Tampereen ammattikorkeakoulu

Ohjausyksikön elinkaaren hallinta

Jani Karstinen

OPINNÄYTETYÖ
Kesäkuu 2019

Konetekniikka
Koneautomaatio

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikka
Koneautomaatio

KARSTINEN, JANI:
Ohjausyksikön elinkaaren hallinta

Opinnäytetyö 28 sivua, joista liitteitä 1 sivua
Kesäkuu 2019

Tämä opinnäytetyö tehtiin Sandvik Mining and Construction Oy:lle. Opinnäytetyössä tutkittiin yleisesti ohjausyksiköiden elinkaarta. Työn tarkoituksena oli helpottaa vanhempien plc-ohjelmien tuontia uudempaan kehitysympäristöön ja mahdollistaa toimenpiteen sujuvuus. Työn tavoitteena oli kerätä spesifikaatiot uusille massaeditointiskripteille, joiden avulla vanhemmasta kehitysympäristöstä tiedot voidaan siirtää uudempaan. Lisäksi osa työn tavoitteista oli selvittää käytössä olevien massaeditointiskriptien toiminta ja tuoda niiden avulla tarvittavat tiedot uuteen kehitysympäristöön.

Opinnäytetyössä tutkittiin uudemman ja vanhemman ohjausyksikön eroja ja yhtäläisyyksiä. Lisäksi massaeditointiskriptien toiminta tarkastettiin ja niiden käyttämiin mallipohjiin tehtiin tarvittavat muutokset. Myös vanhemman kehitysympäristön tiedot tuotiin uuteen kehitysympäristöön. Opinnäytetyön aikana saatiin kerättyä tarvittavat spesifikaatiot uusille massaeditointiskripteille.

Kehitysehdotukseksi opinnäytetyössä annettiin uudemmalla sukupolvella jo käytössä olevien massaeditointiskriptien muokkaus Epec-ohjausyksiköille toimivaksi. Lisäksi opinnäytetyössä otettiin kantaa, mille projektin vaiheille olisi hyvä tuottaa uusi massaeditointiskripti.

Asiasanat: elinkaari, codesys, epec

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Mechanical Engineering
Machine Automation

KARSTINEN, JANI:
Control Unit Life Cycle Management

Bachelor's thesis 28 pages, appendices 1 pages
June 2019

This thesis was made for Sandvik Mining and Construction Oy. This thesis inspects the life cycle of a control unit. The purpose of this thesis is to ease the importing of older PLC programs to a newer development environment and to make the procedure smooth. The objective of this thesis was to gather specifications for new mass editing scripts which help in the importing of data from an older development environment to a newer one. In addition, a part of the objective was to investigate the operation of mass editing scripts currently in use and import data to the new development environment by using them.

The differences and similarities between the older and the newer control unit were inspected. Moreover, the operation of mass editing scripts was tested and the templates used by them were edited. Also the data from the older development environment was imported to the newer one. During this thesis the specifications for new mass editing scripts were gathered.

As a further development proposal, it is suggested that the mass editing scripts currently in use should be modified to be functional with Epec control units. In addition, producing new mass editing scripts for certain phases of the project is also proposed.

Key words: life cycle, codesys, epec

SISÄLLYS

LYHENTEET JA TERMIT (valitse jompikumpi)

CAN	controller area network, automaatioväylä
COB-ID	communication object identifier, laitteen ID väylällä
CoDeSyS	controller development system, ohjelmointiympäristö
EXP	export-file, ohjelmaan tuotava tiedostomuoto
PDO	process data object, can-väylän viesti
PLC	programmable logic controller, logiikka
PLM	product lifecycle management, tuotteen elinkaaren hallinta
RPDO	receive process data object, vastaanotettava PDO
SVN	subversion (version control system), versionhallinta
TPDO	transmit process data object, lähetettävä PDO
XDC	XDC-file, XML-muodossa oleva kirjoitinkuvaus
XML	extensible markup language, rakenteellinen kuvauskieli

1 JOHDANTO

Opinnäytetyön toimeksiantaja on Sandvik Mining and Construction Oy, joka on osa ruotsalaista metalli- ja kaivosalan teollisuuskonsernia Sandvik AB. Yritys valmistaa Tampereen Myllypurossa muun muassa kaivos- ja louhintakoneita. Yrityksen valmistamia vanhemman sukupolven laitteita ja niiden toimintoja ohjataan tällä hetkellä Epec 2024 -ohjausykisköillä.

