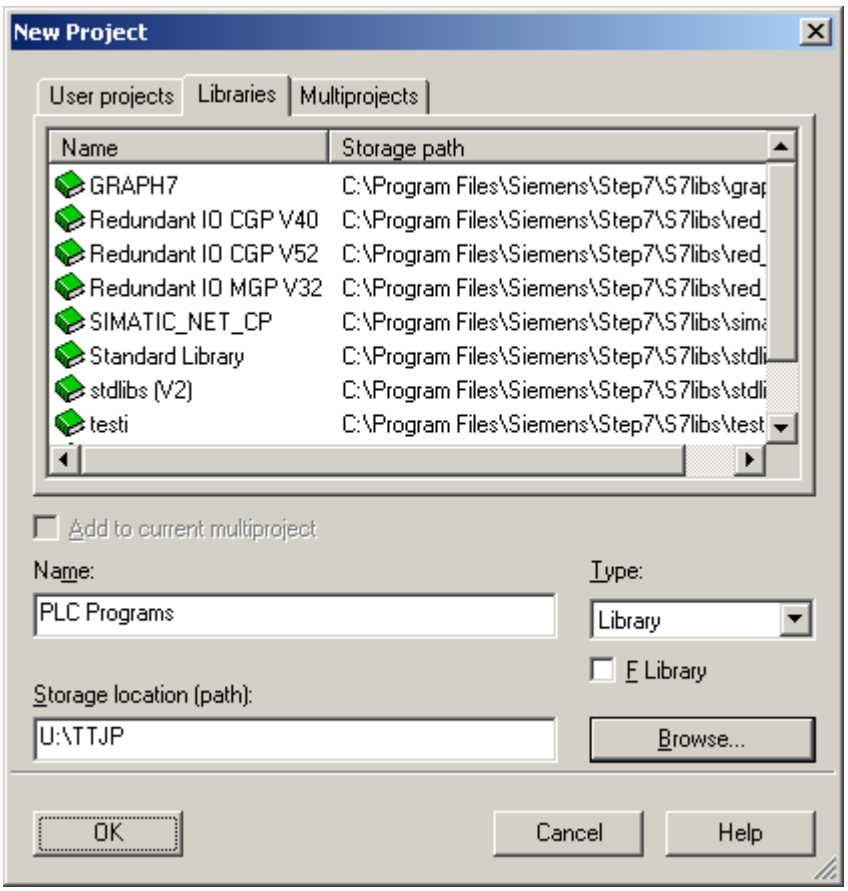
4 KIRJASTON LUOMINEN PLC-AUTOMATION OY:LLE

Työn tarkoituksena oli luoda ohjelmoitavien logiikkaohjelmien kirjasto PLC-Automation Oy:n yhteiseen käyttöön. Kirjaston materiaalin keräsin itsenäisesti erilaisista projekteista sekä haastattelemalla yrityksen työntekijöitä. Tarkoitus oli että kirjastolle saataisiin luotua hyvä pohja, jota pystytään jatkossa päivittämään.

4.1 Kirjaston luominen Simatic Manageriin

Uuden kirjastokomponentin luominen Simatic Managerilla oli helppoa. Simatic Managerin aloitusikkunan File-vetovalikosta valitaan New Project. Tämä onnistuu myös pikanäppäin yhdistelmällä CTRL+N. New Project valikosta valitaan libraries kohta. Tämän jälkeen kirjasto nimetään ja sille määrätään tallennuspaikka (Kuva 10).

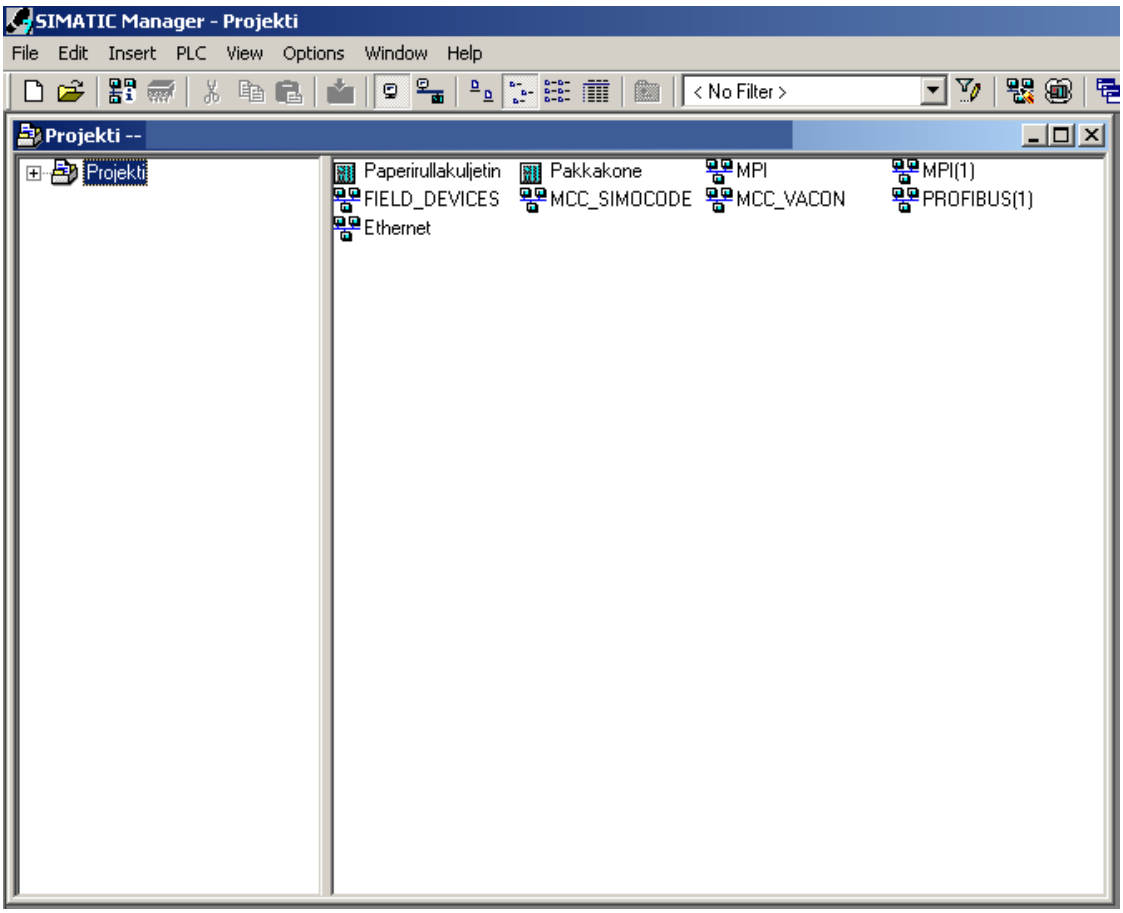
Työssäni kirjastolle annettiin nimeksi PLC Programs ja se tallennettiin yrityksen yhteiselle asemalle, josta jokainen voi hakea sen omalle tietokoneelle. Kirjaston tyypiksi valittiin Standardi-kirjasto, koska tämän tyypin kirjastokomponentti on muokattavissa ja se on ainoa jota tälläisessä tilanteessa voi käyttää. Muita kirjastokomponentteja on F-Library ja Mastet Data Library. F-Libaraya voidaan käyttämään ainoastaan F-järjestelmissä ja Master Data Librarya voidaan käyttää silloin kun tehdään Step 7:n multiprojektia.



