Sami Saari

PLCopen XML -esitystapa sovellusten siirrossa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

11.5.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Sami Saari PLCopen XML -esitystapa sovellusten siirrossa 36 sivua + 2 liitettä 11.5.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	
Ohjaaja(t)	Konecranes Oyj, Development Engineer Juha-Pekka Suomela Metropolia AMK, Yliopettaja Kari Björn

Insinöörityössä selvitettiin PLCopen XML -esitystavan soveltuvuutta ohjelmoitavien logiikkojen eri ohjelmointiympäristöjen välillä. Ohjelmoitavien logiikkojen sovellusohjelmien siirtäminen eri ohjelmointiympäristöjen välillä helpottaisi sovellusohjelmien ylläpitoa ja sovellusohjelmien määrää pystyttäisiin vähentämään.

PLCpen XML -esitystapaa tukevia ohjelmointiympäristöjä löytyi testeihin kolmelta eri valmistajalta. Saksalainen, laitemerkkiriippumaton 3S-Smart Software Solutions GmbH lupaa CoDeSys V3.5 -ohjelmointiympäristölleen PLCopen XML -skeeman version 2.0 tuen ja saksalainen Beckhoff Automation GmbH lupaa TwinCAT 3 -ohjelmointiympäristölleen myös PLCopen XML -skeeman version 2.0 tuen. Kolmantena testattavana oli myöskin saksalaisen Phoenix Contact Software GmbH yhtiön Multiprog 5.50 -ohjelmointiympäristö, joka tukee PLCopen XML -skeeman versiota 1.0. Työssä siirrettiin näiden ohjelmointiympäristöjen välillä standardin IEC 61131-3 mukaisia sovellusohjelmia, jotka kyettiin tallentamaan testissä mukana olleista ohjelmointiympäristöistä PLCopen XML -muotoisina tiedostoina ja lataamaan PLCopen XML -muotoisina näihin ohjelmointiympäristöihin.

CoDeSys- ja TwinCAT-ohjelmointiympäristöt toimivat testissä hyvin samankaltaisesti ja lopputuloksetkin näiden ohjelmointiympäristöjen osalta olivat lähes samanlaisia. Multiprog-ohjelmointiympäristö ei tallentanut muiden ohjelmointiympäristöjen kanssa yhteensopivaa PLCopen XML -tiedostoa. Multiprog-ohjelmointiympäristöön ei myöskään saanut ladattua muiden ohjelmointiympäristöjen vastaavia PLCopen XML -tiedostoja.

CoDeSys- ja TwinCAT-ohjelmointiympäristöjen samankaltaisuusden syyksi paljastuu se, että TwinCAT-ohjelmointiympäristö on rakennettu perustuen CoDeSys-ohjelmointiympäristöön. Multiprog-ohjelmointiympäristön toimimattomuuden syynä on todennäköisesti toiminnalliset virheet ohjelmointiympäristössä ja PLCopen XML -skeeman versioiden erot. On kuitenkin toivoa että PLCopen XML -tuki saadaan myös muihinkin kuin näihin testissä mukana olleisiin ohjelmointiympäristöihin, koska lokakuussa 2014 on aloitettu työ PLCopen XML -esitystavan saamiseksi mukaan standardiin IEC 61131.

Avainsanat	PLCopen XML, IEC 61131-3, PLC-ohjelmansiirto

Author(s) Title Number of Pages Date	Sami Saari PLC control project exchange in PLCopen XML format 36 pages + 2 appendices 11 May 2015
Degree	Bachelor of Engineering
Degree Programme	Automation Technology
Specialisation option	
Instructor(s)	Juha-Pekka Suomela, Development Engineer, Konecranes Oyj Kari Björn, Principal lecturer, Metropolia University of Applied Sciences

The purpose of this Bachelor's thesis was to examine the ability to use PLCopen XML format to transfer the programmable logic controller project between different program development environments. The capability to transfer PLC projects between program development environments would facilitate the administration and the reducing the number of programs.

It was possible to find three program development environments with PLCopen XML support. The first was CoDeSys V3.5 made by 3S-Smart Software Solutions GmbH and the second is TwinCAT 3 from Beckhoff Automation GmbH. These two program development environments are supporting PLCopen XML schema version 2.0. The third was Multiprog 5.50 from Phoenix Contact Software GmbH with the support of PLCopen XML schema version 1.0. The different kind of IEC 61131-3 compliant PLC projects were exported from development environments in PLCopen XML format. These PLCopen XML files were checked and imported to all development environments.

CoDeSys and TwinCAT development environments worked similarly and the test results were almost identical. CoDeSys and TwinCAT environments are not able to import PLCopen XML files from Multiprog environment. Multiprog is not either able import PLCopen XML files from other development environments.

The reason why CoDeSys and TwinCAT -environments are so identical is because TwinCAT is based to CoDeSys V3 environment. Multiprog programming environment is not PLCopen XML compliant with others because it supports schema version 1.0 and there is some errors in that programming environment. However, there is hope that the PLCopen XML support is expanding to also other programming environments, because on October 2014, work is started to get PLCopen XML as the part of standard IEC 61131.

Keywords	PLCopen XML, IEC 61131-3, PLC program

Sisällys

Lyhenteet

IEC	International Electrotechnical Commission. Sähkö- ja elektroniikkateknologioiden kansainvälinen standardisoimisjärjestö.
PLC	Programmable Logic Controller, ohjelmoitava logiikka
POU	Program Organization Unit
SFC	Sequential Function Chart
IL	Instruction List
ST	Structured Text
LD	Ladder Diagram, tikapuukaavio
FBD	Function Block Diagram
HTML	Hypertext Markup Language
XML	Extensible Markup Language
ISO	International Organization for Standardization

1 Johdanto

Tämän insinöörityön tarkoituksena on tutkia Konecranes Oyj:n käytössä olevien ohjelmoitavien logiikkojen ohjelmointiympäristöjen välistä tiedonsiirtoa PLCopen XML -muodossa. Tämän PLCopen XML -muotoisen tiedonsiirron avulla tulisi olla mahdollista siirtää ohjelmoitavien logiikkalaitteiden sovellusohjelmia eri laitevalmistajien ohjelmointiympäristöjen välillä. Tällöin pystyttäisiin vähentämään testattavan ja ylläpidettävän ohjelmakoodin määrää, minkä ansiosta pystyttäisiin keskittämään enemmän resursseja ohjelmakoodin laadun parantamiseen.

Konecranes on maailman johtava nostolaitevalmistaja. Sähkökäyttöisiä nostureita on valmistettu vuodesta 1933 asti. Yhtiö on laajentuessaan ostanut yrityksiä laajalti Euroopasta, Yhdysvalloista, Intiasta ja Saudia-Arabiasta. Yhteistyökuviot Kiinassa ja Japanissa ovat myöskin laajentaneet toimialuetta ja tukea edellyttävien laitemerkkien kirjoa.

Konecranes-konserni huoltaa kaikkien nosturivalmistajien nostureita ja tarjoaa ylläpito- ja modernisointiratkaisuja kaikentyyppisille teollisuusnostureille, satamalaitteille ja työstökoneille. Konecranes on myöskin toteuttanut itse toimittamiensa nosturien ohjauksia eri valmistajien ohjelmoitavien logiikkojen avulla. Tämän takia eri ohjelmoitavien logiikkojen valmistajia ja erilaisia ohjelmointiympäristöjä on pystyttävä tukemaan. Ohjelmointityössä käytettyjä ohjelmointikieliäkin on useita. Omienkaan tuotteiden osalta Konecranesilla ei ole ollut yhtenäistä linjaa vaan erilaisien tottumuksien mukaan on valittu kulloinkin käytetty ohjelmointikieli. Myös loppukäyttäjät ovat esittäneet vaatimuksiaan nosturissa käytetystä PLC-valmistajan merkistä, jolloin eteen saattaa tulla sovellusohjelman siirtäminen ohjelmointiympäristöstä toiseen.

Jopa suurimpienkin PLC-valmistajien ohjelmointiympäristöjen välillä on yhteensopivuudessa ongelmia saman ohjelmointikielen osalta, vaikka nämä ovat esittäneet standardin IEC61131-3 mukaisen vaatimustenmukaisuuslausunnon ja PLC-valmistajat kertovat tukevansa tätä standardissa määriteltyä ohjelmointikieltä. Samalla ohjelmointikielellä tehtyjen ohjelmien siirtäminen eri ohjelmointiympäristöjän välillä ei siis ole ollut ongelmatonta. Sovellusohjelman siirto ohjelmointiympäristöstä toiseen on ollut työlästä ja osittain jopa mahdotonta ilman sovellusohjelman uudelleen syöttämistä. Sovellusohjelmien siirtoon eri ohjelmointiympäristöjen välillä on olemassa myös erillisiä

sovelluksia. Näiden sovelluksien ja muiden sovellusohjelman siirtotapojen tutkiminen olisi jo aivan oma työnsä ja rajattiin pois tästä työstä, jossa haluttiin keskittyä PLCopen XML-ratkaisuun.

PLCopen on valmistajariiippumaton ja tuoteriippumaton kansainvälinen järjestö. Tähän kuuluu monia PLC-valmistajia, ohjelmistotaloja ja itsenäisiä järjestöjä. Järjestön tarkoituksena on luoda IEC61131-3 standardin mukaisia määrittelyjä ja toteutuksia, jotta voidaan vähentää suunnittelukustannuksia. PLCopen XML on työkalu standardisoitujen ohjelmien, kirjastojen ja projektien välittämiseksi ohjelmointi ympäristöjen välillä.

Työssä selvitetään PLCopen XML -muotoista sovellusohjelman siirtoa tukevat ohjelmointiympäristöt. Näiden välillä testataan sovellusohjelman siirto PLCopen XML-muodossa niin, että pystytään määrittelemään, miten näiden välillä tämä onnistuu ja millaisia asioita tulisi ottaa huomioon, mikäli ohjelmia ryhdytään siirretään ohjelmointiympäristöjen välillä.

Testausta varten Konecranes toimittaa ohjelmoitavalle logiikalle sovellusohjelman, joka antaa kuvan heitä erityisesti kiinnostavista sovellusohjelman toiminnallisuuksista. Testaus suoritetaan tallentamalla testaukseen käytetyt ohjelmoitavien logiikkojen sovellusohjelmat eri ohjelmointiympäristöistä PLCopen XML -muodossa. Näiden PLCopen XML -tiedostojen sisällön tarkastamiseen ei valitettavasti ole saataville tähän tarkoitukseen vartavasten olevaa sovellusta, joten tarkastuksessa tyydytään silmämääräiseen tarkastukseen siltä osin että kaikki tarvittavat PLCopen XML-tiedoston rakenteelliset osat ovat mukana. PLCopen XML -tiedostot ladataan testissä mukana oleviin ohjelmointiympäristöihin. Tässä tarkkaillaan, että näiden mukana siirtyy kaikki tarvittavat ohjelmoitavan logiikan sovellusohjelman osat ja näiden kääntäminen onnistuu kuten lähtökohtana olleessa ohjelmointiympäristössä.

2 Standardi IEC 61131

Standardissa IEC 61131 esitetään ohjelmoitavien logiikkojen ohjelmointikielien vaatimukset. Kokonaisuutena standardi IEC 61131 yhdistää ohjelmoitaville logiikoille esitetyt vaatimukset, joihin sisältyvät sekä laite- ja ohjelmointijärjestelmät. Standardi

IEC 61131 yhdistää ja jatkaa monia muita kansainvälisiä standardeja, jotka käsittelevät ohjelmoitavia logiikkoja.

Standardi IEC 61131 muodostuu tällä hetkellä kahdeksasta eri osiosta. Ensimmäinen osio eli standardi IEC 61131-1 on päivitetty viimeksi vuonna 2003 ja siinä on esitetty määritelmät ja yksilöity perusominaisuudet ohjelmoitaville logiikoille ja näiden oheislaitteille.

Standardi IEC 61131-2 tarkentaa laitevaatimukset ja siihen liittyvät testit ohjelmoitaville logiikoille ja näiden oheislaitteille. Tämä standardin osio on jo vuodelta 2007. Kolmas osio eli standardi IEC 61131-3 määrittelee vähimmäisvaatimukset ohjelmoitavien logiikkojen ohjelmointikielille. Se sisältää perusohjelmointielementit, kieliopit ja säännöt sen määrittelemille ohjelmointikielille. Tämä standardi on päivitetty vuonna 2013.

Standardi IEC 61131-4 on tekninen raportti sisältäen yleiskatsauksen ja standardin sovelluksille ohjesäännöt ohjelmoitavien logiikkojen käyttäjille. Tämä osio on viimeksi päivitetty vuonna 2004. Tiedonsiirto ohjelmoitavien logiikkojen ja muiden elektronisien laitteiden välillä käyttäen Manufacturing Message Specification (MMS) -standardia on määritelty standardin viidennessä osiossa IEC 61131-5. Manufacturing Message Specification on määritelty alunperin standardissa ISO/IEC 9506.

Standardissa IEC 61131-6 on määritelty vaatimukset turva-automaatioon liittyville ohjelmoitaville logiikoille ja näiden lisälaitteille. Sumean logiikan ohjelmointielementit on määritelty standardissa IEC 61131-7. Standardi IEC 61131-8 on tekninen raportti, joka sisältää ohjeet standardin IEC 61131-3 ohjelmointikieliä käyttäville ohjelmoijille. (1, s.12–16.)