Kuva 10. Uuden kirjastokomponentin luominen

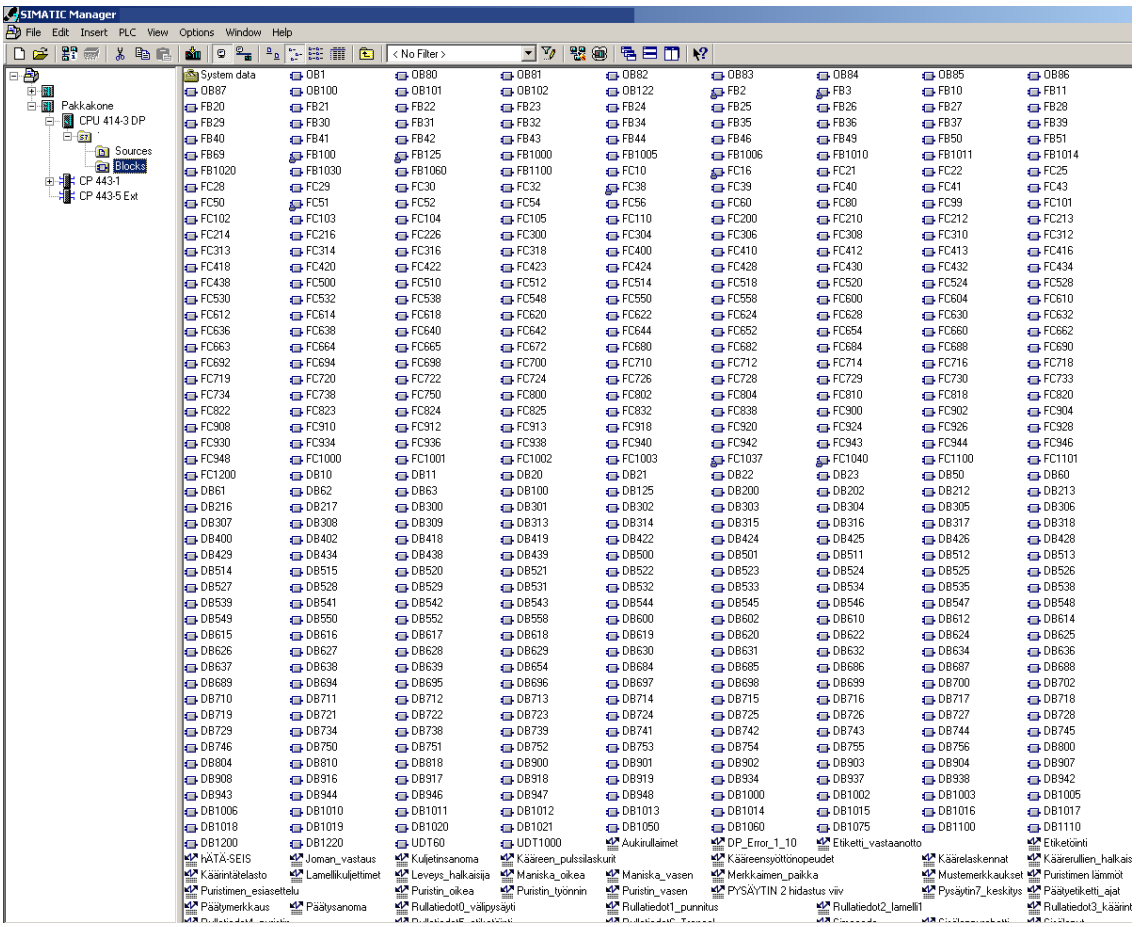
4.2 Esimerkki projektista

Kun kirjastokomponentti oli luotu, aloin tutkia yrityksen tekemiä projekteja. Alla olevassa kuvassa (Kuva 11) on esimerkki erään paperitehtaan ohjelmointiprojektista. Projekti sisältää paperirullakuljettimen ja pakkauskoneen ohjelmat. Molemmilla laitteilla on omat logiikkaohjaimet, jotka on yhdistetty toisiinsa ethernet-väylän avulla. Lisäksi logiikkaohjaimiin on yhdistetty SIMOCODE moottorisuojaukset, taajuusmuuttajat ja kenttälaitteet profibus-väylillä.



Kuva 11. Paperitehdas projekti

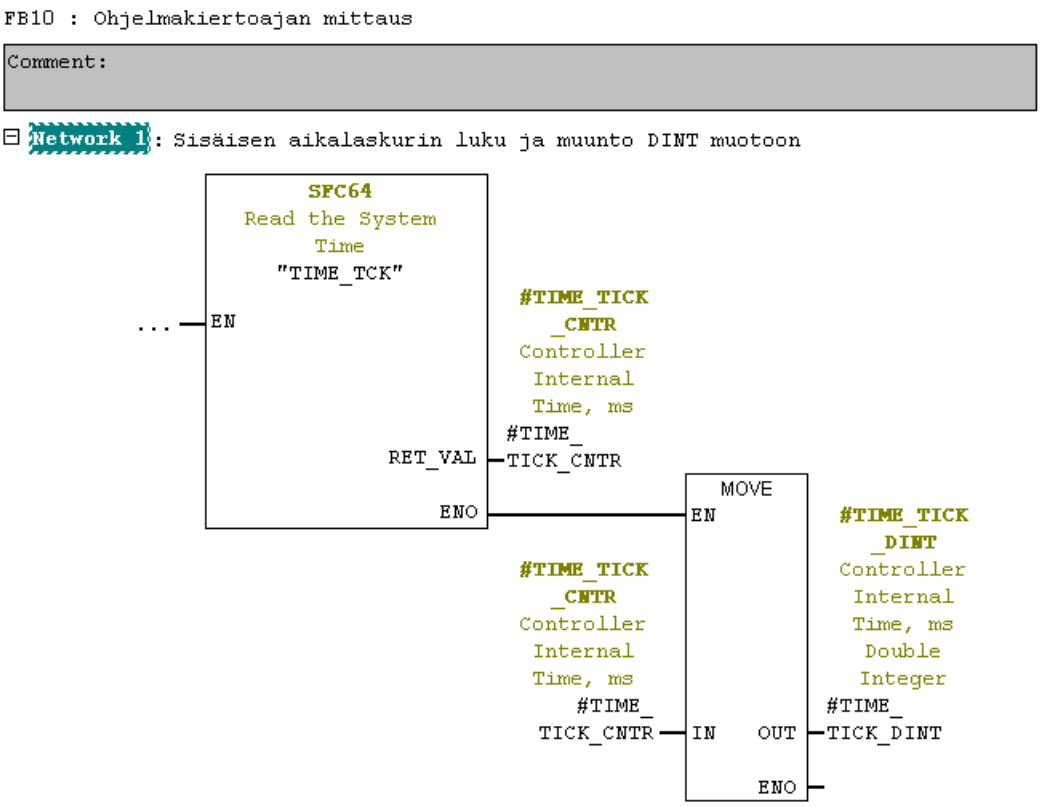
Ohjelmalohkot, joita kirjastoon lisätään, sijaitsevat CPU:n alavalikossa BLOCKS. Seuraavassa kuvassa (Kuva 12) on paperitehtaan projektin pakkauskoneen ohjelmalohkot. Projektit ovat erittäin laajoja kokonaisuuksia ja tämänkin pakkakoneen ohjelma sisältää 13 organisaatiolohkoa, noin 40 toimintayksikkölohkoa (FB), noin 100 toimintalohkoa (FC) sekä useita muuttujatauluja.



Kuva 12. Pakkakoneen ohjelmalohkot

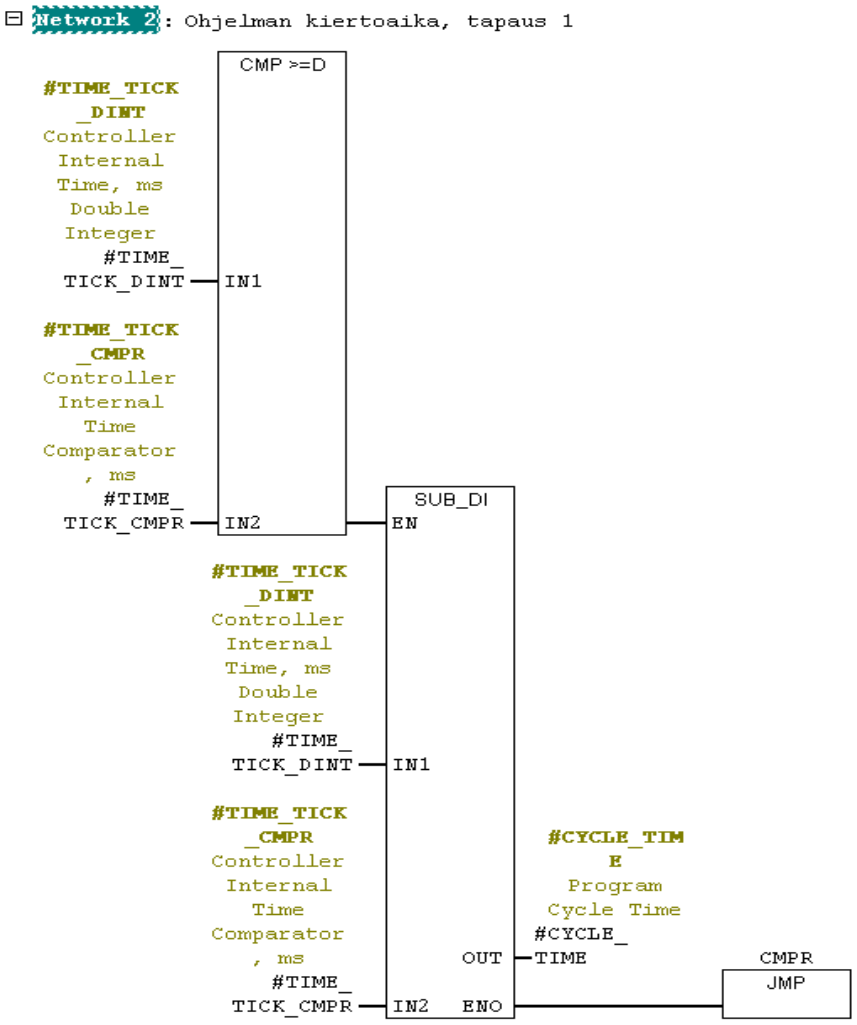
4.3 Esimerkki kirjaston ohjelmasta

Yksi kirjastoon lisätyistä ohjelmista oli ohjelmakiertoajan mittaus. Tämä valittiin kirjastoon, koska tämmöinen ohjelma esiintyi muutamassa muussakin projektissa ja todennäköisesti tätä ohjelmaa käytetään tulevaisuuden projekteissakin. Kuvista 13 - 16 näkyy, millainen ohjelma on kyseessä.



Kuva 13. Ohjelmankiertoajan mittaus network 1

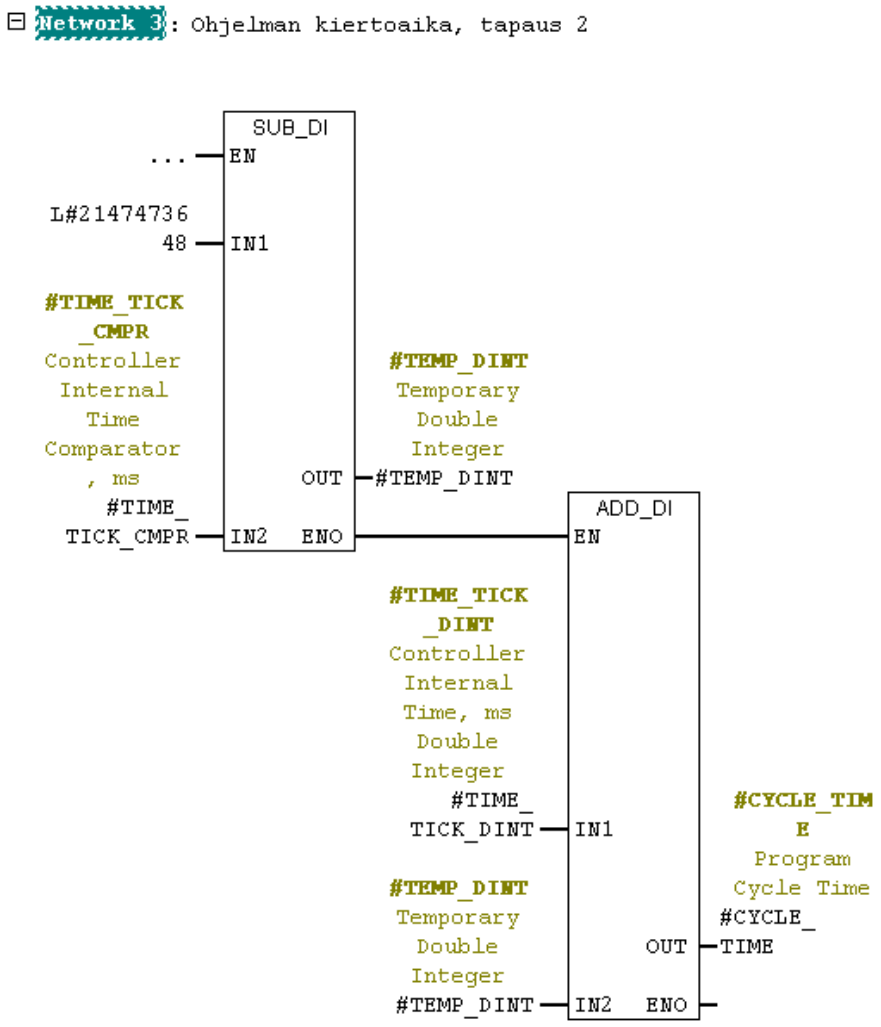
Network 1 (Kuva 13) kohdassa luetaan järjestelmän sisäisen laskurin lukuarvo ja muunnetaan se 32 bittiseksi kokonaisluvuksi, DINT muotoon. SFC 64 ”TIME_TCK” lukee järjestelmäajan keskusprosessorilta. Laskuri kasvaa automaattisesti järjestelmän ohjaamana yhden millisekunnin välein, 0 millisekunnista 2147483647 millisekuntiin.



Kuva 14. Ohjelmankiertoajan mittaus network 2

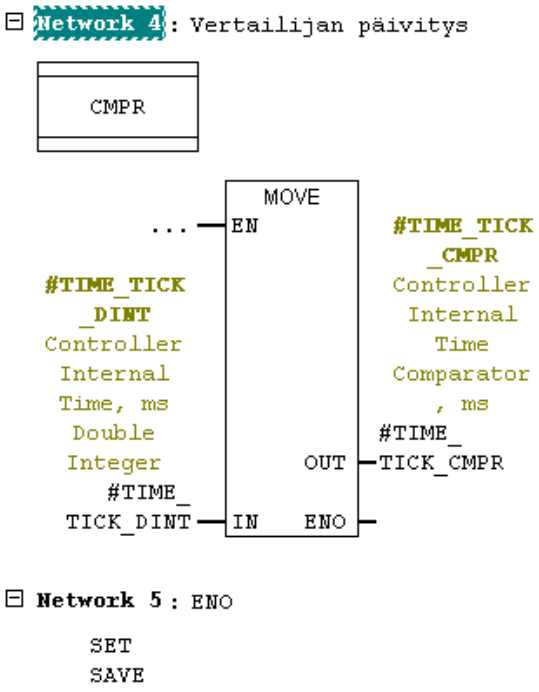
Network 2 (Kuva 14) kohdassa on ohjelman kiertoajan ensimmäinen tapaus. CMP >=D eli compare double integer vertailee kahden kokonaisluvun arvoa. Jos IN1 tulopisteen kokonaisluku on IN2 tulopistettä suurempi tai yhtä suuri, lähdön signaalitila on ”1”. Tässä tilanteessa verrataan sitä, onko sisäinen järjestelmäaika suurempi tai yhtä suuri kuin vertailuaika.

Vertailijasta signaali kulkee lohkoon, jossa suoritetaan kokonaislukujen erotus. Erotuksesta saatu kokonaisluku saadaan lähtöpisteestä OUT. Lisäksi ENO lähtöön on lisätty ehdollinen hyppykäsky CMPR kohtaan, joka löytyy network 4 kohdasta. Tämän käskyn toteutuessa signaali hyppää network 3 kohdan yli.



Kuva 15. Ohjelmankiertoajan mittaus network 3

Network 3 (Kuva 15) kohdassa on ohjelman kiertoajan toinen tapaus. Jos sisäisen aikalaskurin arvo on pienentynyt ohjelmakierron aikana, eli laskuri on kääntynyt ympäri, ohjelma laskee sisäisen laskurin maksimiarvon ja vertailijan erotuksen. Erotuksesta saatu tulos lisätään aikalaskuriin ADD_DI kohdassa.