3 Standardi IEC 61131-3

IEC 61131-3 on standardi ohjelmoitavien logiikkojen ohjelmointikielille. Tämän standardin ohjelmointikieliä koskevan kolmannen osan julkaisi kansainvälinen sähkö- ja elektroniikkateknologioiden standardisoimisjärjestö IEC (International Electrotechnical Commission) vuonna 1992. Nykyään käytössä oleva kolmas versio on julkaistu vuonna 2013. Standardin mukaan kaikkien PLC-valmistajien on esitettävä

vaatimustenmukaisuuslausunto, jossa määritellään mitä standardin ominaisuuksia ja standardin laajennuksia heidän tuotteensa tukee. (2)

3.1 Standardin sisältö

Standardi voidaan jakaa kahteen pääosioon, yleiset elementit (Common Elements) ja ohjelmointikielet. Edellä mainittujen kahden osion lisäksi standardissa on liitteiden muodossa kuvattu ohjelmointikielien rajoitukset, avainsanat, virhemahdollisuudet ja merkkien heksadesimaaliset koodaukset. (2)

3.2 Yleiset elementit

Standardin käsittelee myös ohjelmointikielien yleisiä elementtejä (Common Elements), joiden avulla sovellusohjelmat muodostetaan. Näitä ovat muun muassa tietoelementtien määrittelyt, muuttujat, laitekokoonpanot, resurssit ja toimintayksiköt. (2)

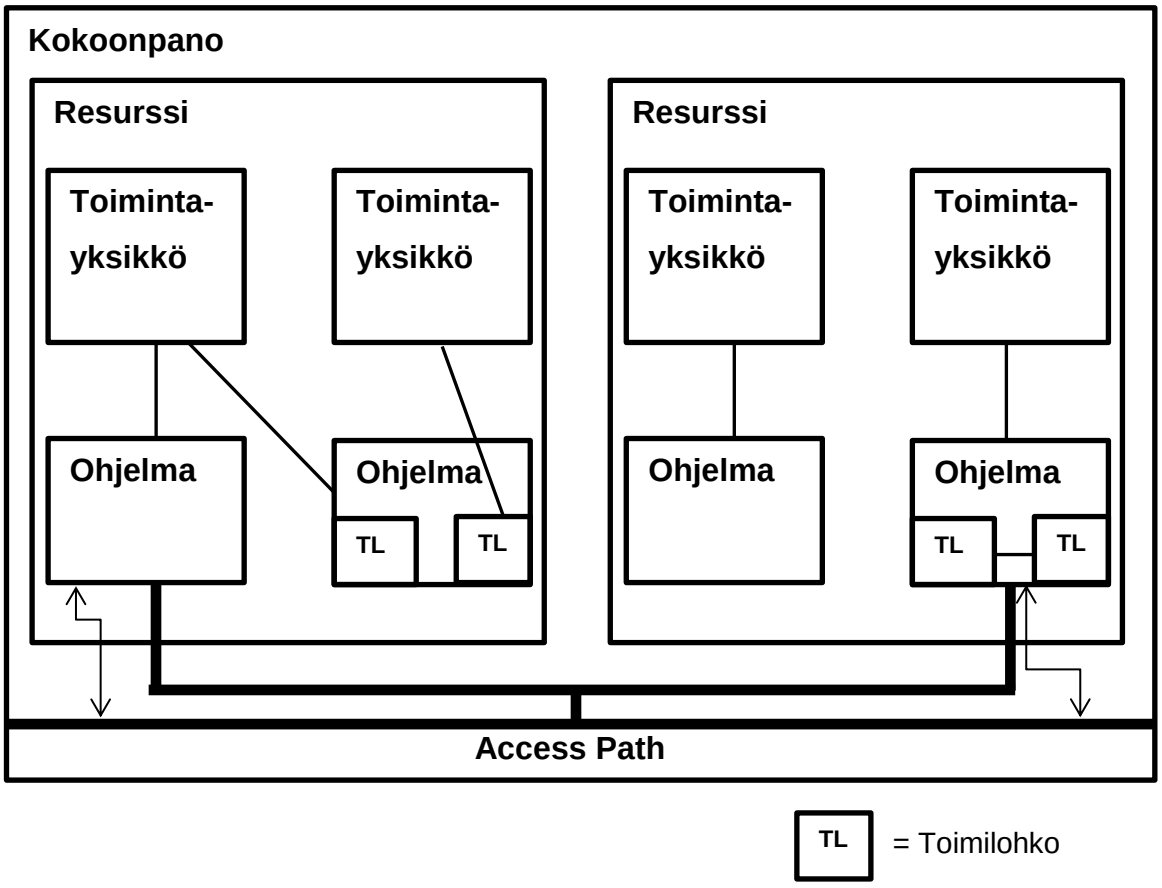
Eri ohjelmointikielille yhteisiä tietotyyppejä (data typing) eli muuttujien määrittelyitä ovat muun muossa totuusarvo (boolean), kokonaisluvut (integer), reaaliluvut (real), sana (word), päiväys (date) ja merkkijono (string). Näiden esitystavat ovat määritelty standardissa tarkkaan, jolloin näiden määrittelyt auttavat estämään virheitä muuttujien käsittelyssä. Näiden määrittely on pakollista kaikille sovellusohjelmassa käytetyille muuttujille. On mahdollista luoda myös käyttäjän omia tietotyyppejä perustuen standardin tietotyyppeihin.

Näiden tietotyyppien käsittelyssä on eri valmistajien välillä eroja. Esimerkiksi Siemens käsittelee totuusarvo- eli Boolen-tietotyyppiä yksibittisenä ja toiset yksitavuisena tietotyyppinä. Näiden jälkimmäisenä mainittujen yksibittisenä tietotyyppinä on bitti (bit). (1, s.74–86.)

Muuttujat ovat tietyn laitekokoonpanon osoitteita (esimerkiksi input ja output). Tällä tavoin korkeamman laitetason riippumattomuus tukien ohjelmiston uudelleen käytettävyyttä on muodostettu. Muuttujat tunnetaan yleensä paikallisesti esimerkiksi

kyseisen organisaation yksikössä. Muuttuja voidaan myös määritellä laajemmin tunnetuksi VAR_GLOBAL -muuttujaksi.

Standardissa määritellyn ohjelmistomallin osa-alueita ovat kuvassa 1 esitetyt kokoonpanot, resurssit ja toiminnot. Kokoonpano eli konfiguraatio (Configuration) on tietty ohjausjärjestelmä sisältäen yhden tai useamman laitteistoresurssin (Resource). Resurssina voidaan pitää sovellusohjelman (Program) suoritusyksikköä eli ohjelmoitavaa logiikkaa tai ohjelman suorittamisen ohjaava tietokone.



Kuva 1. Kokoonpanon rakenne

Toimintayksiköt (tasks) ohjaavat ohjelmien ja toimilohkojen suoritusta. Nämä sovellusohjelmat koostuvat standardin IEC 61131-3 mukaisien ohjelmointikielien avulla tehdyistä rakenneyksiköistä, joita nimitetään funktioiksi (function) ja toimilohkoiksi (function block). (2)

Ohjelmien ja konfiguraatioiden välisenä tietojen siirto väylänä toimii Access path -yhteydet. Näiden toiminta on esitetty kuvassa 1. Access path -nimitykselle ei ole

muodostunut tässä yhteydessä sopivaa suomenkielistä nimitystä, mutta tätä voisi nimittää yhteysväyläksi tai -poluksi. Toimilohkoista ei ole sallittua käyttää suoraan Access Path -väylän muuttujatyyppejä (VAR_ACCESS ja VAR_GLOBAL). Toimilohkossa näitä pitää käyttää ulkoisen muuttujamäärittelyn kautta (VAR_EXTERNAL). Ohjelmoitavien logiikkojen välinen tiedonsiirto määritellään standardin IEC 61131 osassa 5. (1, s.46.)

3.3 Rakenneyksiköt

Standardissa IEC 61131-3 rakenneyksiköitä eli ohjelmaelementtejä (POU-yksikkö, Program Organization Unit) ovat ohjelmat, toimilohkot ja funktiot. Funktiota ovat standardissa IEC 61131-3 on määritellyt standardifunktiot, kuten AND, OR ja SQRT. Järjestelmään voidaan myös määritellä omia funktioita. Toimilohkojen tarkoituksena on suorittaa jokin niille määrätty toiminto, esimerkiksi PID-säätö. Toimilohkolla on tietty hyvin tarkkaan määritelty käyttöliittymä, kuten em. funktioillakin.

Funktioiden ja toimilohkojen avulla rakennetaan sovellusohjelmat. Nämä ohjelmat ovat ohjelmoitavalle logiikalle suoritusohjeita. Näiden muodostamiseen käytetään standardin IEC 61131-3 mukaisia ohjelmointikieliä. (2)

3.4 Ohjelmointikielet

Standardin IEC 61131-3 ensimmäinen puolisko oli edellä mainitut yleiset elementit ja toinen puolisko on ohjelmointikielet. Stardardissa on määritelty neljä ohjelmointikieltä, jotka jakautuvat tekstimuotoisiin ja graafisiin ohjelmointikieliin. Seuraavaksi näistä ohjelmointikielistä esitetyt esimerkit kuvaavat toiminnaltaan toisiaan vastaavia ohjelmia.

3.4.1 SFC-ohjelmointikieli

Standardissa on määritelty myös näiden neljän ohjelmointikielen lisäksi ylemmän tason SFC-ohjelmointikieli eli Sequential Function Chart. Tämä kuvaa graafisesti ohjelman jaksollisen käyttäytymisen. SFC-ohjelmointikieli on kehitetty Petri net -nimisestä matemaattisesta mallinnuskielestä. Petri net -kieltä käytetään hajautettujen järjestelmien esittämiseen.

SFC muodostuu ohjelman toimintojen etenemisaskeleista (step), jotka ovat toisiinsa linkitettyjä toimintalohkoja (Action Blocks) ja siirtymistä (transitions). Jokainen toimintalohko voidaan muodostaa millä tahansa IEC-kielellä. SFC:n jaksot voivat olla jopa vaihtoehtoisia tai rinnakkaisia toisilleen. SFC tarjoaa myös kommunikaatiotyökalun yhdistäen eri osastoja tai organisaatioita.

SFC-kielen avulla voidaan monimutkainen sovellusohjelma jakaa pienempiin, helpommin käsiteltäviin yksiköihin ja kuvata näiden yksiköiden välistä ohjelmavuota. SFC-sovellusohjelma koostuu askelmista ja siirtymistä sekä linkeistä näiden välillä. SFC-sovellusohjelmalle on olemassa sekä tekstimuotoinen ja graafinen esitystapa. SFC-kielen askelmat kirjoitetaan standardissa IEC 61131-3 esitetyillä neljällä alemman tason ohjelmointikielellä. (1, s.169–202.)

3.4.2 IL-ohjelmointikieli

Tekstimuotoinen ohjelmointikieli IL eli Instruction List (suomeksi käskylista) perustuu assembly-ohjelmointikieleen. IL on riviperustainen kieli, jossa yhdellä rivillä kuvataan ohjelmoitavalle logiikalle yksi suoritettava komento. IL-ohjelmointikielessä on kuitenkin ulkoasun selkeyttämiseksi sallittu tyhjät rivit ja otsikointi. Otsikointia käytetään myös ohjelman suorittamisen siirtymiin. Seuraavassa esimerkissä näkyvässä ensimmäisessä sarakkeessa sijaitsevat komennot, joita ohjelma suorittaa toisella rivillä näkyville muuttujille.

```
LD          T_CMD_MAX
ST          CMD_TMR.PT
LD          AUTO_CMD
AND         AUTO_MODE
OR(         MAN_CMD
ANDN        AUTO_MODE
ANDN        MAN_CMD_CHK
)
ST          CMD
IN          CMD_TMR
LD          CMD_TMR.Q
ANDN        FDBK
ST          ALRM_FF.S1
LD          ACK
R           ALRM_FF
LD          ALRM_FF.Q1
ST          ALRM
```

Kuva 2. Esimerkki IL-ohjelmointikielen ohjelmoinnista

IL-ohjelmointikieli alemman tason ohjelmointikieli, joka on lähimpänä ohjelmaa suorittavan prosessorin ymmärtämää niin sanottua ”konekieltä”. IL-ohjelmointikieli toimii välittäjäkielenä muille tekstiperusteisille ja graafisille ohjelmointikielille, joiden ohjelmointi käännetään IL-kielelle ohjelmointiympäristön käännöstoiminnolla. (1, s. 100–113.)

3.4.3 ST-ohjelmointikieli

Toinen tekstimuotoinen ohjelmointikieli on ST eli Structured Text (suomeksi rakenteellinen teksti). ST-ohjelmointikieli perustuu Pascal-ohjelmointikieleen. ST-ohjelmakoodi muodostuu lauseista, joiden erottimena toimii kuvassa 3 näkyvä puolipilkku. Ohjelmakoodiin lisättävien kommenttien ja tekstin asettelun käyttöä suositellaan helpottamaan ohjelmakoodin lukemisen helpottamiseksi. (1, s. 116–130.)

```
CMD := AUTO_CMD & AUTO_MODE
          OR MAN_CMD & NOT MAN_CMD_CHK & NOT AUTO_MODE ;
CMD_TMR (IN := CMD, PT := T_CMD_MAX);
ALRM_FF (S1 := CMD_TMR.Q & NOT FDBK, R := ACK);
ALRM := ALRM_FF.Q1;
```

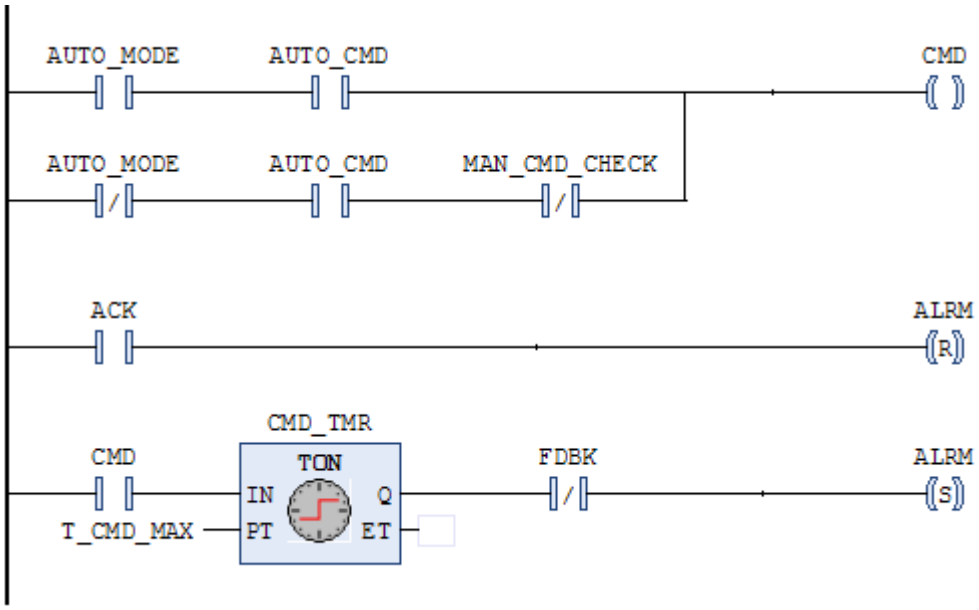
Kuva 3. Esimerkki ST-ohjelmointikielen ohjelmoinnista

ST-ohjelmointikieli on korkeamman tason ohjelmointikieli, jossa on mahdollista käyttää lohko- ja silmukkarakenteita, ehdollisia suorituksia ja funktioita. Näiden käyttö on ST-kielessä hyvin tehokasta ja helposti hallittavaa. ST-ohjelmakoodin lauseessa voidaan tavanomaisesti laskea ja osoittaa arvoja, mutta myös siis hallita ohjelman suorittamista käyttämällä (kutsumalla) ohjelman POU-rakenteita eli ohjelmointikielen valmiita funktioita tai itse tehtyjä funktioita. Kuvan 3 esimerkissä on määritelty CMD_TMR-funktioksi standardin IEC 61131-3 funktiota TimeOnTimer (TON). Tämä funktio viivästää ulostulon aktivoitumisen halutun aikajakson verran sisäänmenon aktivoitumisesta.

3.4.4 Ladder-ohjelmointikieli

Graafinen ohjelmointikieli LD eli Ladder Diagram (suomeksi tikapuukaavio) on amerikassa kehitetty tapa esittää ohjelman toiminta relelogiikan tavoin. Tämän takia

sen mieltäminen on helpompaa sähkötekniikan ja relelogiikkaa tuntevalle henkilölle. Esimerkkikuvassa 4 vasemmassa reunassa näkyvä pystyviiva kuvaa positiivista virtakiskoa. Vanhemmissa LD-kuvauksissa oikeaan reunaan on myös piirretty vaakakytkennät yhdistävä pystykisko, joka on kuvannut kytkennän ulostuloa. LD-ohjelman eteneminen tapahtuu kuvassa ylhäältä alaspäin.



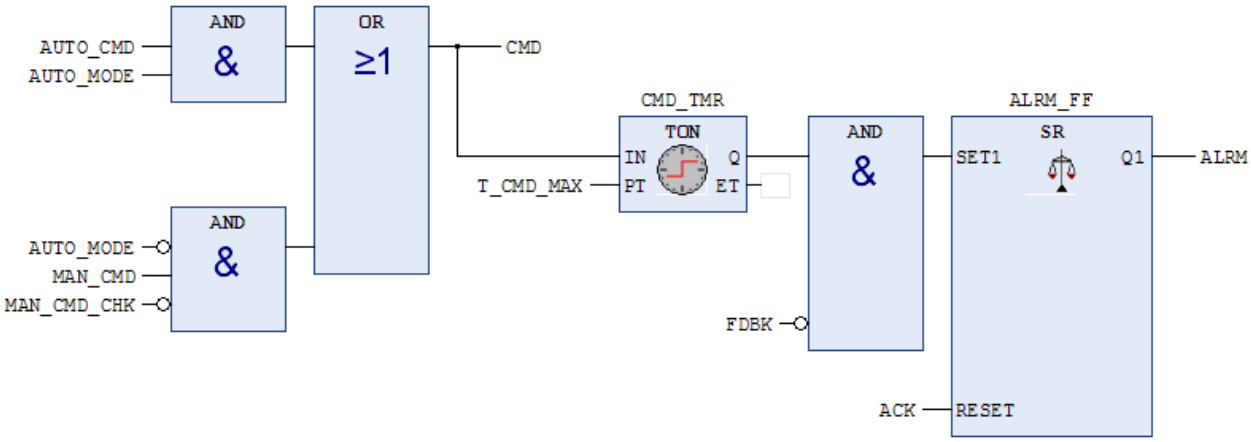
Kuva 4. Esimerkki LD-ohjelmointikielestä.

Ohjelmakoodin suoritettavat toiminteet ovat kuvattu vaakaviivoilla kytkiminä ja kytkentöinä. Suoritettavan toiminteen tallennuspaikka on kytkennän oikeassa reunassa eli esimerkki kuvan 4 vaakaviivan lopussa. Kuvan 4 esimerkistä on havaittavissa helpommin kuin aikaisemmasta ST-ohjelmakoodista TON-ajastinfunktion sisäänmeno (heräte) ja ulostulot. LD-ohjelmointi soveltuu helpoiten totuusarvo (boolean) -tyyppisien muuttujien käsittelyyn näiden muistuttaessa myös sähköteknistä päälle tai pois päältä kytkintä. LD-ohjelmointikielen yhteensopivuus eri valmistajien välillä on ollut hyvin heikkoa, mutta onneksi standardi IEC 61131-3 on auttanut tässä. (1, s. 147–158.)

3.5 FBD-ohjelmointikieli

Toinen standardissa esitetty graafinen ohjelmointikieli on FBD-ohjelmointikieli eli Function Block Diagram (suomeksi toimilohkokaavio). FBD-esitystapa on alunperin lähtöisin signaalinkäsittelytekniikasta, jossa on tärkeää erilaisten tarkkojen numeeristen arvojen muuttumiseen vaikuttavien asioiden kuvaaminen. Tämä muistuttaa hyvin paljon

prosessi- ja sähkötekniikan kaavioita. FBD-ohjelmointikieli soveltuukin hyvin teollisuuden prosessien ohjaukseen, mutta myös esimerkkikuvan 5 mukaiseen totuusarvo (boolean) -tyyppisien muuttujien käsittelyyn. (1, s. 134–143.)



Kuva 5. Esimerkki FBD-ohjelmointikielestä.

FBD-kielessä voidaan helposti ja havainnollisesti kuvata funktio sisään tulevien ja ulos tulevien muuttujien välillä. Kuvan lukeminen on helppoa ja loogista, vasemmalta oikealle. Kuten LD-ohjelmointikielisessä esimerkkikuvassa 4 ja tässä FBD-esimerkissä kuvassa 5 on aikaisemmin mainittu TON-funktio löydettävissä helposti ja nähtävissä tämän vaikuttavat muuttujat ja tämän ulostulon vaikutukset ohjelmassa eteenpäin.

4 PLCopen

PLCopen on valmistajariiippumaton ja tuoteriippumaton kansainvälinen järjestö. Tähän kuuluu monia PLC-valmistajia, ohjelmistotaloja ja itsenäisiä järjestöjä. PLCopen on perustettu vuonna 1992 heti standardin IEC 61131-3 julkaisemisen jälkeen. Järjestön tarkoituksena on luoda standardin IEC 61131-3 mukaisia määrittelyitä ja toteutuksia, jotta voidaan vähentää suunnittelukustannuksia. (1, s.16.)

PLCopen-järjestön jäsenet ovat organisoituneet kuuteen tekniseen komiteaan muodostaakseen oman erikoistumisalansa määrittelyt yhteistyössä loppukäyttäjien kanssa. Työryhmä TC1 käsittelee standardeja. Tämä työryhmä kerää jäseniltään ehdotukset IEC 65B WG7 -työryhmälle. Se muodostaa myös yhteisen kannanoton ehdotuksista ja jakaa aiheeseen liittyvää informaatiota. TC2-työryhmä käsittelee ohjelmoitavien logiikkojen sovellusohjelmien käyttämiä toimintoja. Se määrittelee

yhteisiä funktiokirjastoja, joiden avulla näitä kirjastoja ja näiden avulla tehtyjä ohjelmia voidaan käyttää erilaisissa ratkaisuissa ja jopa PLC-valmistajien laitteissa.

TC3 -työryhmä määrittelee standardin IEC 61131-3 mukaisten ohjelmointiympäristöjen sertifikointijärjestelmän. TC4-työryhmä työskentelee tiedonsiirron ja ohjelmointikielien välisen yhteistyön ja kommunikaation parantamiseksi. TC5-työryhmä valmistelee turva-automaatiosovelluksien suosituksia standardiin IEC 61131-3. TC6 -työryhmä työstää tähän insinöörityöhön liittyvän XML-skeeman määrittelyjä standardin IEC 61131-3 ohjelmointikielille. (2)

4.1 XML-esitysmuoto

XML on lyhenne sanoista eXtended Markup Language. XML-kieli on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin. XML:n kehittäjä on World Wide Web Consortium. Se on tekstimuotoista ja muistuttaa HTML-kieltä, jolla WWW-sivut kirjoitetaan, ja ne kummatkin ovat SGML-kielen yksinkertaistettuja osajoukkoja. XML-kieli ei kuitenkaan ole tarkoitettu sivunkuvauskieleksi kuten HTML, vaan sillä kuvataan tiedon rakenne ilman ennalta määrättyjä koodeja. (3)

Tietoverkoissa normaalisti välitettävät XML-tiedostot ovat lyhyitä ja informaatio on hyvin tiivistä. Tällöin saatetaan esimerkiksi lähettää jostakin tapahtumasta lyhyt ilmoitus toiselle osapuolelle jotta tämä osaa aloittaa tietyt toimenpiteet tämän tapahtuman johdosta. Tällaisia voi olla raja-arvon ylittyminen tieliikenteen nopeuden seurannassa tai arvontapainikkeen painaminen kuluttajia palvelevassa arpajaiskoneessa. Näissä tapauksissa XML-viestin vastaanottaja suorittaa sille määrätyt toimenpiteet XML-viestissä saatujen tietojen perusteella. Esimerkkitapauksissa tämä siis voi olla ajoneuvon ylinopeudesta seuraavat sanktiotoimet tai virallisen arpajaispalvelimen suorittamat arvontatoimet.

4.2 PLCopen XML

Standardin IEC 61131-3 julkaisun jälkeen on aloitettu myös kehittämään työkalua standardisoitujen ohjelmien, kirjastojen ja projektien välittämiseksi kehitysympäristöjen välillä. Tämän komitean nimi on TC6, joka työskentelee muodostaakseen avoimen

XML-mallin määrittelyn eli skeeman kaikille standardin IEC 61131-3 ohjelmointikielille. PLCopen XML -skeema ei siis ole standardi, vaan se on eri tahojen yhdessä sopima toimintatapa standardin IEC 61131-3 mukaisien sovellusohjelmien siirrolle eri ohjelmointiympäristöjen välillä.

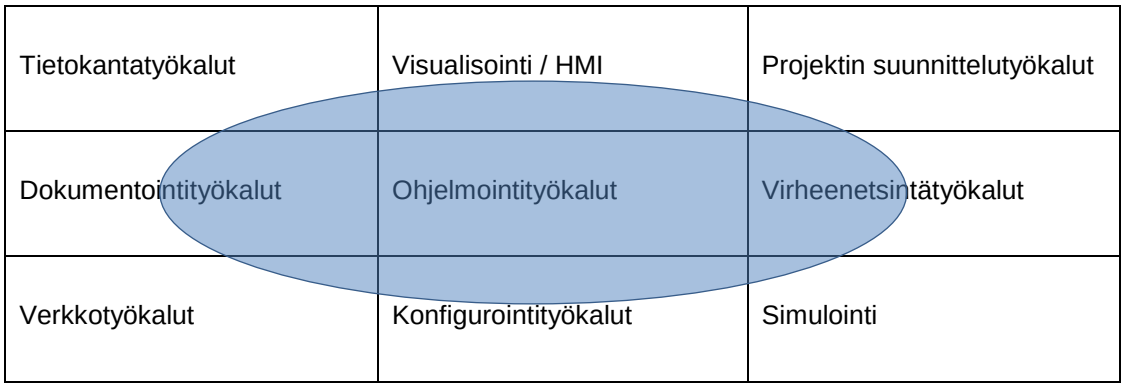
Ensimmäinen PLCopen XLM-skeeman versio 1.01 julkaistiin virallisesti vuonna 2005. Seuraavan suurempia muutoksia tuonut versio 2.0 julkaistiin muutama vuosi myöhemmin vuonna 2008. Tämän version muutokset on lueteltu seuraavassa taulukossa.

PLCopen XML -skeema	Julkaisu-ajankohta	Muutokset edellisestä versiosta
0.99	16.4.2004	Epävirallinen vedosversio.
1.01	10.6.2005	Ei tiedossa.
2.0	3.12.2008	1. POU-yksiköiden kuvauksien määrä. 2. SFC -kaavion määritys ”ConnectionPointIn” lisätty. 3. Toimilohkon määrittelyihin lisätty korkeus ja leveys. 4. Toimilohkon määrittelyt ”localID”, ”executionOrderID” ja ”connectionPointOut” lisätty. 5. Lisätty tietotyyppeihin Anytype. 6. Lisätty määrittelyt ”varAccess” ja ”varConfig”. 7. Lisätty globaalit tunnisteet. 8. Lisätty mahdollisuus valmistajakohtaisille tiedoille. 9. Lisätty pouInstance-osioon elementtien tyypin määrittely.
2.01	8.5.2009	Graafisen sovellusohjelman osien esitykseen vähäisiä muutoksia, jotka eivät vaikuta yhteensopivuuteen.

Taulukko 1. PLCopen XML -skeeman versiot (4, s. 8).

Uusien PLCopen XML -skeemojen julkaisutahti ei ole kovin tiheä. Viimeisimmästä julkaisusta on vierähtänyt jo kuusi vuotta, mikä verrattuna yleiseen tekniikan kehitystahtiin hyvinkin hidas.

Määrittely tarjoaa pohjan PLC-sovellusohjelmien siirrolle näitä ohjelmia käsittelevien työkalujen ja myös korkeamman tason kehitystyökalujen välille. Seuraavassa kuvassa 6 on esitetty näiden työkalujen moninaisuutta ja määrää. (5)



Kuva 6. PLCopen XML -toimintakenttä. (5)

Tämä insinöörityö käsittelee ainoastaan ohjelmointityökalun välistä PLC-sovellusohjelmien siirtoa. PLCopen XML -tiedostomuoto soveltuu ohjelmointityökalujen lisäksi myös helposti ja mainiosti etenkin tietokanta- ja dokumentointityökalujen valiseksi tallennusmuodoksi.

PLCopen XML:llä välitetyn tiedon tarkkaa vastaanottajaa ei ole tiedossa, kuten kappaleen 4.1 XML-esimerkkitapaukset. Tämän vuoksi PLCopen XML -viestiin on sisällytettävä kaikki mahdollinen informaatio jotta siirretty sovellusohjelma kyetään esittämään samankaltaisena uudelleen PLC-järjestelmän ohjelmointiympäristössä, johon se on siirretty. Tämän vuoksi PLCopen XML -tiedostoon taltioidaan paljon informaatiota siinä esitetystä sovellusohjelmasta, kuten:

- Tekstipohjaista ja graafista ohjelmakoodia,
- kaaviomaista ohjelmaa (SFC),
- graafista informaatiota, taso- ja paikkatietoa,
- kommentteja,
- laitekokoonpanon omia funktioita, toimintolohkoja ja ohjelmia (POU-yksikköjä),
- laitekokoonpanon omia tietotyyppejä,
- projekti-informaatiota,
- linkitystietoja ja
- laitetoimittajakohtaista informaatiota.

Tarjotusta tiedon määrästä kannattaa seuloa vain tarvittavat tiedot ja tarkistaa tämän tiedon paikkaansapitävyys. Näiden tietojen avulla pitää pystyä kuitenkin luomaan alkuperäistä sovellusohjelmaa vastaava sovellusohjelma, joten paikkaansapitävyyden ja tarvittavan tiedon tarkistaminen on tärkeää yhteensopivuuden kannalta. (1)

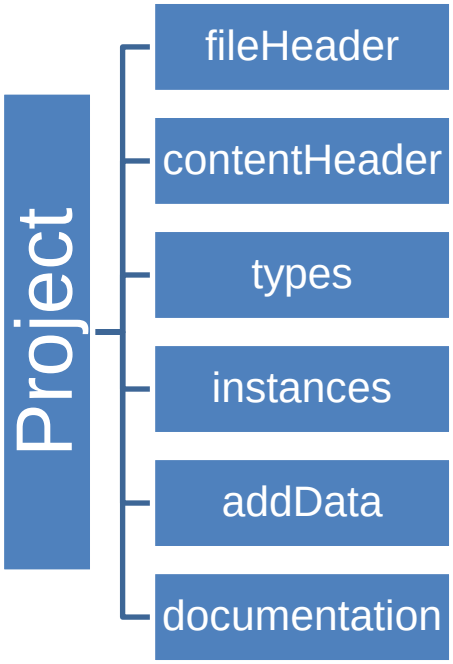
PLCopen XML -pohjainen sovellusohjelmien siirto ohjelmointiympäristöstä toiseen on tiedoston sisällön ja tätä tulkitsevan vastaanottajan osalta yhteensopivuudesta riippuvaa kuten internet-sivujen tekeminen ja lataaminen selaimeen. Internetistä ladattu HTML-tiedosto ja tämän sisältämä kuvaus internet-sivun sisältö saattavat näyttää erilaiselta kuin HTML-sivun luonut henkilö on suunnitellut, koska selain, johon sivu on ladattu tulkitsee HTML-sivun eri näköiseksi mikäli näiden yhteensopivuudessa on ongelmia.