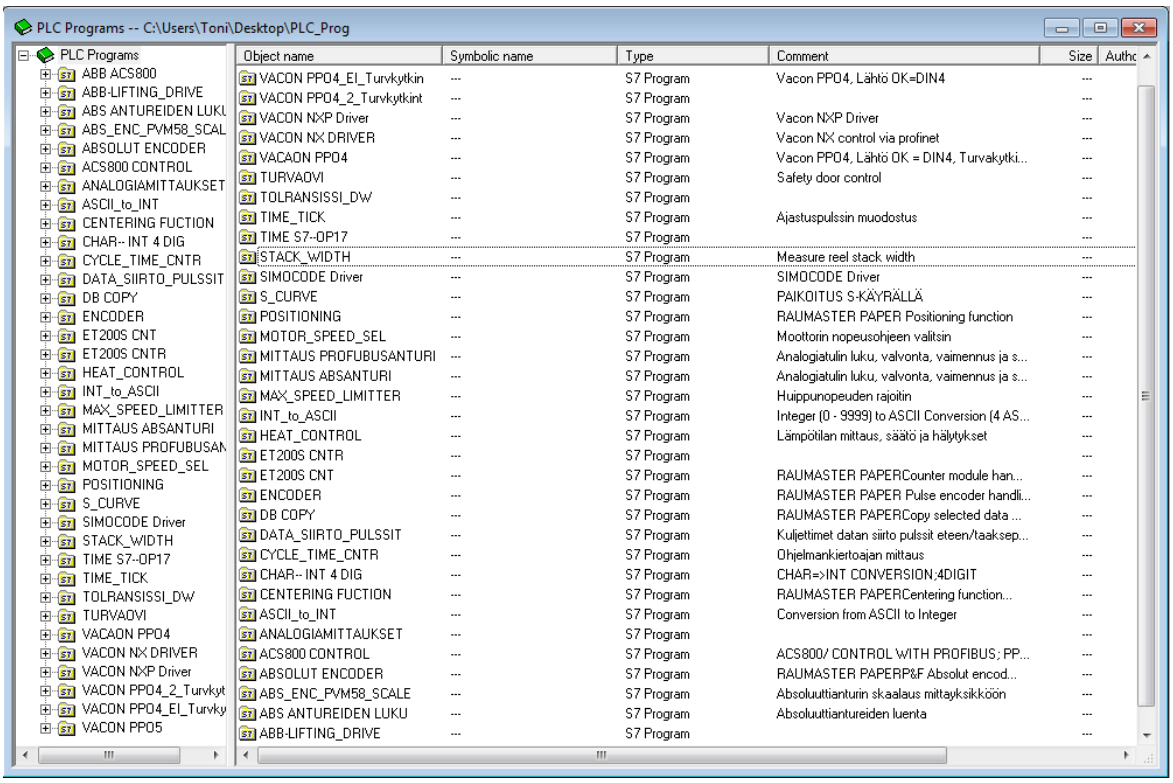


Kuva 16. Ohjelmankiertoajan mittaus network 4 ja 5

Network 4 (Kuva 16) kohdassa tapahtuu vertailijan päivitys. Tämä kohta aktivoituu kun CMPR saa signaalin network2 kohdasta. Move lohkossa sisäisen ajan arvo kopioituu vertailijan arvoksi.

4.4 Kirjasto

Yritykselle luodusta logiikkaohjelmien kirjastosta ei saatu tehtyä niin kattavaa kokonaisuutta kuin alussa oli tarkoitus. Kirjastolle saatiin kuitenkin luotua hyvä perusta, jota yrityksen työntekijät voivat jatkossa päivittää. Kirjasto pitää sisällään noin 40 erilaista ohjelmalohkoa. Pääosa kirjaston materiaalista koostuu paperitehtaiden projekteista kerätyistä ohjelmista. Logiikkaohjelmien kirjasto käsittää muun muassa eri taajuusmuuttajien ohjauksia, absoluuttiantureiden luentaa, paikoitusohjauksia ja erilaisia muuntolohkoja. Alla olevassa kuvassa (Kuva 17) on työssä aikaansaatu logiikkaohjelmien kirjasto. Ohjelmien kommenttiriville on kirjoitettu tarkennus lohkojen toiminnasta.



Kuva 17. Logiikkaohjelmien kirjasto

5 POHDINTA

Yrityksen yhteistä ohjelmoitavien logiikoiden kirjastoa STEP 7 -ympäristöön oli suunniteltu perustettavan yrityksessä monien vuosien ajan, mutta idea kirjastosta oli jäänyt ajatuksen tasolle. Opinnäytetyölle oli siis olemassa suoranainen tarve yrityksen ohjelmointiyksikössä. Yhteisen kirjaston ansiosta työntekijöiden ajankäyttö tehostuu ja tämä tuo yritykselle kustannussäästöjä. Lisäksi yhtenäisen toimintatavan myötä projektit toteutuvat samassa linjassa.

Työn tavoitteena oli luoda mahdollisimman kattava logiikkaohjelmienkirjasto sisältäen selkeät käyttöohjeet. Materiaali kirjastoon oli tarkoitus kerätä yrityksen työntekijöiltä. Tavoite ei kuitenkaan toteutunut täysin, sillä työntekijöiden ja minun aikataulujen yhteensovittaminen ei onnistunut kiireisen syksyn vuoksi. Lisäksi käyttöohjeiden laatiminen ohjelmille itsenäisesti oli lähes mahdotonta tällä aikataululla. Saimme kuitenkin yhteistyössä yrityksen toimitusjohtajan kanssa luotua hyvän perustan kirjastolle, jota pystytään tulevaisuudessa kehittämään lisää.

Opinnäytetyön tekeminen osoittautuikin luultua haastavammaksi, mutta samalla se oli erittäin mielenkiintoista. Päädyin työskentelemään itsenäisesti lähes koko työn ajan, joten pääsin kunnolla tutustumaan yrityksen tekemiin projekteihin. Sain itsenäisesti tutkia projekteja ja etsiä niistä kirjastoon materiaalia. Suuriin projekteihin perehtyminen oli kiehtovaa ja hyvin opettavaista. Lisäksi useissa projektien lohkoissa oli käytetty STL- tai LAD-ohjelmointikieltä koulussa opetetun FBD-kielen sijasta. Näiden lohkojen tulkitseminen oli vaikeaa ja aikaa vievää. Simatic Manageria en ollut paljon käyttänyt ennen opinnäytetyön aloittamista, joten siihen perehtyminen ja sen käyttäminen veivät ison osan ajastani.

LÄHTEET

Berger, Hans 2012. Automating with STEP 7 in LAD and FBD 5th edition. Erlangen: Publicis Publishing

Berger Hans 2013. Automating with Simatic Controllers, Software, Programming, Data Communication, Operator Control and Process Minotoring 5th Edition. Erlangen: Publicis Publishing

PLC-Yhtiön www-sivut 2012. Hakupäivä 28.19.2013. <www.plc.fi>

Siemensin www-sivut 2013. Hakupäivä 23.10.2013
<http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automatiotekniikka/ohjelmoitavat_logiikat_simatic/s7_300.php>

Siemens Oy:n teollisuuden teknisentuen verkkosivut 2011, hakupäivä 29.10.2013
<http://support.automation.siemens.com/ww/llisapi.dll?func=cslib.csinfo&objId=23450941&load=treecontent&lang=en&siteid=cseus&aktprim=0&objaction=csview&extranet=standard&viewreg=WW>

Siemens Simatic 2010 Ladder Logic (LAD) for S7-300 and S7-400 Programming Reference Manual
<http://www.automation.siemens.com/DocOnWeb/pdf/SINUMERIK_SINAMICS_10_2012_E/S7_KOP.pdf?p=1>

Siemens Simatic 1996 S7 FUB/FBD S7-300/400 Yksiköiden ohjelmointi toimintakaaviomuodossa Käsikirja.
<http://www.siemens.fi/pool/products/industry/iadt_is/tuotteet/automaatiotekniikka/ohjelmoitavat_logiikat/s7_300/simatic-s7-300-400-ohjelmointikasikirja-fup-fbd_c79000-g7000-c508-02.pdf>

Siemens Simatic 2006 System software for S7-300/400 System and Standard Function Volume 1/2 Reference Manual.
<https://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SINAMICS_02_2012_E/S7_SFC.pdf?p=1>