Tämän vuoksi komitea on määritellyt PLCopen XML -skeeman, jota kaikkien hyväksyttyjen ohjelmointijärjestelmien tulee tukea. Jokainen toimittaja voi luoda ja julkaista oman skeeman, jossa toimittajakohtaiset omaisuudet ovat määritelty. (4, s 7.)

PLCopen-skeema sisältää kuvauksen PLCopen XML -tiedoston rakenteesta ja sisällöstä. PLCopen-järjestö julkaisee päivitetyn version PLCopen-skeemasta, jonka järjestössä mukana olevat yritykset ovat yhteistyönä luoneet. Lokakuussa 2014 IEC-standardisointijärjestön työryhmä IEC TC65 SC65B WG7 aloitti työn PLCopen XML -skeeman julkaisemiseksi osana IEC 61131 -standardia. Standardin osan nimitys tulee olemaan IEC 61131-10. (6)

4.3 PLCopen XML -tiedoston rakenne

Tiedostorakenne on hyvin lähellä ohjelmointiympäristöjen normaalia rakennetta, jossa on ohjelmoitavan projektin muodostavat tämän erilaiset osiot. PLCopen XML -tiedosto muodostuu aina samankaltaisesta tiedostorakenteesta huolimatta siitä miten paljon kyseisestä ohjelmistoprojektista halutaan siirtää tämän avulla. Tiedoston pääelementtinä on alla olevan kuvan 7 mukaisesti project-niminen kehys, joka määritellään PLCopen XML -skeemassa.



Kuva 7. PLCopen XML -projektihierarkia (4, s.19)

Project-kehyksen sisällä ovat tämän alikehykset, jotka sisältä siirrettävän PLC-sovellusohjelman tietyt osat. Näitä alikehyksiä ovat edellisessä kuvassa näkyvät fileHeader, contentHeader, types, instances, addData ja documentation. (4, s.19)

Tässä dokumentissa aikaisemmin käytetty esimerkki ST-kielinen sovellusohjelma on esitetty seuraavassa esimerkissä PLCopen XML -muodossa. Esimerkkiin on lisätty rivien alkuihin osioiden numerot, joihin viitataan seuraavassa sisällönkuvauksessa. Seuraavan PLCopen XML -esimerkin rivillä 1 kerrotaan PLCopen XML-skeeman versio, jota on käytetty luotaessa PLCopen XML -tiedostoa. Rivi 2 on kyseisen sovellusohjelman kuvauksen aloitusotsake eli osoittaa Project-osion alkamisen tiedostosta.

FileHeader-osio on löydettävissä riviltä 3. FileHeader-osiossa kuvataan attribuuttien avulla XML-tiedoston luonut ohjelmointiympäristö ja luontiajankohta. Tässä tapauksessa PLCopen XML -tiedosto on luotu CoDeSys-ohjelmointiympäristössä. Riviltä 4 alkavassa ContentHeader—osiossa siirretään yleiskuva XML-tiedostosta mukaan lukien tämän kuvaamiseen tarvittavia attribuutteja. Näitä ovat graafisia ohjelmointikieli varten tarvittavat koordinaattitiedot. Rivillä 5 näkyvä </contentHeader> -merkintä osoittaa tämän osion loppumisen PLCopen XML -tiedostossa. </osion nimi> -rakennetta käyttävä merkintä osoittaa XML-tiedoston kyseisen osion loppumisen.

Riviltä 6 alkavassa Types-osiossa kerrotaan tietoelementtien ja rakenneyksiköiden määrittelyt. Riviltä 7 alkaa <pous>-merkinnällä alkava osio, jossa kuvataan ohjelman sisältävät rakenneyksiköt, kuten tässä tapauksessa muuttujat ja itse ohjelmakoodi.

Instances-osiossa rivillä 8 määritellään kokoonpanot (configurations), näiden väliset yhteydet, resurssit ja globaalit muuttujat. Esimerkissä näitä ei ole määritelty eli Instances-osio on tyhjä. Riviltä 9 alkavaan addData-osioon voi ohjelmointiympäristön toimittaja sijoittaa toimittajakohtaista tietoa.

Rivillä 10 oleva </project> -merkintä osoittaa PLCopen XML-tiedostossa kuvatun sovellusohjelman (projektin) loppumisen. (4, s. 19–22)

Rivi	PLCopen XML-tiedoston rakenteen osiot
1	`<?xml version="1.0" encoding="utf-8"?>`
2	`<project xmlns="http://www.plcopen.org/xml/tc6_0200">`
3	`<fileHeader companyName="" productName="CODESYS" productVersion="CODESYS V3.5 SP4 Patch 2" creationDateTime="2015-02-25T20:41:47.368" />`

4	`<contentHeader name="st.project" modificationDateTime="2015-01-25T20:41:30.937">`
	`<coordinateInfo>`
	`<fbd>`
	`<scaling x="1" y="1" />`
	`</fbd>`
	`<ld>`
	`<scaling x="1" y="1" />`
	`</ld>`
	`<sfc>`
	`<scaling x="1" y="1" />`
	`</sfc>`
	`</coordinateInfo>`
	`<addData>`
	`<data name="http://www.3s-software.com/plcopenxml/projectinformation" handleUnknown="implementation">`
	`<ProjectInformation />`
	`</data>`
	`</addData>`

5	`</contentHeader>`
6	`<types>`

7	`<pous>`
	`<pou name="PLC_PRG" pouType="program">`
	`<interface>`
	`<localVars>`
	`<variable name="CMD">`
	`<type>`

	<INT /> </type> </variable> <variable name="AUTO_CMD"> <type> <INT /> </type> </variable> </localVars> </interface> <body> <ST> <xhtml xmlns="http://www.w3.org/1999/xhtml"> CMD := AUTO_CMD & AUTO_MODE OR MAN_CMD & NOT MAN_CMD_CHK & NOT AU- TO_MODE ; CMD_TMR (IN := CMD, PT := T_CMD_MAX); ALRM_FF (S1 := CMD_TMR.Q & NOT FDBK, R := ACK); ALRM := ALRM_FF.Q1; </xhtml> </ST> </body> <addData /> </pou> </pous> </types>
8	<instances> <configurations /> </instances>
9	<addData> <data name="http://www.3s- software.com/plcopenxml/projectstructure" handleUnknown="discard"> <ProjectStructure> <Object Name="PLC_PRG" /> </ProjectStructure> </data> </addData>
10	</project>

Taulukko 2. PLCopen XML-tiedoston rakennemalli.

PLCopen XML -tiedoston tulee noudattaa tätä rakennemallia, jotta sen tulkitseminen onnistuu ohjelmointiympäristössä, johon PLC-sovellusohjelmaa ollaan siirtämässä. Tiedostorakenteen osan puuttuminen tai skeemaan kuulumattoman osan löytyminen sekoittaa PLCopen XML-tiedoston lataamistoiminnon ohjelmointiympäristössä. Ohjelmointiympäristöt osaavat ilmaista jossakin määrin nämä puuttuvat ja odottamattomat tiedostorakenteet

5 Ohjelmointiympäristöt

Konecranes käyttää useiden valmistajien ohjelmointiympäristöjä. Näistä valitsimme tämän työn tarkastelun kohteiksi ohjelmistoympäristöt valmistajilta Siemens, Beckhoff, CoDeSys ja KW-Software.

5.1 Siemens

Siemensin uusin TIA Portal -ohjelmointiympäristö yhdistää Siemensin Simatic Step 7 -logiikkaohjelmoinnin, Simatic WinCC -käyttöjärjestelmäsuunnittelun ja Sinamics StartDrive -taajuusmuuttajat. (7)

Ohjelmistoympäristön Simatic Step 7 -osio tukee IEC 61131-3 -standardin mukaisia graafisia LD- ja toimilohkokaavioita (FBD). Siemensin SCL (Structured Control Language) -kielen tulisi Siemensin mukaan vastaavan IEC 61131-3 -standardin vaatimuksia ST-kielelle kuten myös Siemensin STL (Statement List) -kielen tulisi vastata IEC 61131-3 -standardin IL-kielen vaatimuksia. Standardin IEC 61131-3 SFC-ohjelmointikieltä vastaa Siemensin S7-Graph. Siemensin ohjelmointiympäristön käyttämät ohjelmointikielet eivät ole kuitenkaan täysin yhteensopivia muiden valmistajien vastaavien ohjelmointikielien kanssa. (8)

Testejä tehdessä tarjolla ollut Siemensin ohjelmointiympäristö TIA PORTAL V13 ei valitettavasti tue PLCopen XML -tiedostomuotoa. Tehdäkseen PLCopen XML -siirtoa vastaavia sovellusohjelman siirtoja Siemensin Step 7 -ohjelmointiympäristön ja muiden valmistajien ohjelmointiympäristöjen välillä on käytettävä erillisiä muunnossovelluksia.

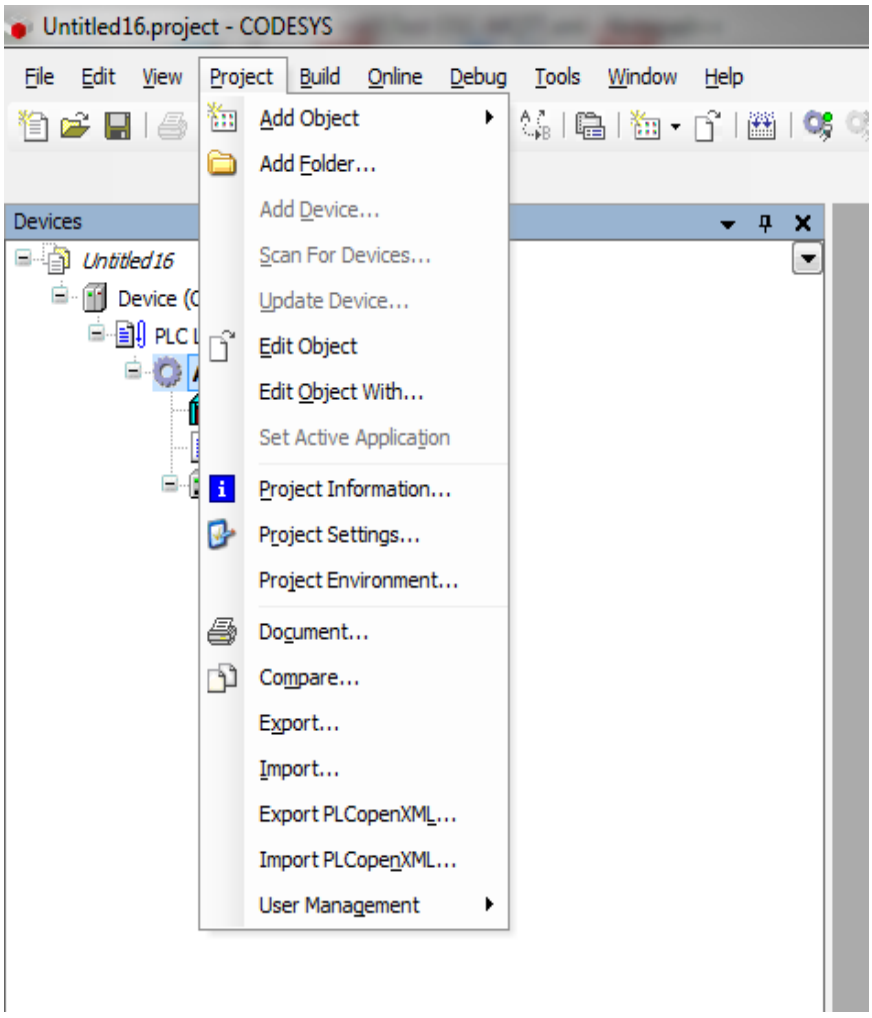
5.2 CoDeSys

Testissä käytetty ohjelmistoversio on CoDeSys V3.5 SP4. Ohjelmiston ilmoitetaan pystyvän tuottamaan sekä lukemaan PLCopen XML -tiedoston. CoDeSys-ohjelmiston kehittäjät ovat olleet PLCopen-työryhmässä aktiivisesti mukana.

CoDeSys-ohjelmiston valmistaja 3S-Smart Software Solutions GmbH ilmoittaa PLCopen XML -toimintojen puutteiksi kaksi asiaa. Ohjelmisto ei tue IL-ohjelmointikielen

(Information List) rakenneyksiköiden siirtoa PLCopen XML-muodossa. Toinen mainittu seikka on oikeastaan puute PLCopen XML -tiedostomuodossa, koska PLCopen XML -tiedostomuoto ei salli sekä VAR_GLOBAL- ja VAR_GLOBAL CONSTANT- osioita yhdessä muuttujalistassa. Tällöin he kehottavat erottamaan nämä osiot kahteen eri muuttujaluetteloon. (9)

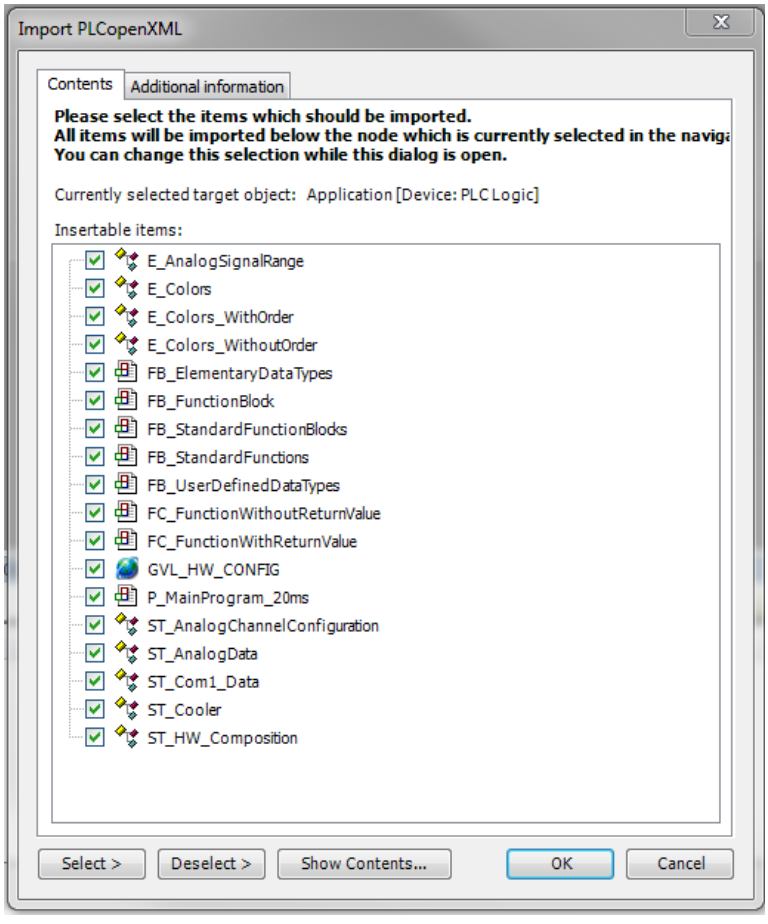
PLCopen XML -toiminnot löytyvät helposti ohjelmiston Project-alasvetovalikosta kuvan 8 mukaisesti. CoDeSys-ohjelmointiympäristössä käytössä oleva sovellusohjelma (project) tai osa siitä saadaan PLCopen XML -muotoon kohdasta Export PLCopenXML. Import PLCopenXML -kohdasta saadaan tuotua PLCopen XML-tiedosto kokonaisuudessaan tai osa siitä käytössä olevaan CoDeSys-sovellusohjelmaan.