LIITTEET

Liite 1 1 (2) Organisaatiolohkojen toiminnot

OB	Start Event	Default Priority Class	Explanation
OB1	End of startup or end of OB1	1	Free cycle
OB10	Time-of-day interrupt 0	2	No default time specified
OB11	Time-of-day interrupt 1	2	
OB12	Time-of-day interrupt 2	2	
OB13	Time-of-day interrupt 3	2	
OB14	Time-of-day interrupt 4	2	
OB15	Time-of-day interrupt 5	2	
OB16	Time-of-day interrupt 6	2	
OB17	Time-of-day interrupt 7	2	
OB20	Time-delay interrupt 0	3	No default time specified
OB21	Time-delay interrupt 1	4	
OB22	Time-delay interrupt 2	5	
OB23	Time-delay interrupt 3	6	
OB30	Cyclic interrupt 0 (default interval: 5 s)	7	Cyclic interrupts
OB31	Cyclic interrupt 1 (default interval: 2 s)	8	
OB32	Cyclic interrupt 2 (default interval: 1 s)	9	
OB33	Cyclic interrupt 3 (default interval: 500 ms)	10	
OB34	Cyclic interrupt 4 (default interval: 200 ms)	11	
OB35	Cyclic interrupt 5 (default interval: 100 ms)	12	
OB36	Cyclic interrupt 6 (default interval: 50 ms)	13	
OB37	Cyclic interrupt 7 (default interval: 20 ms)	14	
OB38	Cyclic interrupt 8 (default interval: 10 ms)	15	
OB40	Hardware interrupt 0	16	Hardware interrupts
OB41	Hardware interrupt 1	17	
OB42	Hardware interrupt 2	18	
OB43	Hardware interrupt 3	19	
OB44	Hardware interrupt 4	20	
OB45	Hardware interrupt 5	21	
OB46	Hardware interrupt 6	22	
OB47	Hardware interrupt 7	23	
OB55	Status interrupt	2	DPV1 interrupts
OB56	Update interrupt	2	
OB57	Manufacturer specific interrupt	2	
OB60	SFC35 "MP_ALM" call	25	Multicomputing interrupt

Liite 1 2 (2) Organisaatiolohkojen toiminnot

OB	Start Event	Default Priority Class	Explanation
OB 61	Synchronous Cycle Interrupt 1	25	Synchronous Cycle Interrupt
OB 62	Synchronous Cycle Interrupt 2	25	
OB 63	Synchronous Cycle Interrupt 3	25	
OB 64	Synchronous Cycle Interrupt 4	25	
OB 65	Technology synchronization interrupt	25	Technology synchronization interrupt
OB70	I/O redundancy error (only in H CPUs)	25	Redundancy error interrupts
OB72	CPU redundancy error (only in H CPUs)	28	
OB 73	Communication redundancy error OB (only in H CPUs)	25	
OB80	Time error	26, 28 1)	Asynchronous error interrupts
OB81	Power supply fault	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB82	Diagnostic interrupt	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB83	Insert/remove module interrupt	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB84	CPU hardware fault	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB85	Program error	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB86	Failure of an expansion rack, DP master system or station for distributed I/Os	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB87	Communication error	26, 28 1) with S7-300, 25, 28 1) with S7-400 and CPU 318	
OB 88	Processing interrupt	28	
OB90	Warm or cold restart or delete a block being executed in OB90 or load an OB90 on the CPU or terminate OB90	29 2)	Background cycle
OB100	Warm restart	27 1)	Startup
OB101	Hot restart	27 1)	
OB102	Cold restart	27 1)	
OB121	Programming error	Priority of the OB causing the error	Synchronous error interrupts
OB122	I/O access error	Priority of the OB causing the error	

Liite 2 1 (4) Kaikki LAD-käskyt

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
---| |---	---| |---	Bit logic Instruction	Normally Open Contact (Address)
---|/|---	---|/|---	Bit logic Instruction	Normally Closed Contact (Address)
---( )	---( )	Bit logic Instruction	Output Coil
---(#)---	---(#)---	Bit logic Instruction	Midline Output
==0 ---| |---	==0 ---| |---	Status bits	Result Bit Equal 0
>0 ---| |---	>0 ---| |---	Status bits	Result Bit Greater Than 0
>=0 ---| |---	>=0 ---| |---	Status bits	Result Bit Greater Equal 0
<=0 ---| |---	<=0 ---| |---	Status bits	Result Bit Less Equal 0
<0 ---| |---	<0 ---| |---	Status bits	Result Bit Less Than 0
<>0 ---| |---	<>0 ---| |---	Status bits	Result Bit Not Equal 0
ABS	ABS	Floating point Instruction	Establish the Absolute Value of a Floating-Point Number
ACOS	ACOS	Floating point Instruction	Establish the Arc Cosine Value
ADD_DI	ADD_DI	Integer Math Instruction	Add Double Integer
ADD_I	ADD_I	Integer Math Instruction	Add Integer
ADD_R	ADD_R	Floating point Instruction	Add Real
ASIN	ASIN	Floating point Instruction	Establish the Arc Sine Value
ATAN	ATAN	Floating point Instruction	Establish the Arc Tangent Value
BCD_DI	BCD_DI	Convert	BCD to Double Integer
BCD_I	BCD_I	Convert	BCD to Integer
BR ---| |---	BIE ---| |---	Status bits	Exception Bit Binary Result
----(CALL)	----(CALL)	Program control	Call FC SFC from Coil (without Parameters)
CALL_FB	CALL_FB	Program control	Call FB from Box
CALL_FC	CALL_FC	Program control	Call FC from Box
CALL_SFB	CALL_SFB	Program control	Call System FB from Box

Liite 2 2 (4) Kaikki LAD-käskyt

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
CALL_SFC	CALL_SFC	Program control	Call System FC from Box
----(CD)	----(ZR)	Counters	Down Counter Coil
CEIL	CEIL	Convert	Ceiling
CMP >=D	CMP >=D	Compare	Compare Double Integer (==, <>, >, <, >=, <=)
CMP >=I	CMP >=I	Compare	Compare Integer (==, <>, >, <, >=, <=)
CMP >=R	CMP >=R	Compare	Compare Real (==, <>, >, <, >=, <=)
COS	COS	Floating point Instruction	Establish the Cosine Value
----(CU)	---(ZV)	Counters	Up Counter Coil
DI_BCD	DI_BCD	Convert	Double Integer to BCD
DI_R	DI_R	Convert	Double Integer to Floating-Point
DIV_DI	DIV_DI	Integer Math Instruction	Divide Double Integer
DIV_I	DIV_I	Integer Math Instruction	Divide Integer
DIV_R	DIV_R	Floating point Instruction	Divide Real
EXP	EXP	Floating point Instruction	Establish the Exponential Value
FLOOR	FLOOR	Convert	Floor
I_BCD	I_BCD	Convert	Integer to BCD
I_DI	I_DI	Convert	Integer to Double Integer
INV_I	INV_I	Convert	Ones Complement Integer
INV_DI	INV_DI	Convert	Ones Complement Double Integer
---(JMP)	---(JMP)	Jumps	Unconditional Jump
---(JMP)	---(JMP)	Jumps	Conditional Jump
---(JMPN)	---(JMPN)	Jumps	Jump-If-Not
LABEL	LABEL	Jumps	Label
LN	LN	Floating point Instruction	Establish the Natural Logarithm
---(MCR>)	---(MCR>)	Program control	Master Control Relay Off
---(MCR<)	---(MCR<)	Program control	Master Control Relay On
---(MCRA)	---(MCRA)	Program control	Master Control Relay Activate
---(MCRD)	---(MCRD)	Program control	Master Control Relay Deactivate
MOD_DI	MOD_DI	Integer Math Instruction	Return Fraction Double Integer
MOVE	MOVE	Move	Assign a Value
MUL_DI	MUL_DI	Integer Math Instruction	Multiply Double Integer
MUL_I	MUL_I	Integer Math Instruction	Multiply Integer
MUL_R	MUL_R	Floating point Instruction	Multiply Real
---(N)---	---(N)---	Bit logic Instruction	Negative RLO Edge Detection

Liite 2 3 (4) Kaikki LAD-käskyt

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
NEG	NEG	Bit logic Instruction	Address Negative Edge Detection
NEG_DI	NEG_DI	Convert	Twos Complement Double Integer
NEG_I	NEG_I	Convert	Twos Complement Integer
NEG_R	NEG_R	Convert	Negate Floating-Point Number
---| NOT |---	---| NOT |---	Bit logic Instruction	Invert Power Flow
---(OPN)	---(OPN)	DB call	Open Data Block: DB or DI
OS ---| |---	OS ---| |---	Status bits	Exception Bit Overflow Stored
OV ---| |---	OV ---| |---	Status bits	Exception Bit Overflow
---(P)---	---(P)---	Bit logic Instruction	Positive RLO Edge Detection
POS	POS	Bit logic Instruction	Address Positive Edge Detection
---(R)	---(R)	Bit logic Instruction	Reset Coil
---(RET)	---(RET)	Program control	Return
ROL_DW	ROL_DW	Shift/Rotate	Rotate Left Double Word
ROR_DW	ROR_DW	Shift/Rotate	Rotate Right Double Word
ROUND	ROUND	Convert	Round to Double Integer
RS	RS	Bit logic Instruction	Reset-Set Flip Flop
---(S)	---(S)	Bit logic Instruction	Set Coil
---(SAVE)	---(SAVE)	Bit logic Instruction	Save RLO into BR Memory
---(SC)	---(SZ)	Counters	Set Counter Value
S_CD	Z_RUECK	Counters	Down Counter
S_CU	Z_VORW	Counters	Up Counter
S_CUD	ZAEHLER	Counters	Up-Down Counter
---(SD)	---(SE)	Timers	On-Delay Timer Coil
---(SE)	---(SV)	Timers	Extended Pulse Timer Coil
---(SF)	---(SA)	Timers	Off-Delay Timer Coil
SHL_DW	SHL_DW	Shift/Rotate	Shift Left Double Word
SHL_W	SHL_W	Shift/Rotate	Shift Left Word
SHR_DI	SHR_DI	Shift/Rotate	Shift Right Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Shift Right Double Word
SHR_I	SHR_I	Shift/Rotate	Shift Right Integer
SHR_W	SHR_W	Shift/Rotate	Shift Right Word
SIN	SIN	Floating point Instruction	Establish the Sine Value
S_ODT	S_EVERZ	Timers	On-Delay S5 Timer
S_ODTS	S_SEVERZ	Timers	Retentive On-Delay S5 Timer
S_OFFDT	S_AVERZ	Timers	Off-Delay S5 Timer
---(SP)	---(SI)	Timers	Pulse Timer Coil
S_PEXT	S_VIMP	Timers	Extended Pulse S5 Timer
S_PULSE	S_IMPULS	Timers	Pulse S5 Timer
SQR	SQR	Floating point Instruction	Establish the Square

Liite 2 4 (4) Kaikki LAD-käskyt

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
SQRT	SQRT	Floating point Instruction	Establish the Square Root
SR	SR	Bit logic Instruction	Set-Reset Flip Flop
---(SS)	---(SS)	Timers	Retentive On-Delay Timer Coil
SUB_DI	SUB_DI	Integer Math Instruction	Subtract Double Integer
SUB_I	SUB_I	Integer Math Instruction	Subtract Integer
SUB_R	SUB_R	Floating point Instruction	Subtract Real
TAN	TAN	Floating point Instruction	Establish the Tangent Value
TRUNC	TRUNC	Convert	Truncate Double Integer Part
UO ---| |---	UO ---| |---	Status bits	Exception Bit Unordered
WAND_DW	WAND_DW	Word logic Instruction	AND Double Word
WAND_W	WAND_W	Word logic Instruction	AND Word
WOR_DW	WOR_DW	Word logic Instruction	OR Double Word
WOR_W	WOR_W	Word logic Instruction	OR Word
WXOR_DW	WXOR_DW	Word logic Instruction	Exclusive OR Double Word
WXOR_W	WXOR_W	Word logic Instruction	Exclusive OR Word

Liite 3 1 (5) Kaikki STL-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
+	+	Integer math Instruction	Add Integer Constant (16, 32-Bit)
=	=	Bit logic Instruction	Assign
)	)	Bit logic Instruction	Nesting Closed
+AR1	+AR1	Accumulator	AR1 Add ACCU 1 to Address Register 1
+AR2	+AR2	Accumulator	AR2 Add ACCU 1 to Address Register 2
+D	+D	Integer math Instruction	Add ACCU 1 and ACCU 2 as Double Integer (32-Bit)
–D	–D	Integer math Instruction	Subtract ACCU 1 from ACCU 2 as Double Integer (32-Bit)
*D	*D	Integer math Instruction	Multiply ACCU 1 and ACCU 2 as Double Integer (32-Bit)
/D	/D	Integer math Instruction	Divide ACCU 2 by ACCU 1 as Double Integer (32-Bit)
? D	? D	Compare	Compare Double Integer (32-Bit) ==, <>, >, <, >=, <=
+I	+I	Integer math Instruction	Add ACCU 1 and ACCU 2 as Integer (16-Bit)
–I	–I	Integer math Instruction	Subtract ACCU 1 from ACCU 2 as Integer (16-Bit)
*I	*I	Integer math Instruction	Multiply ACCU 1 and ACCU 2 as Integer (16-Bit)
/I	/I	Integer math Instruction	Divide ACCU 2 by ACCU 1 as Integer (16-Bit)
? I	? I	Compare	Compare Integer (16-Bit) ==, <>, >, <, >=, <=
+R	+R	Floating point Instruction	Add ACCU 1 and ACCU 2 as a Floating-Point Number (32-Bit IEEE 754)
–R	–R	Floating point Instruction	Subtract ACCU 1 from ACCU 2 as a Floating-Point Number (32-Bit IEEE 754)
*R	*R	Floating point Instruction	Multiply ACCU 1 and ACCU 2 as Floating-Point Numbers (32-Bit IEEE 754)
/R	/R	Floating point Instruction	Divide ACCU 2 by ACCU 1 as a Floating-Point Number (32-Bit IEEE 754)
? R	? R	Compare	Compare Floating-Point Number (32-Bit) ==, <>, >, <, >=, <=
ABS	ABS	Floating point Instruction	Absolute Value of a Floating-Point Number (32-Bit IEEE 754)
ACOS	ACOS	Floating point Instruction	Generate the Arc Cosine of a Floating-Point Number (32-Bit)
ASIN	ASIN	Floating point Instruction	Generate the Arc Sine of a Floating-Point Number (32-Bit)
ATAN	ATAN	Floating point Instruction	Generate the Arc Tangent of a Floating-Point Number (32-Bit)
AUF	OPN	DB call	Open a Data Block
BE	BE	Program control	Block End
BEA	BEU	Program control	Block End Unconditional
BEB	BEC	Program control	Block End Conditional
BLD	BLD	Program control	Program Display Instruction (Null)
BTD	BTD	Convert	BCD to Integer (32-Bit)
BTI	BTI	Convert	BCD to Integer (16-Bit)