Kuva 8. PLCopen XML -toimintojen sijainti CoDeSys 3.5:ssä.

CoDeSys-projektiin tuotavasta PLCopen XML -tiedostosta voidaan valita halutut osiot kuvan 9 mukaisessa valintaikkunassa. Suuremman projektin sisältävästä PLCopen XML-tiedostosta pystytään valitsemaan vain haluttu osa toiseen projektiin mukaan. Additional information -välilehdeltä on luettavissa PLCopen XML-tiedoston File header- ja Content head -osioiden sisällöt. Näiden tietojen avulla voidaan esimerkiksi tarkistaa että käytössä on juuri haluttu PLCopen XML -tiedosto.

Vastaavanlainen valintaikkuna on myös PLCopen XML -tiedoston luontitoiminnossa. Tämän avulla pystytään valitsemaan yksitellen PLCopen XML -tiedostoon mukaan halutut sovellusohjelman osiot. Suuremmasta CoDeSys-projektista voidaan siirtää vain haluttu osa PLCopen XML-muodossa.



Kuva 9. CoDeSys-projektiin PLCopen XML -tiedostosta tuotavien osioiden valitseminen.

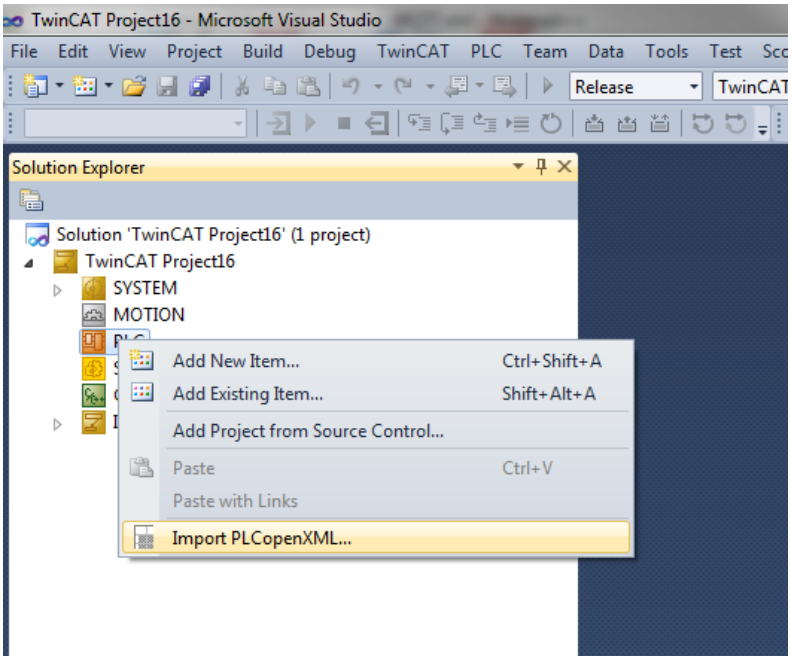
Ohjelmiston käyttämä PLCopen XML -skeeman versio on 2.0. CoDeSys-ohjelmiston kehittäjä ei ole vielä katsonut tarpeelliseksi käyttää uusinta PLCopen XML-skeeman versiota 2.01. Tämä selittyy versioiden välisillä vähäisillä muutoksilla.

5.3 TwinCAT

Testissä on käytetty ohjelmisto TwinCAT XAE (eXtended Automation Engineering), jossa Beckhoffin TwinCAT 3 -ohjelmointiympäristö on yhdistetty Microsoftin Visual Studio -ympäristöön. Beckhoffin TwinCAT on rakennettu edellisessä kappaleessa esitellyn CoDeSys-ohjelmointiympäristön pohjalta. Myös tämän ohjelmiston ilmoitetaan pystyvän tuottamaan sekä lukemaan PLCopen XML -tiedoston.

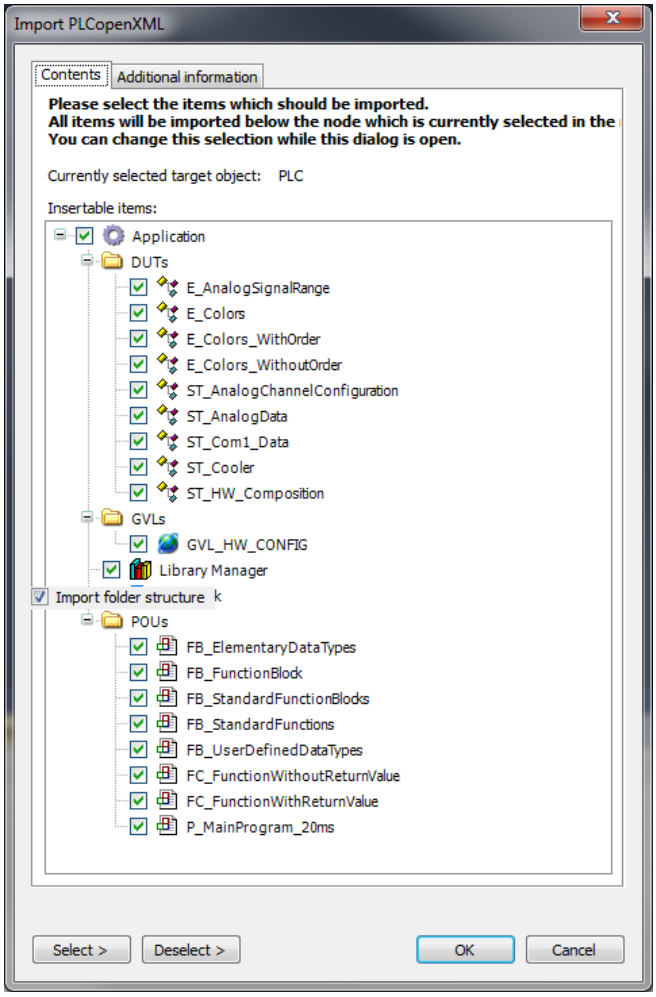
TwinCAT-ohjelmiston valmistaja Beckhoff Automation GmbH ilmoittaa PLCopen XML -toimintojen puutteiksi saman kuin CoDeSys-ohjelmiston puutteenakin mainittiin. Puutteena on se että PLCopen XML -skeema ei salli sekä VAR_GLOBAL- ja VAR_GLOBAL CONSTANT- osioita yhdessä muuttujalistassa. Tällöin hekin kehoittavat erottamaan nämä osiot kahteen eri muuttujaluetteloon. (10)

TwinCAT-ohjelmointiympäristössä PLCopen XML -tiedostojen käsittelyyn ei ole valintaa ohjelmointiympäristön alasvetovalikoissa. PLCopen XML-tiedoston tuominen käynnissä olevaan projektiin tapahtuu hiiren oikealla painikkeella ohjelmoitavaa logiikkaa kuvaavan PLC-ikonin kohdalla. Tuolloin aukeaa kuvan 10 mukainen käytössä olevien toimintojen valikko. Tästä voidaan valita Import PLCopenXML-valinta.



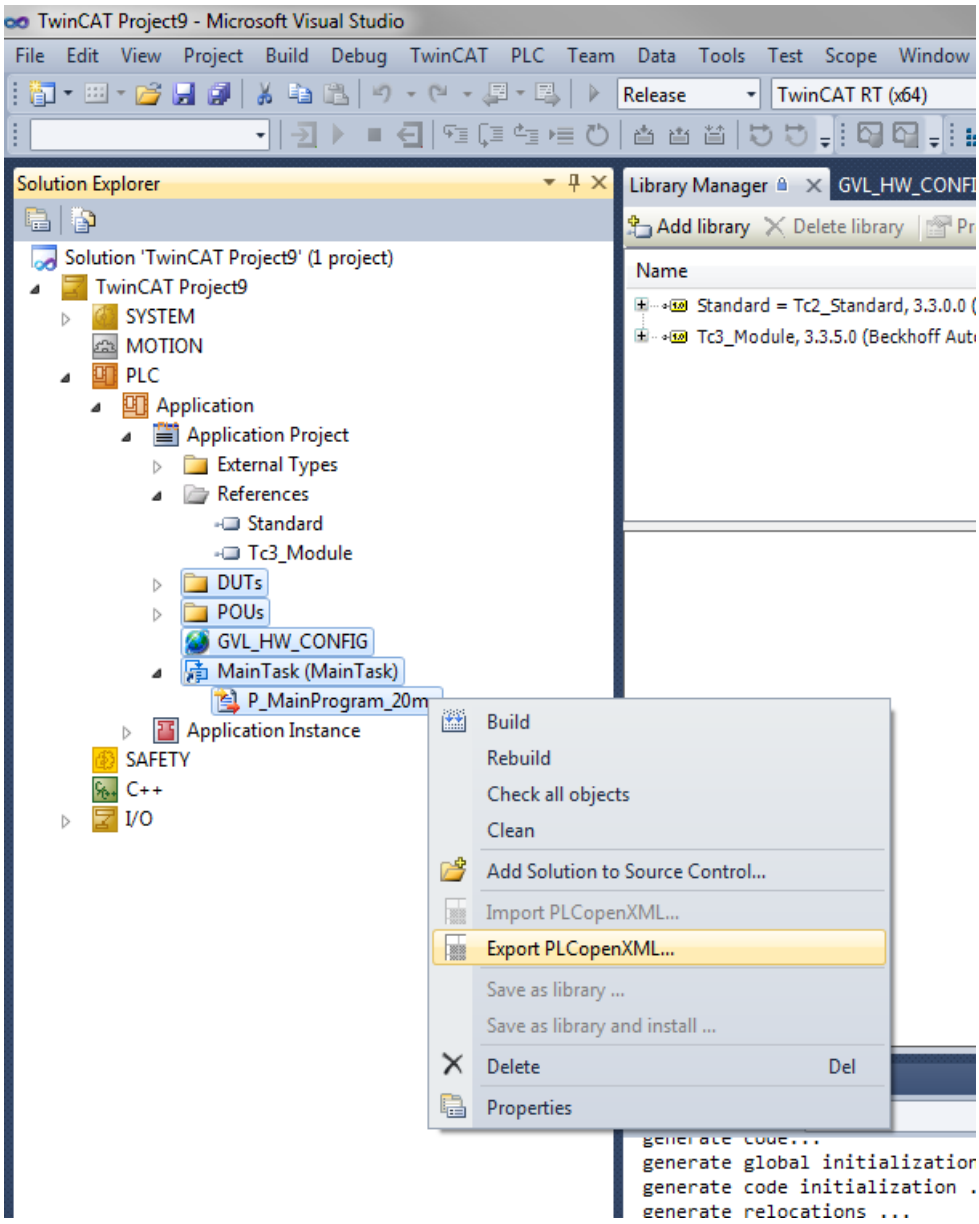
Kuva 10. TwinCAT-projektiin PLCopen XML -tiedoston tuonti.

Tällöin päästään aivan normaalin Windows-käyttöjärjestelmän tiedoston valintatyökalun avulla valitsemaan haluttu PLCopen XML -tiedosto. Halutun XML-tiedoston valinnan jälkeen avautuu seuraava kuvan 11 mukainen Import PLCopenXML -ikkuna, josta voidaan valita XML-tiedoston sisällöstä siirrettäväksi halutut osiot. Kuten CoSeSys-ohjelmointiympäristössäkin on tämän ikkunan Additional information -välilehdeltä on luettavissa PLCopen XML-tiedoston File header- ja Content head -osioiden sisällöt.



Kuva 11. TwinCAT:ssä XML-tiedoston sisällön valinta.

TwinCAT-ohjelmointiympäristössä PLCopen XML -tiedoston luonti on toteutettu hieman eri tavalla kuin CoDeSys-ohjelmointiympäristössä. XML-tiedostoon mukaan valittavat projektin osat valitaan Solution Explorer -ikkunasta. Solution Explorer -ikkuna on oletuksena normaalin käyttöliittymän vasemmassa reunassa. Export PLCopen XML -toimintoon päästään painamalla hiiren oikeaa painiketta näiden valittujen projektin osien kohdalla. Tällöin aukeaa seuraava kuvan 12 mukainen valikko, josta voidaan valita Export PLCopen XML -toiminto.



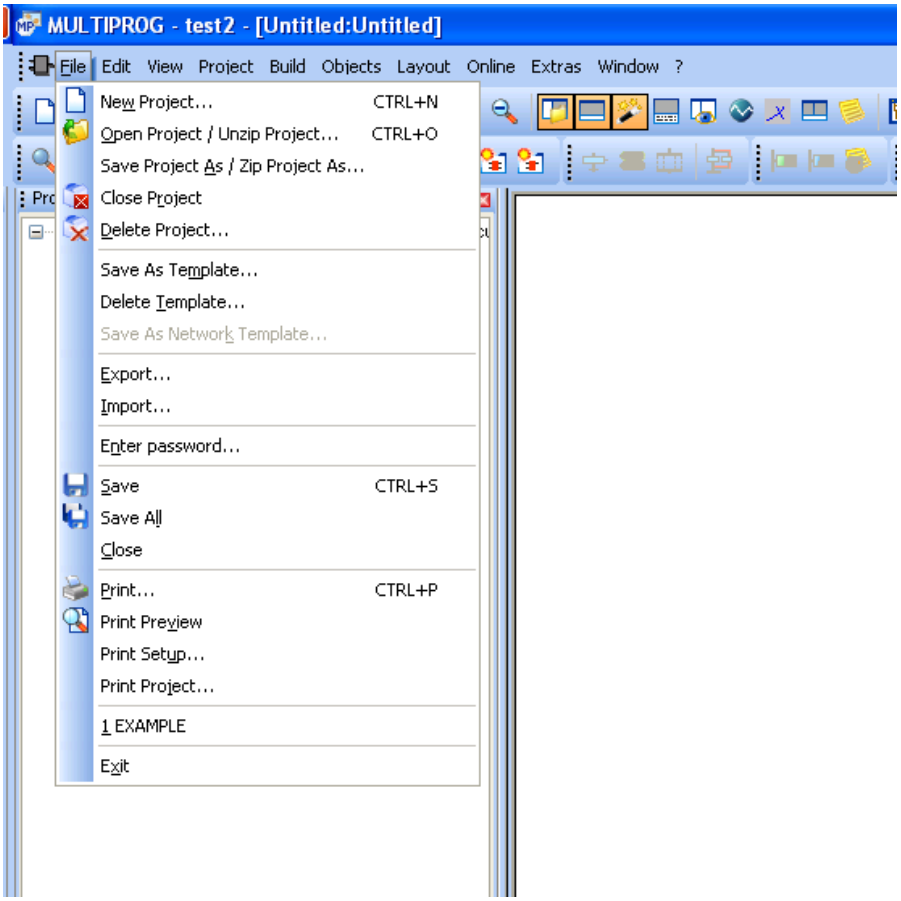
Kuva 12. XML-tiedoston luonti TwinCAT:ssä.