Liite 3 2 (5) Kaikki STL-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
CALL	CALL	Program control	Block Call
CALL	CALL	Program control	Call Multiple Instance
CALL	CALL	Program control	Call Block from a Library
CC	CC	Program control	Conditional Call
CLR	CLR	Bit logic Instruction	Clear RLO (=0)
COS	COS	Floating point Instruction	Generate the Cosine of Angles as Floating-Point Numbers (32-Bit)
DEC	DEC	Accumulator	Decrement ACCU 1-L-L
DTB	DTB	Convert	Double Integer (32-Bit) to BCD
DTR	DTR	Convert	Double Integer (32-Bit) to Floating-Point (32-Bit IEEE 754)
ENT	ENT	Accumulator	Enter ACCU Stack
EXP	EXP	Floating point Instruction	Generate the Exponential Value of a Floating-Point Number (32-Bit)
FN	FN	Bit logic Instruction	Edge Negative
FP	FP	Bit logic Instruction	Edge Positive
FR	FR	Counters	Enable Counter (Free) (free, FR C 0 to C 255)
FR	FR	Timers	Enable Timer (Free)
INC	INC	Accumulator	Increment ACCU 1-L-L
INVD	INVD	Convert	Ones Complement Double Integer (32-Bit)
INVI	INVI	Convert	Ones Complement Integer (16-Bit)
ITB	ITB	Convert	Integer (16-Bit) to BCD
ITD	ITD	Convert	Integer (16-Bit) to Double Integer (32-Bit)
L	L	Load/Transfer	Load
L DBLG	L DBLG	Load/Transfer	Load Length of Shared DB in ACCU 1
L DBNO	L DBNO	Load/Transfer	Load Number of Shared DB in ACCU 1
L DILG	L DILG	Load/Transfer	Load Length of Instance DB in ACCU 1
L DINO	L DINO	Load/Transfer	Load Number of Instance DB in ACCU 1
L STW	L STW	Load/Transfer	Load Status Word into ACCU 1
L	L	Load/Transfer	Load Current Timer Value into ACCU 1 as Integer (the current timer value can be a number from 0 to 255, for example, L T 32)
L	L	Load/Transfer	Load Current Counter Value into ACCU 1 (the current counter value can be a number from 0 to 255, for example, L C 15)
LAR1	LAR1	Load/Transfer	Load Address Register 1 from ACCU 1
LAR1	LAR1	Load/Transfer	Load Address Register 1 with Double Integer (32-Bit Pointer)
LAR1	LAR1	Load/Transfer	Load Address Register 1 from Address Register 2
LAR2	LAR2	Load/Transfer	Load Address Register 2 from ACCU 1
LAR2	LAR2	Load/Transfer	Load Address Register 2 with Double Integer (32-Bit Pointer)
LC	LC	Counters	Load Current Counter Value into ACCU 1 as BCD (the current timer value can be a number from 0 to 255, for example, LC C 15)
LC	LC	Timers	Load Current Timer Value into ACCU 1 as BCD (the current counter value can be a number from 0 to 255, for example, LC T 32)
LEAVE	LEAVE	Accumulator	Leave ACCU Stack

Liite 3 3 (5) Kaikki STL-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
LN	LN	Floating point Instruction	Generate the Natural Logarithm of a Floating-Point Number (32-Bit)
LOOP	LOOP	Jumps	Loop
MCR(	MCR(	Program control	Save RLO in MCR Stack, Begin MCR
)MCR	)MCR	Program control	End MCR
MCRA	MCRA	Program control	Activate MCR Area
MCRD	MCRD	Program control	Deactivate MCR Area
MOD	MOD	Integer math Instruction	Division Remainder Double Integer (32-Bit)
NEGD	NEGD	Convert	Twos Complement Double Integer (32-Bit)
NEGI	NEGI	Convert	Twos Complement Integer (16-Bit)
NEGR	NEGR	Convert	Negate Floating-Point Number (32-Bit, IEEE 754)
NOP 0	NOP 0	Accumulator	Null Instruction
NOP 1	NOP 1	Accumulator	Null Instruction
NOT	NOT	Bit logic Instruction	Negate RLO
O	O	Bit logic Instruction	Or
O(	O(	Bit logic Instruction	Or with Nesting Open
OD	OD	Word logic Instruction	OR Double Word (32-Bit)
ON	ON	Bit logic Instruction	Or Not
ON(	ON(	Bit logic Instruction	Or Not with Nesting Open
OW	OW	Word logic Instruction	OR Word (16-Bit)
POP	POP	Accumulator	CPU with Two ACCUs
POP	POP	Accumulator	CPU with Four ACCUs
PUSH	PUSH	Accumulator	CPU with Two ACCUs
PUSH	PUSH	Accumulator	CPU with Four ACCUs
R	R	Bit logic Instruction	Reset
R	R	Counters	Reset Counter (the current counter can be a number from 0 to 255, for example, R C 15)
R	R	Timers	Reset Timer (the current timer can be a number from 0 to 255, for example, R T 32)
RLD	RLD	Shift/Rotate	Rotate Left Double Word (32-Bit)
RLDA	RLDA	Shift/Rotate	Rotate ACCU 1 Left via CC 1 (32-Bit)
RND	RND	Convert	Round
RND+	RND+	Convert	Round to Upper Double Integer
RND–	RND–	Convert	Round to Lower Double Integer
RRD	RRD	Shift/Rotate	Rotate Right Double Word (32-Bit)
RRDA	RRDA	Shift/Rotate	Rotate ACCU 1 Right via CC 1 (32-Bit)
S	S	Bit logic Instruction	Set
S	S	Counters	Set Counter Preset Value (the current counter can be a number from 0 to 255, for example, S C 15)
SA	SF	Timers	Off-Delay Timer
SAVE	SAVE	Bit logic Instruction	Save RLO in BR Register
SE	SD	Timers	On-Delay Timer
SET	SET	Bit logic Instruction	Set
SI	SP	Timers	Pulse Timer

Liite 3 4 (5) Kaikki STL-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
SIN	SIN	Floating point Instruction	Generate the Sine of Angles as Floating-Point Numbers (32-Bit)
SLD	SLD	Shift/Rotate	Shift Left Double Word (32-Bit)
SLW	SLW	Shift/Rotate	Shift Left Word (16-Bit)
SPA	JU	Jumps	Jump Unconditional
SPB	JC	Jumps	Jump if RLO = 1
SPBB	JCB	Jumps	Jump if RLO = 1 with BR
SPBI	JBI	Jumps	Jump if BR = 1
SPBIN	JNBI	Jumps	Jump if BR = 0
SPBN	JCN	Jumps	Jump if RLO = 0
SPBNB	JNB	Jumps	Jump if RLO = 0 with BR
SPL	JL	Jumps	Jump to Labels
SPM	JM	Jumps	Jump if Minus
SPMZ	JMZ	Jumps	Jump if Minus or Zero
SPN	JN	Jumps	Jump if Not Zero
SPO	JO	Jumps	Jump if OV = 1
SPP	JP	Jumps	Jump if Plus
SPPZ	JPZ	Jumps	Jump if Plus or Zero
SPS	JOS	Jumps	Jump if OS = 1
SPU	JUO	Jumps	Jump if Unordered
SPZ	JZ	Jumps	Jump if Zero
SQR	SQR	Floating point Instruction	Generate the Square of a Floating-Point Number (32-Bit)
SQRT	SQRT	Floating point Instruction	Generate the Square Root of a Floating-Point Number (32-Bit)
SRD	SRD	Shift/Rotate	Shift Right Double Word (32-Bit)
SRW	SRW	Shift/Rotate	Shift Right Word (16-Bit)
SS	SS	Timers	Retentive On-Delay Timer
SSD	SSD	Shift/Rotate	Shift Sign Double Integer (32-Bit)
SSI	SSI	Shift/Rotate	Shift Sign Integer (16-Bit)
SV	SE	Timers	Extended Pulse Timer
T	T	Load/Transfer	Transfer
T STW	T STW	Load/Transfer	Transfer ACCU 1 into Status Word
TAD	CAD	Convert	Change Byte Sequence in ACCU 1 (32-Bit)
TAK	TAK	Accumulator	Toggle ACCU 1 with ACCU 2
TAN	TAN	Floating point Instruction	Generate the Tangent of Angles as Floating-Point Numbers (32-Bit)
TAR	CAR	Load/Transfer	Exchange Address Register 1 with Address Register 2
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to ACCU 1
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to Destination (32-Bit Pointer)
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to Address Register 2
TAR2	TAR2	Load/Transfer	Transfer Address Register 2 to ACCU 1
TAR2	TAR2	Load/Transfer	Transfer Address Register 2 to Destination (32-Bit Pointer)
TAW	CAW	Convert	Change Byte Sequence in ACCU 1-L (16-Bit)
TDB	CDB	Convert	Exchange Shared DB and Instance DB
TRUNC	TRUNC	Convert	Truncate
U	A	Bit logic Instruction	And
U(	A(	Bit logic Instruction	And with Nesting Open

Liite 3 5 (5) Kaikki STL-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
UC	UC	Program control	Unconditional Call
UD	AD	Word logic Instruction	AND Double Word (32-Bit)
UN	AN	Bit logic Instruction	And Not
UN(	AN(	Bit logic Instruction	And Not with Nesting Open
UW	AW	Word logic Instruction	AND Word (16-Bit)
X	X	Bit logic Instruction	Exclusive Or
X(	X(	Bit logic Instruction	Exclusive Or with Nesting Open
XN	XN	Bit logic Instruction	Exclusive Or Not
XN(	XN(	Bit logic Instruction	Exclusive Or Not with Nesting Open
XOD	XOD	Word logic Instruction	Exclusive OR Double Word (32-Bit)
XOW	XOW	Word logic Instruction	Exclusive OR Word (16-Bit)
ZR	CD	Counters	Counter Down
ZV	CU	Counters	Counter Up