Ohjelmiston käyttämä PLCopen XML -skeeman versio on 2.0. TwinCAT-kehittäjä ei ole vielä katsonut tarpeelliseksi käyttää uusinta PLCopen XML-skeeman versiota 2.01.

5.4 Multiprog

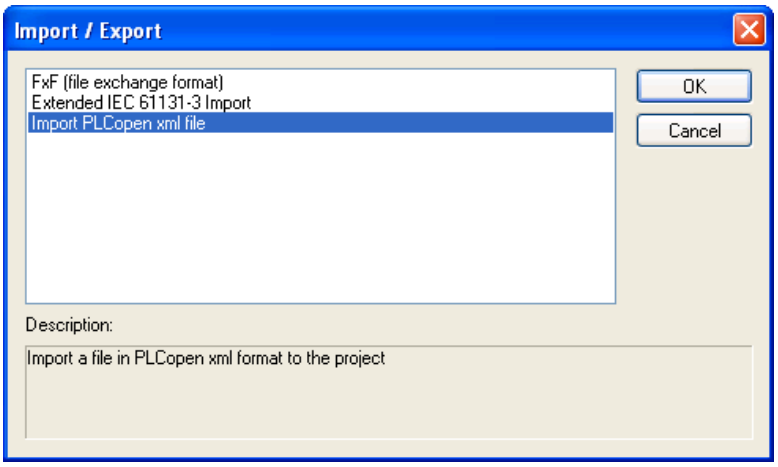
Testissä käytetty Phoenix Contact Software GmbH nimisen (aikaisemmin KW-software GmbH) ohjelmointiympäristö on Multiprog 5.50 (Build 371). Multiprog-ohjelmointiympäristön on myös mainittu tukevan PLCopen XML -siirtoa.

Ohjelmointiympäristössä käytössä oleva sovellusohjelma (projekti) pitää olla valittuna Project Tree -ikkunasta että pystyy valitsemaan kuvassa 13 näkyvästä ohjelmointiympäristön File-alasvetovalikosta Export- ja Import-toiminnot.



Kuva 13. MultiProg-ohjelmointiympäristön Export- ja Import-toiminnot.

Halutun siirtosuunnan valitsemisen jälkeen ohjelmisto kyselee siirrettävän tiedoston tiedostomuotoa eli formaattia. Tässä on valittavissa PLCopen XML -tiedostomuoto kuvan 14 esittämästä ikkunasta.



Kuva 14. Siirrettävän tiedostomuodon valinta Multiprog-ohjelmointiympäristössä.

Kun ollaan tuomassa Multiprog-ohjelmointiympäristöön PLCopen XML -muotoista tiedostoa, niin ohjelmisto pyytää valitsemaan halutun PLCopen XML -tiedoston.

PLCopen XML-tiedoston muodostaminen Multiprog-ohjelmointiympäristön ohjelmasta suoritetaan Export-toiminnolla. Siirtoon mukaan otettava sovellusohjelma tai sovellusohjelman osat tulee olla valittuna Project Tree -ikkunasta. Tämän jälkeen ohjelmisto kyselee luotavan tiedostonsiirtomuodon eli tässä tapauksessa valitaan Export PLCopen xml file -valinta edellä näkyvästä ikkunasta. Tämän jälkeen valitaan haluttu tallennuspaikka ja syötetään PLCopen XML -tiedoston nimi.

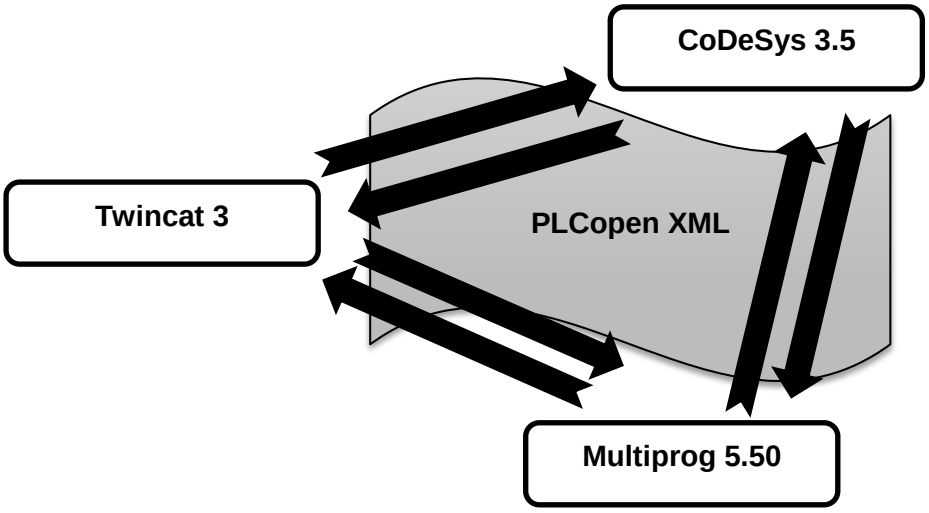
Multiprog-ohjelmointiympäristön dokumentoinnin mukaan tämä tukee PLCopen XML -skeema versioita 0.99 ja 1.01 (11). Ottaen huomioon tämän PLCopen XML-skeeman version iän, 10 vuotta, tuntuu oudolta ettei tätä ole päivitetty.

6 PLCopen XML -testaus

Testauksen tarkoituksena oli selvittää kuinka Konecranes Oyj:n käyttämät, edellä kuvatut ohjelmointiympäristöt tukevat PLCopen XML -muotoista sovellusohjelman siirtoa. Valitettavasti Siemensin TIA Portal -ohjelmointiympäristö jouduttiin jättämään pois testistä, koska tämän ei edes luvattu tukevan PLCopen XML -tiedostomuotoa. Testissä siirrettiin kolmen jäljelle jääneen ohjelmointiympäristöjen välillä Konecranes Oyj:n toimittamaa ST-kielistä testisovellusohjelmaa sekä muita LD- ja FBD-

ohjelmointikielillä toteutettuja testisovellusohjelmia, joiden avulla voitiin tutkia PLCopen XML -siirtotavan toimivuutta.

Testissä mukana olleiden kolmen ohjelmointiympäristöjen välillä siirrettiin testisovellusohjelmia kunkin ohjelmointiympäristön tukeman PLCopen XML-version mukaisilla tiedostoilla. Testin tarkoituksena on siirtää kaikista kolmesta ohjelmointiympäristöistä sovellusohjelmia PLCopen XML -muodossa kahteen muuhun ohjelmointiympäristöön kuvan 15 mukaisella tavalla



Kuva 15. Testijärjestely

Näiden PLCopen XML -tiedostojen rakenteen ja sisällön tarkastamiseen ei valitettavasti ollut saataville tähän tarkoitukseen varta vasten olevaa sovellusta, joka osaisi erottaa puutteet tiedostossa verrattuna haluttuun XML-skeemaan. Tarkastuksessa oli tyydyttävä testissä mukana olevien ohjelmointiympäristöjen PLCopen XML -tiedoston sisäänlukutoimintoon (import) ja tekstieditorilla tapahtuvaan silmämääräiseen tarkastukseen siltä osin että kaikki tarvittavat PLCopen XML-tiedoston rakenteelliset osat ovat mukana. Tähän tarkoitukseen sopivaksi sovellukseksi osoittautui Notepad++, joka kykenee tekstin värein erottelemaan XML-tiedoston rakenteen osiot. Tämän avulla XML-tiedosto on huomattavasti luettavampaa. Mahdollisien PLC-ohjelmien siirto-ongelmia olisi helpompaa ja nopeampaa tutkia mikäli PLCopen-järjestö tarjoaisi avoimesti käyttöön esimerkiksi testaussovelluksen, jolla voisi tarkistaa PLC-ohjelmasta luodun PLCopen XML -tiedosto on tietyn PLCopen XML -skeeman mukainen.

6.1 PLCopen XML -tiedoston validointi

Ohjelmointiympäristöjen PLCopen XML -tiedostojen sisäänlukutoiminnot suorittavat luettavalle tiedostolle validoinnin eli tarkistuksen. Tässä yhteydessä sisäänlukutoiminto vertailee XML-tiedostoa ohjelmointiympäristön käyttämään PLCopen XML-skeemaan ja suorittaa XML-tiedostolle jäsentelyn, jonka mukaan XML-tiedoston sisällöstä luodaan ohjelmointiympäristöön PLC-sovellusohjelma. Jäsentelyssä sisäänlukutoiminto tulkitsee sillä ladattavan XML-tiedoston PLCopen XML -skeeman ja mahdollisen laitevalmistajakohtaisen skeeman avulla, jolloin sisäänlukutoiminto osaa muodostaa alkuperäisestä XML-tiedostosta löytyneet sovellusohjelman rakenteet ja osat. Mikäli PLCopen XML -skeema ja sisäänluettava XML-tiedosto eivät ole yhteensopivia niin osa siirretystä PLC-sovellusohjelmasta jää uupumaan siirrosta tai tämän siirto ei onnistu lainkaan. XML -tiedostosta voi löytyä myös ohjelmiston käytössä olevien skeemojen osalta tuntemattomia XML-tiedoston rakenteita. Näitä sisäänlukutoiminto ei kykene tällöin tulkitsemaan ja näitä XML-tiedoston rakenteita ei luoda kohteena olevaan sovellusohjelmaan. PLCopen XML -skeema toimii siis lähes kuin salakirjoituksen purkuavain. Ellei salauksen purkuavain ole vastaava kuin salausavain, jolla informaatio on salattu, niin informaatio ei ole tulkittavissa oikein. (4, s.12–14).

Jos jokin PLC-sovellusohjelma on luotu IEC 61131-3 -yhteensopivalla ohjelmointiympäristöllä ja siirretty toiseen vastaavaan ohjelmointiympäristöön PLCopen XML -skeemaa käyttäen, niin PLC-sovellusohjelman tulisi olla samanlainen kummassakin ohjelmointiympäristössä. PLCopen-työryhmä TC3 määrittelee ja kehittää testausohjelmistoja ohjelmointiympäristöjen standardin IEC 61131-3 yhteensopivuuden testaamiseksi. Näiden perusteella ohjelmointiympäristöjen valmistajien on pystyttävä esittämään vaatimuksenmukaisuuslausunto ohjelmointiympäristönsä standardin IEC 61131-3 mukaisuudesta. PLCopen-järjestöllä ei ole määritelty testejä PLCopen XML-yhteensopivuudesta.

6.2 ST-testikoodi

Testeissä käytetty ST-ohjelmakoodi on saatu Konecranes edustajalta. Tämä edustaa heidän käyttämiään sovellusohjelman toteutustapoja. Tämän avulla pystytään varmistamaan että testit ja näiden tulokset vastaavat heidän tarpeitaan ohjelmien

siirrosta eri ohjelmointiympäristöjen välillä. Testikoodi toimitettiin CoDeSys-ohjelmistoympäristön projektina.

6.3 Graafisten ohjelmointikielien esitys PLCopen XML -muodossa

Graafisten LD- ja FBD-ohjelmien muuttaminen XML-koodiksi lisää XML-koodin rivien määrää huomattavasti. Koodissa pitää esittää eri graafisien esitystapojen vaatimia kuvion muodostamiseen tarvittavia parametrejä kuten graafisen sovellusohjelman osien koordinaatteja. Tässä yhteydessä käytetyt sovellusohjelmat eivät ole Konecranesin toimittamia testiohjelmia. Graafisien testiohjelmien sisällössä käytettiin ohjelmoinnissa tavanomaisia objekteja ja toimintoja.

6.4 MultiProg-ohjelmointiympäristön toiminta

PLCopen XML -tiedoston tuontia varten löytyy toiminto Multiprog-ohjelmointiympäristöstä ja tämä havaitsee virheellisen XML-tiedoston, mutta oikeaksi testatun XML-koodin tuloksena ei tapahdu mitään. Multiprog ei anna mitään virheilmoitusta epäonnistuneesta siirrosta, mutta käytössä olevaan sovellusohjelmaan ei tule siirrettyä sovellusohjelmaa tai siitä valittua ohjelman osaa.

Multiprog-ohjelmointiympäristössä PLCopen XML -tiedoston luonti ohjelmointiympäristössä luodusta sovellusohjelmasta onnistuu ongelmitta. Valitettavasti tämän PLCopen XML -tiedoston siirto CoDeSys-ohjelmointiympäristöön ei onnistu. Suurin osa Multiprog-ohjelmointiympäristössä luoduista PLCopen XML -tiedostojen sisällöistä jäi siirtymättä CoDeSys- ja TwinCAT-ohjelmointiympäristöihin. CoDeSys ja TwinCAT muodostivat hyvin samankaltaiset vajavaiset sovellusohjelmat ja osin samoja lukemattomia virheilmoituksia näistä PLCopen XML-tiedostoista.

PLCopen XML -tiedostojen avulla sovellusohjelmien siirto Multiprog ohjelmointiympäristöön ei siis onnistu. Myöskään Multiprog-ohjelmointiympäristössä luotu PLCopen XML -tiedoston siirtäminen muihin testissä mukana olleisiin ohjelmointiympäristöihin ei onnistu. Syynä tämän on hyvin suurella todennäköisyydellä muista testattavana olevista ohjelmointiympäristöistä poikkeava PLCopen XML -

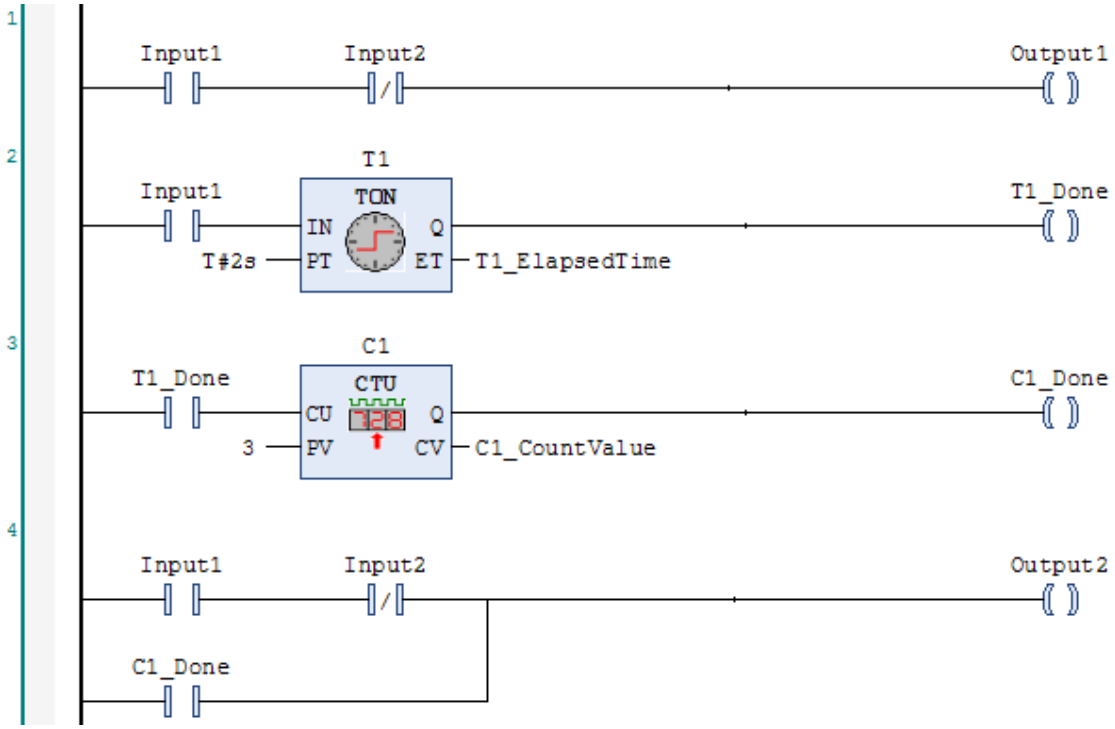
skeeman versio. Multiprog 5.50 tukee PLCopen XML -skeema versioita 0.99 ja 1.01, mutta CoDeSys ja TwinCAT tukevat versiota 2.0.