Liite 4 1 (4) Kaikki FBD-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
&	&	Bit logic instruction	AND Logic Operation
>=1	>=1	Bit logic instruction	OR Logic Operation
=	=	Bit logic instruction	Assign
#	#	Bit logic instruction	Midline Output
--|	--|	Bit logic instruction	Insert Binary Input
--o|	--o|	Bit logic instruction	Negate Binary Input
==0	==0	Status bits	Result Bits
>0	>0	Status bits	Result Bits
>=0	>=0	Status bits	Result Bits
<0	<0	Status bits	Result Bits
<=0	<=0	Status bits	Result Bits
<>0	<>0	Status bits	Result Bits
ABS	ABS	Floating point instruction	Forming the Absolute Value of a Floating-Point Number
ACOS	ACOS	Floating point instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
ADD_DI	ADD_DI	Integer math instruction	Add Double Integer
ADD_I	ADD_I	Integer math instruction	Add Integer
ADD_R	ADD_R	Floating point instruction	Add Real
ASIN	ASIN	Floating point instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
ATAN	ATAN	Floating point instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
BCD_DI	BCD_DI	Convert	BCD to Double Integer
BCD_I	BCD_I	Convert	BCD to Integer
BIE	BR	Status bits	Exception Bit BR Memory
CALL	CALL	Program control	Calling an FC/SFC without Parameters
CALL_FB	CALL_FB	Program control	CALL_FB (Call FB as Box)
CALL_FC	CALL_FC	Program control	CALL_FC (Call FC as Box)
CALL_SFB	CALL_SFB	Program control	CALL_SFB (Call System FB as Box)
CALL_SFC	CALL_SFC	Program control	CALL_SFC (Call System FC as Box)
CEIL	CEIL	Convert	Ceiling
CMP ? D	CMP ? D	Compare	Compare Double Integer

Liite 4 2 (4) Kaikki FBD-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
CMP ? I	CMP ? I	Compare	Compare Integer
CMP ? R	CMP ? R	Compare	Compare Real
COS	COS	Floating point instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
DI_BCD	DI_BCD	Convert	Double Integer to BCD
DI_R	DI_R	Convert	Double Integer to Real
DIV_DI	DIV_DI	Integer math instruction	Divide Double Integer
DIV_I	DIV_I	Integer math instruction	Divide Integer
DIV_R	DIV_R	Floating point instruction	Divide Real
EXP	EXP	Floating point instruction	Forming the Exponential Value of a Floating-Point Number
FLOOR	FLOOR	Convert	Floor
I_BCD	I_BCD	Convert	Integer to BCD
I_DI	I_DI	Convert	Integer to Double Integer
INV_I	INV_I	Convert	Ones Complement Integer
INV_DI	INV_DI	Convert	Ones Complement Double Integer
JMP	JMP	Jumps	Unconditional Jump in a Block
JMP	JMP	Jumps	Conditional Jump in a Block
JMPN	JMPN	Jumps	Jump-If-Not
LABEL	LABEL	Jumps	Jump Label
LN	LN	Floating point instruction	Forming the Natural Logarithm of a Floating-Point Number
MCR>	MCR>	Program control	Master Control Relay On/Off
MCR<	MCR<	Program control	Master Control Relay On/Off
MCRA	MCRA	Program control	Master Control Relay Activate/Deactivate
MCRD	MCRD	Program control	Master Control Relay Activate/Deactivate
MOD_DI	MOD_DI	Integer math instruction	Return Fraction Double Integer
MOVE	MOVE	Move	Assign Value
MUL_DI	MUL_DI	Integer math instruction	Multiply Double Integer
MUL_I	MUL_I	Integer math instruction	Multiply Integer
MUL_R	MUL_R	Floating point instruction	Multiply Real
N	N	Bit logic instruction	Negative RLO Edge Detection
NEG	NEG	Bit logic instruction	Address Negative Edge Detection
NEG_DI	NEG_DI	Convert	Twos Complement Double Integer
NEG_I	NEG_I	Convert	Twos Complement Integer
NEG_R	NEG_R	Convert	Negate Real Number
OPN	OPN	DB call	Open Data Block
OS	OS	Status bits	Exception Bit Overflow Stored
OV	OV	Status bits	Exception Bit Overflow
P	P	Bit logic instruction	Positive RLO Edge Detection
POS	POS	Bit logic instruction	Address Positive Edge Detection
R	R	Bit logic instruction	Reset Output
RET	RET	Program control	Return
ROL_DW	ROL_DW	Shift/Rotate	Rotate Left Double Word

Liite 4 3 (4) Kaikki FBD-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
ROUND	ROUND	Convert	Round to Double Integer
ROR_DW	ROR_DW	Shift/Rotate	Rotate Right Double Word
RS	RS	Bit logic instruction	Reset_Set Flip Flop
S	S	Bit logic instruction	Set Output
SA	SF	Timers	Start Off-Delay Timer
SAVE	SAVE	Bit logic instruction	Save RLO to BR Memory
S_AVERZ	S_OFFDT	Timers	Assign Off-Delay Timer Parameters and Start
SE	SD	Timers	Assign On-Delay Timer Parameters and Start
S_EVERZ	S_ODT	Timers	Assign On-Delay Timer Parameters and Start
SHL_DW	SHL_DW	Shift/Rotate	Shift Left Double Word
SHL_W	SHL_W	Shift/Rotate	Shift Left Word
SHR_DI	SHR_DI	Shift/Rotate	Shift Right Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Shift Right Double Word
SHR_I	SHR_I	Shift/Rotate	Shift Right Integer
SHR_W	SHR_W	Shift/Rotate	Shift Right Word
SI	SP	Timers	Start Pulse Timer
S_IMPULS	S_PULSE	Timers	Assign Pulse Timer Parameters and Start
SIN	SIN	Floating point-instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
SQR	SQR	Floating point-instruction	Forming the Square (SQR) of a Floating-Point Number
SQRT	SQRT	Floating point-instruction	Forming the Square Root (SQRT) of a Floating-Point Number
SR	SR	Bit logic instruction	Set_Reset Flip Flop
SS	SS	Timers	Start Retentive On-Delay Timer
S_SEVERZ	S_ODTS	Timers	Assign Retentive On-Delay Timer Parameters and Start
SUB_DI	SUB_DI	Integer math-instruction	Subtract Double Integer
SUB_I	SUB_I	Integer math-instruction	Subtract Integer
SUB_R	SUB_R	Floating point-instruction	Subtract Real
SV	SE	Timers	Start Extended Pulse Timer
S_VIMP	S_PEXT	Timers	Assign Extended Pulse Timer Parameters and Start
SZ	SC	Counters	Set Counter Value
TAN	TAN	Floating point-instruction	Forming Trigonometric Functions of Angles as Floating-Point Numbers
TRUNC	TRUNC	Convert	Truncate Double Integer Part
UO	UO	Status bits	Exception Bit Unordered
WAND_DW	WAND_DW	Word logic instruction	AND Double Word (Word)
WAND_W	WAND_W	Word logic instruction	AND Word (Word)
WOR_DW	WOR_DW	Word logic instruction	OR Double Word (Word)
WOR_W	WOR_W	Word logic instruction	OR Word (Word)
WXOR_DW	WXOR_DW	Word logic instruction	Exclusive OR Double Word (Word)
WXOR_W	WXOR_W	Word logic instruction	Exclusive OR Word (Word)
XOR	XOR	Bit logic instruction	Exclusive OR Logic Operation

Liite 4 4 (4) Kaikki FBD-käskyt

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
ZAEHLER	S_CUD	Counters	Assign Parameters and Count Up/Down
ZR	CD	Counters	Down Counter
Z_RUECK	S_CD	Counters	Assign Parameters and Count Down
ZV	CU	Counters	Up Counter
Z_VORW	S_CU	Counters	Assign Parameters and Count Up