Multiprog-ohjelmointiympäristöä valmistavan Phoenix Contact Software GmbH nimisen yrityksen tuotetuki kertoi ohjelmointiympäristössä edelleen olevista ohjelmointivirheistä, jotka vaikuttavat PLCopen XML -tiedoston siirtotoiminnon käytettävyyteen. Tuotetukeen otettiin yhteyttä, jotta voitiin varmistaa että testissä käytössä oleva Multiprog 5.50 tukee valmistajan mukaan PLCopen XML -muotoista sovellusohjelman siirtoa.

6.5 LD-sovellusohjelman siirto PLCopen XML -muodossa

CoDeSys-ohjelmistoympäristöllä luotu PLCopen XML -koodi LD-sovellusohjelmasta, joka mahtuu muuttujamäärittelyineen ja kuvineen yhdelle tämän työn sivulle, muodostuu yli 300 rivistä XML-koodia. Alla olevan lyhyen LD-sovellusohjelman version 2.0 PLCopen XML -koodi on esitetty liitteessä 1. Tästä liitteen 1 PLCopen XML -koodista on löydettävissä muun muassa ”<scaling x="1" y="1" />” ja ”<position x="0" y="0" />” -rivejä, joiden avulla XML-koodissa esitetään LD-ohjelman osien sijainteja ohjelmaa kuvaavassa kaaviossa (kuva 16).

```
FUNCTION_BLOCK POU
VAR
          T1: TON;
          T1_Done: BOOL;
          T1_ElapsedTime: TIME;
          C1: CTU;
          C1_Done: BOOL;
          C1_CountValue: WORD;
END_VAR
```

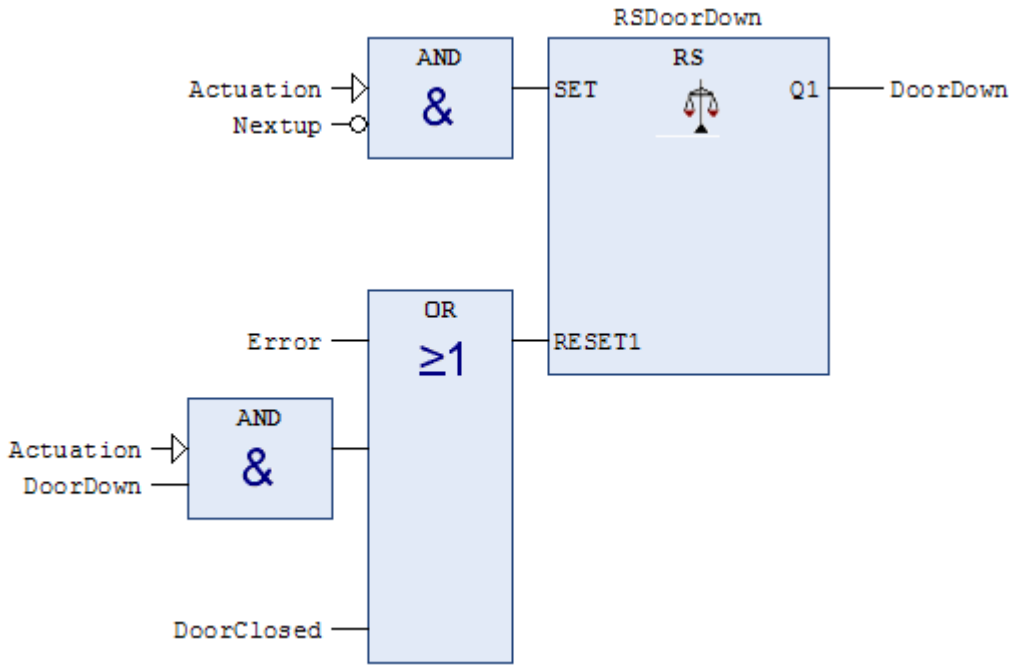


Kuva 16. Esimerkki LD-ohjelmasta

LD-sovellusohjelman siirto onnistui CoDeSys-ympäristöstä TwinCAT-ympäristöön ongelmitta, kuten myös TwinCAT:stä CoDeSys:in. Kummatkin ohjelmointiympäristöt kykenivät luomaan ja tulkitsemaan LD-sovellusohjelman version 2.0 PLCopen XML -muodossa, niin että tässä siirtyi kaikki siirrettävään ohjelmaan kuuluva informaatio. Multiprog-ohjelmointiympäristön luomaa version 1.0 PLCopen XML -tiedostoa CoDeSys- ja TwinCAT-ohjelmointiympäristöt eivät kyenneet tulkitsemaan oikealla tavalla. Tämän XML-siirron tuloksesta jäi pois PLC-sovellusohjelman määrittelyjä ja rakenteita, PLCopen XML -validointi ei onnistu eri skeemaversioiden takia.

6.6 FBD-sovellusohjelman siirto PLCopen XML -muodossa

FBD-sovellusohjelman PLCopen XML -esitystä koskee aivan samat piirteet kuin edellisen kappaleen LD-sovellusohjelmaankin. Tässäkin graafisen esityksen kuvaaminen lisää XML-koodin rivien määrää huomattavasti. Sovellusohjelman siirtäminen Codesys-ympäristöstä TwinCAT-ympäristöön PLCopen XML -koodina onnistui ongelmitta. Alla olevan kuvan 17 mukainen FBD-ohjelma on PLCopen XML -muodossa liitteessä 2.



Kuva 17. Esimerkki FBD-sovellusohjelmasta

Tämän FBD-sovellusohjelman osan kuvaaminen PLCopen XML -muodossa vaatii 160 riviä XML-koodia. TwinCAT-ohjelmointiympäristöstä saadaan FBD-ohjelma PLCopen XML -tiedostona ongelmitta ja tämän siirtäminen CoDeSys-ohjelmointiympäristöön onnistuu ongelmitta.

FBD-sovellusohjelman tulee olla PLCopen XML -muunnosta tehtäessä käännettävissä ohjelmoitavaa logiikkaa varten virheittä. Mikäli FBD-sovellusohjelmasta puuttuu esimerkiksi muuttujien määrittelyitä, niin tällöin PLCopen XML -koodista tulee puutteellista. Tämä ilmenee vasta kun PLCopen XML -koodia siirretään toiseen ohjelmointiympäristöön.

Kun CoDeSys-ohjelmointiympäristössä suoritetaan sisäänlukua puutteelliselle PLCopen XML—koodille niin vasta tässä vaiheessa CoDeSys antaa yksityiskohtaiset virheilmoitukset puutteellisista koodin kohdista.

CoDeSys- ja TwinCAT-ohjelmointiympäristöjen välillä siis kykenee täydellisesti siirtämään FBD-sovellusohjelman version 2.0 PLCopen XML -muodossa. Multiprog-ohjelmointiympäristön luomaa version 1.0 PLCopen XML -tiedostoa CoDeSys- ja TwinCAT-ohjelmointiympäristöt eivät kyenneet tulkitsemaan oikealla tavalla ja tästä jäi pois merkittävä määrä sovellusohjelman määrittelyjä ja rakenteita.

6.7 ST-sovellusohjelman siirto CoDeSys:stä TwinCAT:iin

CoDeSys-ohjelmointiympäristö muodostaa ongelmitta XML-tiedoston testisovellusohjelmasta, ilman virheilmoituksia. TwinCAT-ohjelmointiympäristö havaitsee puutteen XML-tiedostossa liittyen osuuteen, jossa määritellään testikoodin globaaleja muuttuja. CoDeSys on jättänyt pois osion tunnuksen, jonka olettaisi olevan ”configVars” CoDeSys-koodin mukaisen osiomäärittelyn mukaisesti. Yllättäen tämän pitää kuitenkin olla ”globalVars”, jotta tiedosto olisi PLCopen XML -standardin mukainen. Osion tunnuksen lisääminen tekstitiedoston muokkausohjelmalla osion alkuun ja loppuun mahdollistaa XML-tiedoston lukemisen TwinCAT:iin.

TwinCAT ei kuitenkaan ole tyytyväinen tähän XML-tiedoston avulla siirrettyyn sovellusohjelmaan eli tämän kääntäminen ei onnistu TwinCAT:ssä. TwinCAT ei tunnista XML:n mukana tullutta standard-kirjastoa, vaan haluaa tietysti käytettävän omaa Tc2-Standard-kirjastoa. Tämän lisäksi on lisättävä mukaan TwinCAT:n Tc3_Module -kirjasto.

Aikaisemmin jo XML-koodissa muokattu globaalien muuttujien määrittely vaatii myös TwinCAT:ssä muutoksen muuttujien määrittelyille. Nämä on vaihdettava VAR_GLOBAL-muuttujista VAR_CONFIG-muuttujiksi. Tämän jälkeen ohjelmakoodin kääntö onnistuu ongelmitta.

6.8 ST-testikoodin siirto TwinCAT:stä CoDeSys:iin

TwinCAT:n PLCopen XML -koodin muodostustoiminne lisää myös XML-tiedostoon TwinCAT:in omat External Types-määrittelyt. Nämä aiheuttavat XML-tiedoston koon huomattavan kasvamisen. Käytössä olevalla testikoodilla XML-tiedoston koko on noin 70 kilotavua, mutta kun External Types -määrittelyt ovat mukana niin tiedoston koko on n. 750 kilotavua. Rivimäärä kasvaa myös n. 2200 rivistä n. 18500 riviin.

Globaalien muuttujien kohdalla CoDeSys toimii vastaavalla tavalla kuin TwinCAT. CoDeSys:iin tuotavassa XML-tiedostossa globaalit muuttujat pitää määritellä globalVars-muuttujiksi, mutta CoDeSys:ssä nämä pitää muuttaa configVars-tyyppisiksi.

Siirrettäessä sovellusohjelma PLCopen XML -muodossa TwinCAT:stä CoDeSys:iin jää osa rivien lopetusmerkeistä eli puolipilkuista ”;” saattaa jäädä pois. Tämän ilmiön kohdalla on havaittavissa vaihtelua, jonka aiheuttajaa ei löytynyt.

7 Yhteenveto

PLCopen XML -muotoisen tiedonsiirron avulla mahdollisuus siirtää PLC-laitteiden sovellusohjelmia eri laitevalmistajien ohjelmointiympäristöjen välillä jää hyvin rajatuksi. Tämän työn yhteydessä on pystytty siirtämään sovellusohjelmia ainoastaan CoDeSys- ja TwinCAT-ohjelmointiympäristöjen väillä. Tämä ei riittäne työn tarkoituksena olleeseen sovellusohjelmakoodien ylläpitämisen helpottamiseen ja tarvittavan työmäärän vähentämiseen merkitsevästi. Myös yhteenvetotaulukosta 3 on havaittavissa, ettei ohjelmistoympäristöjen PLCopen XML -yhteensopivuus ei ole vielä läheskään riittävän kattava, jotta tätä PLC-sovellusohjelmien siirtotapaa pystyttäisiin hyödyntämään järkevästi.

Verrattavat ohjelmointiympäristöt	**CoDeSys**	**TwinCAT**	**Multiprog**	**Siemens**
CoDeSys	OK	OK	Ei toimi	Ei tue
TwinCAT	OK	OK	Ei toimi	Ei tue
Multiprog	Ei toimi	Ei toimi	OK	Ei tue
Siemens	Ei tue	Ei tue	Ei tue	Ei tue

Taulukko 3. Yhteenveto ohjelmointiympäristöjen PLCopen XML -yhteensopivuudesta.

Taulukossa 3 pitäisi olla enemmän PLCopen XML -tuen omaavia ohjelmistoympäristöjä. Etenkin PLCopen XML-tuen puuttuminen yleisesti käytetystä Siemensin TIA Portal -ohjelmointiympäristöstä vie pohjaa tämän PLCopen XML -muotoisen sovellusohjelman siirron käyttämiseltä. Mikäli käytössä olisi ainoastaan CoDeSys- ja TwinCAT-ohjelmointiympäristöt niin tämän hetkinen PLCopen XML -tuki olisi riittävä.

TwinCAT 3-ohjelmointiympäristö perustuu CoDeSys 3-ohjelmointiympäristöön, mutta TwinCAT 3 on yhdistetty Microsoftin Visual Studio -ohjelmointiympäristöön. Tämä selittää, miksi nämä ovat hyvin yhteensopivia ja tukevat yhtäläisesti PLCopen XML -suositusta. Koska muutkin PLC-ohjelmointiympäristöjen valmistajat ovat mukana PLCopen XML -kehitystyössä, niin on suuri pettymys ettei heidän ohjelmointiympäristönsä tue PLCopen XML:ä. Oletettavasti tämä johtuu siitä että laitevalmistajat yrittävät pitää kiinni valmistamiensa laitteiden ja ohjelmointiympäristöjen asiakkaista. Tämän avulla saatetaan varmistaa ettei heidän asiakkaansa kynnys hankkia muiden valmistajien laitteita ja ohjelmointiympäristöjä pääse laskemaan.

PLCopen XML:n muodostaessa virallisesti PLC-sovellusohjelmien siirtostandardin IEC 61131-10, saadaan tähän muutos ja PLCopen XML -tuki saadaan myös muihin ohjelmointiympäristöihin.

Lähteet

1 Karl-Heinz John, Michael Tiegelkamp. 2010. IEC 61131-3: Programming Industrial Automation Systems. Berliini: Springer-Veglag.

2 PLCopen adds independent schemes to IEC 61131-3. Verkkodokumentti. PLCopen-järjestön TC6-työryhmä. <http://www.plcopen.org/pages/tc6_xml/xml_intro/index.htm>. Luettu 16.3.2014.

3 XML. Verkkodokumentti. Wikipedia-tietosanakirja. http://fi.wikipedia.org/wiki/XML. Luettu 16.3.2014.

4 XML Formats for IEC 61131-3, PLCopen TC6 Technical Paper, version 2.01. PLCopen. 8.3.2009.

5 IEC 61131-3: a standard programming resource. Verkkodokumentti. PLCopen. <http://www.plcopen.org/pages/tc1_standards/downloads/intro_iec.pdf> Luettu 15.3.2014.

6 PLCopen XML enters a new phase as IEC 61131-10. 2014. Verkkodokumentti. PLCopen. <http://www.plcopen.org/pages/promotion/publications/downloads/press_releases/xml_nov2014.htm>. Luettu 2.3.2015.

7 TIA Portal - teollisuusautomaation ohjelmistoalusta. Verkkodokumentti. Siemens. <http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/tia_portal.php>. Luettu 18.3.2014.

8 Programming Guideline for S7-1200/S7-1500. Verkkodokumentti. Siemens. <https://support.industry.siemens.com/cs/document/81318674?lc=en-WW>. Luettu 18.3.2014

9 Export PCLopenXML. CoDeSys V3.5 PS4 Patch 2 -ohjelmiston avustustoiminto.

10 Export PCLopenXML. Verkkodokumentti. Beckhoff. <http://infosys.beckhoff.com/index_en.htm>. Luettu 20.3.2014.

11 TC6_XML_V10.xsd. KW-software. Multiprog 5.50 -ohjelmointiympäristön PLCopen XML -skeematiedosto.

PLCopen XML -esitys LD-sovellusohjelmasta

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="" productName="CODESYS" productVersion="CODESYS V3.5 SP4 Patch 2" creationDateTime="2015-03-14T19:22:08.789" />
  <contentHeader name="Railway.project" modificationDateTime="2015-03-16T19:02:08.498">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
      </fbd>
      <ld>
        <scaling x="1" y="1" />
      </ld>
      <sfc>
        <scaling x="1" y="1" />
      </sfc>
    </coordinateInfo>
    <addData>
      <data name="http://www.3s-software.com/plcopenxml/projectinformation" handleUnknown="implementation">
        <ProjectInformation />
      </data>
    </addData>
  </contentHeader>
  <types>
    <dataTypes />
    <pous>
      <pou name="POU" pouType="functionBlock">
        <interface>
          <localVars>
            <variable name="T1">
              <type>
                <derived name="TON" />
              </type>
            </variable>
            <variable name="T1_Done">
              <type>
                <BOOL />
              </type>
            </variable>
            <variable name="T1_ElapsedTime">
              <type>
                <TIME />
              </type>
            </variable>
            <variable name="C1">
              <type>
```

```
                <derived name="CTU" />
              </type>
            </variable>
            <variable name="C1_Done">
              <type>
                <BOOL />
              </type>
            </variable>
            <variable name="C1_CountValue">
              <type>
                <WORD />
              </type>
            </variable>
          </localVars>
        </interface>
        <body>
          <LD>
            <leftPowerRail localId="0">
              <position x="0" y="0" />
              <connectionPointOut formalParameter="none" />
            </leftPowerRail>
            <comment localId="1" height="0" width="0">
              <position x="0" y="0" />
              <content>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </content>
            </comment>
            <vendorElement localId="2">
              <position x="0" y="0" />
              <alternativeText>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </alternativeText>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdelementtype"
handleUnknown="implementation">
                  <ElementType
xmlns="">networktitle</ElementType>
                </data>
              </addData>
            </vendorElement>
            <contact localId="3" negated="false" storage="none"
edge="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="0" />
              </connectionPointIn>
              <connectionPointOut />
              <variable>Input1</variable>
            </contact>
            <contact localId="4" negated="true" storage="none"
edge="none">
              <position x="0" y="0" />
```

```
                <connectionPointIn>
                  <connection refLocalId="3" />
                </connectionPointIn>
                <connectionPointOut />
                <variable>Input2</variable>
              </contact>
              <coil localId="5" negated="false" storage="none">
                <position x="0" y="0" />
                <connectionPointIn>
                  <connection refLocalId="4" />
                </connectionPointIn>
                <connectionPointOut />
                <variable>Output1</variable>
              </coil>
              <comment localId="6" height="0" width="0">
                <position x="0" y="0" />
                <content>
                  <xhtml xmlns="http://www.w3.org/1999/xhtml" />
                </content>
              </comment>
              <vendorElement localId="7">
                <position x="0" y="0" />
                <alternativeText>
                  <xhtml xmlns="http://www.w3.org/1999/xhtml" />
                </alternativeText>
                <addData>
                  <data name="http://www.3s-
software.com/plcopenxml/fbdelementtype"
handleUnknown="implementation">
                    <ElementType
xmlns="">networktitle</ElementType>
                  </data>
                </addData>
              </vendorElement>
              <contact localId="9" negated="false" storage="none"
edge="none">
                <position x="0" y="0" />
                <connectionPointIn>
                  <connection refLocalId="0" />
                </connectionPointIn>
                <connectionPointOut />
                <variable>Input1</variable>
              </contact>
              <inVariable localId="10">
                <position x="0" y="0" />
                <connectionPointOut />
                <expression>T#2s</expression>
              </inVariable>
              <block localId="8" typeName="TON" instanceName="T1">
                <position x="0" y="0" />
                <inputVariables>
                  <variable formalParameter="IN">
                    <connectionPointIn>
```

```
                    <connection refLocalId="9" />
                  </connectionPointIn>
                </variable>
                <variable formalParameter="PT">
                  <connectionPointIn>
                    <connection refLocalId="10" />
                  </connectionPointIn>
                </variable>
              </inputVariables>
              <inOutVariables />
              <outputVariables>
                <variable formalParameter="Q">
                  <connectionPointOut />
                </variable>
                <variable formalParameter="ET">
                  <connectionPointOut>
                    <expression>T1_ElapsedTime</expression>
                  </connectionPointOut>
                </variable>
              </outputVariables>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                  <CallType xmlns="">functionblock</CallType>
                </data>
              </addData>
            </block>
            <coil localId="11" negated="false" storage="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="8" formalParameter="Q"
/>
              </connectionPointIn>
              <connectionPointOut />
              <variable>T1_Done</variable>
            </coil>
            <comment localId="12" height="0" width="0">
              <position x="0" y="0" />
              <content>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </content>
            </comment>
            <vendorElement localId="13">
              <position x="0" y="0" />
              <alternativeText>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </alternativeText>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdelementtype"
handleUnknown="implementation">
```

```
                    <ElementType
xmlns="">networktitle</ElementType>
                  </data>
                </addData>
              </vendorElement>
              <contact localId="15" negated="false" storage="none"
edge="none">
                <position x="0" y="0" />
                <connectionPointIn>
                  <connection refLocalId="0" />
                </connectionPointIn>
                <connectionPointOut />
                <variable>T1_Done</variable>
              </contact>
              <inVariable localId="16">
                <position x="0" y="0" />
                <connectionPointOut />
                <expression>3</expression>
              </inVariable>
              <block localId="14" typeName="CTU" in-
stanceName="C1">
                <position x="0" y="0" />
                <inputVariables>
                  <variable formalParameter="CU">
                    <connectionPointIn>
                      <connection refLocalId="15" />
                    </connectionPointIn>
                  </variable>
                  <variable formalParameter="PV">
                    <connectionPointIn>
                      <connection refLocalId="16" />
                    </connectionPointIn>
                  </variable>
                </inputVariables>
                <inOutVariables />
                <outputVariables>
                  <variable formalParameter="Q">
                    <connectionPointOut />
                  </variable>
                  <variable formalParameter="CV">
                    <connectionPointOut>
                      <expression>C1_CountValue</expression>
                    </connectionPointOut>
                  </variable>
                </outputVariables>
                <addData>
                  <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                    <CallType xmlns="">functionblock</CallType>
                  </data>
                </addData>
              </block>
```

```
            <coil localId="17" negated="false" storage="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="14" formalParameter="Q"
/>
              </connectionPointIn>
              <connectionPointOut />
              <variable>C1_Done</variable>
            </coil>
            <comment localId="18" height="0" width="0">
              <position x="0" y="0" />
              <content>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </content>
            </comment>
            <vendorElement localId="19">
              <position x="0" y="0" />
              <alternativeText>
                <xhtml xmlns="http://www.w3.org/1999/xhtml" />
              </alternativeText>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdelementtype"
handleUnknown="implementation">
                  <ElementType
xmlns="">networktitle</ElementType>
                </data>
              </addData>
            </vendorElement>
            <contact localId="20" negated="false" storage="none"
edge="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="0" />
              </connectionPointIn>
              <connectionPointOut />
              <variable>Input1</variable>
            </contact>
            <contact localId="21" negated="true" storage="none"
edge="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="20" />
              </connectionPointIn>
              <connectionPointOut />
              <variable>Input2</variable>
            </contact>
            <contact localId="22" negated="false" storage="none"
edge="none">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="0" />
              </connectionPointIn>
```

```
            <connectionPointOut />
            <variable>C1_Done</variable>
          </contact>
          <coil localId="23" negated="false" storage="none">
            <position x="0" y="0" />
            <connectionPointIn>
              <connection refLocalId="21" />
              <connection refLocalId="22" />
            </connectionPointIn>
            <connectionPointOut />
            <variable>Output2</variable>
          </coil>
          <rightPowerRail localId="2147483646">
            <position x="0" y="0" />
            <connectionPointIn />
          </rightPowerRail>
        </LD>
      </body>
      <addData />
    </pou>
  </pous>
</types>
<instances>
  <configurations />
</instances>
<addData>
  <data name="http://www.3s-
software.com/plcopenxml/projectstructure"
handleUnknown="discard">
    <ProjectStructure>
      <Object Name="POU" />
    </ProjectStructure>
  </data>
</addData>
</project>
```

PLCopen XML -esitys FBD-sovellusohjelmasta

```
<block localId="20000000002" typeName="AND">
              <position x="0" y="0" />
              <inputVariables>
                <variable formalParameter="In1" edge="rising">
                  <connectionPointIn>
                    <connection refLocalId="20000000000" />
                  </connectionPointIn>
                </variable>
                <variable formalParameter="In2" negated="true">
                  <connectionPointIn>
                    <connection refLocalId="20000000001" />
                  </connectionPointIn>
                </variable>
              </inputVariables>
              <inOutVariables />
              <outputVariables>
                <variable formalParameter="Out1">
                  <connectionPointOut />
                </variable>
              </outputVariables>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                  <CallType xmlns="">operator</CallType>
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/inputparamtypes"
handleUnknown="implementation">
                  <InputParamTypes xmlns="" />
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/outputparamtypes"
handleUnknown="implementation">
                  <OutputParamTypes
xmlns="">BOOL</OutputParamTypes>
                </data>
              </addData>
            </block>
            <inVariable localId="20000000003">
              <position x="0" y="0" />
              <connectionPointOut />
              <expression>Error</expression>
            </inVariable>
            <inVariable localId="20000000004">
              <position x="0" y="0" />
              <connectionPointOut />
              <expression>Actuation</expression>
            </inVariable>
            <inVariable localId="20000000005">
```

```
            <position x="0" y="0" />
            <connectionPointOut />
            <expression>DoorDown</expression>
          </inVariable>
          <block localId="20000000006" typeName="AND">
            <position x="0" y="0" />
            <inputVariables>
              <variable formalParameter="In1" edge="rising">
                <connectionPointIn>
                  <connection refLocalId="20000000004" />
                </connectionPointIn>
              </variable>
              <variable formalParameter="In2">
                <connectionPointIn>
                  <connection refLocalId="20000000005" />
                </connectionPointIn>
              </variable>
            </inputVariables>
            <inOutVariables />
            <outputVariables>
              <variable formalParameter="Out1">
                <connectionPointOut />
              </variable>
            </outputVariables>
            <addData>
              <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                <CallType xmlns="">operator</CallType>
              </data>
              <data name="http://www.3s-
software.com/plcopenxml/inputparamtypes"
handleUnknown="implementation">
                <InputParamTypes xmlns="" />
              </data>
              <data name="http://www.3s-
software.com/plcopenxml/outputparamtypes"
handleUnknown="implementation">
                <OutputParamTypes
xmlns="">BOOL</OutputParamTypes>
              </data>
            </addData>
          </block>
          <inVariable localId="20000000007">
            <position x="0" y="0" />
            <connectionPointOut />
            <expression>DoorClosed</expression>
          </inVariable>
          <block localId="20000000008" typeName="OR">
            <position x="0" y="0" />
            <inputVariables>
              <variable formalParameter="In1">
                <connectionPointIn>
```

```
                    <connection refLocalId="20000000003" />
                  </connectionPointIn>
                </variable>
                <variable formalParameter="In2">
                  <connectionPointIn>
                    <connection refLocalId="20000000006" formal-
Parameter="Out1" />
                  </connectionPointIn>
                </variable>
                <variable formalParameter="In3">
                  <connectionPointIn>
                    <connection refLocalId="20000000007" />
                  </connectionPointIn>
                </variable>
              </inputVariables>
              <inOutVariables />
              <outputVariables>
                <variable formalParameter="Out1">
                  <connectionPointOut />
                </variable>
              </outputVariables>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                  <CallType xmlns="">operator</CallType>
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/inputparamtypes"
handleUnknown="implementation">
                  <InputParamTypes xmlns="" />
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/outputparamtypes"
handleUnknown="implementation">
                  <OutputParamTypes
xmlns="">BOOL</OutputParamTypes>
                </data>
              </addData>
            </block>
            <block localId="20000000009" typeName="RS" in-
stanceName="RSDoorDown">
              <position x="0" y="0" />
              <inputVariables>
                <variable formalParameter="SET">
                  <connectionPointIn>
                    <connection refLocalId="20000000002" formal-
Parameter="Out1" />
                  </connectionPointIn>
                </variable>
                <variable formalParameter="RESET1">
                  <connectionPointIn>
```

```
                    <connection refLocalId="20000000008" formal-
Parameter="Out1" />
                  </connectionPointIn>
                </variable>
              </inputVariables>
              <inOutVariables />
              <outputVariables>
                <variable formalParameter="Q1">
                  <connectionPointOut />
                </variable>
              </outputVariables>
              <addData>
                <data name="http://www.3s-
software.com/plcopenxml/fbdcalltype"
handleUnknown="implementation">
                  <CallType xmlns="">functionblock</CallType>
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/inputparamtypes"
handleUnknown="implementation">
                  <InputParamTypes xmlns="">BOOL
BOOL</InputParamTypes>
                </data>
                <data name="http://www.3s-
software.com/plcopenxml/outputparamtypes"
handleUnknown="implementation">
                  <OutputParamTypes
xmlns="">BOOL</OutputParamTypes>
                </data>
              </addData>
            </block>
            <outVariable localId="20000000010">
              <position x="0" y="0" />
              <connectionPointIn>
                <connection refLocalId="20000000009" formalPa-
rameter="Q1" />
              </connectionPointIn>
              <expression>DoorDown</expression>
            </outVariable>
```