Opinnäytetyön tarve syntyi, koska aiemmin käytetty Epec 2024 -ohjausyksikkö on poistumassa toimittajan tuotevalikoimasta, jolloin yritys ei voi enää toimittaa kyseistä ohjausyksikköä asiakkailleen. Tämä johtaa tilanteeseen, jossa Sandvikin ostamien Epec 2024 -ohjausyksiköiden loppuessa yritys ei voi enää toimittaa kyseistä ohjausyksikköä varaosana. Ohjausyksikkö tullaan vaihtamaan uudempaan Epec 3724 -ohjausyksikköön. Ohjausyksikön vaihto aiheuttaa PLC-ohjelmointiin käytetyn CoDeSyS-ohjelmointiympäristön version muutoksen. CoDeSyS-version muutoksesta aiheutuu PLC-ohjelmaan muutoksia esimerkiksi CAN-väylään ja I/O:iden määritykseen.

Työn tarkoituksena on selvittää tarvittavat spesifikaatiot mahdollisille massaeditointiskripteille, selvittää tämänhetkisien skriptien ja mallipohjien toiminta sekä tuoda CAN-rajapinta, ohjausyksikön sisään- ja ulostulot ja laitteen hälytykset PLC-ohjelmaan.

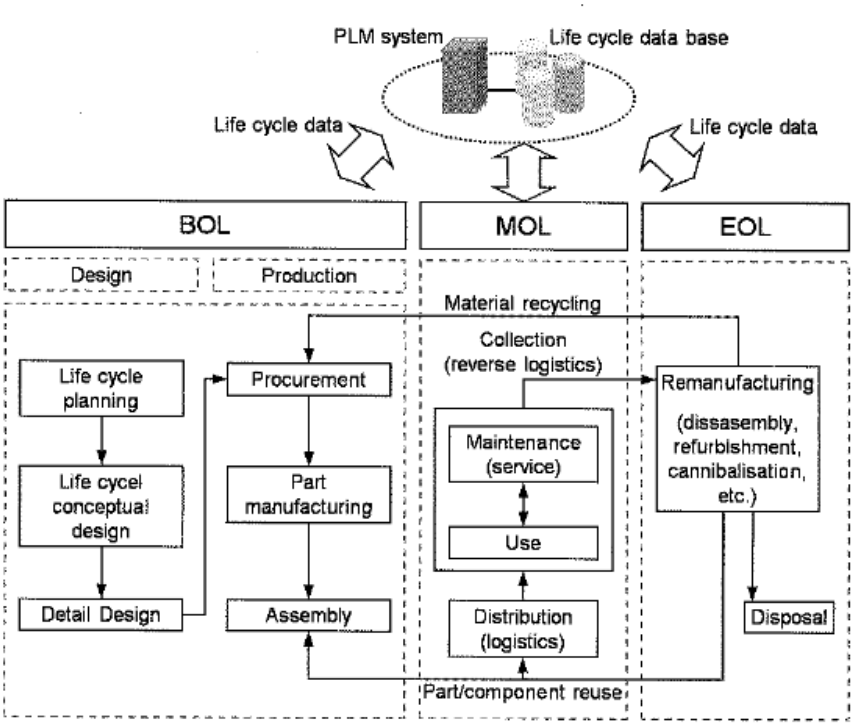
Työssä tutustutaan yleisesti tuotteiden elinkaareen ja sen hallintaan. Sen hallinnan lisäksi opinnäytetyössä käydään läpi yleisesti tuotteiden elinkaaren hallinnan tärkeyttä ja siitä saatavia etuja.

2 ELINKAARI

2.1 Tuotteen elinkaaren hallinta

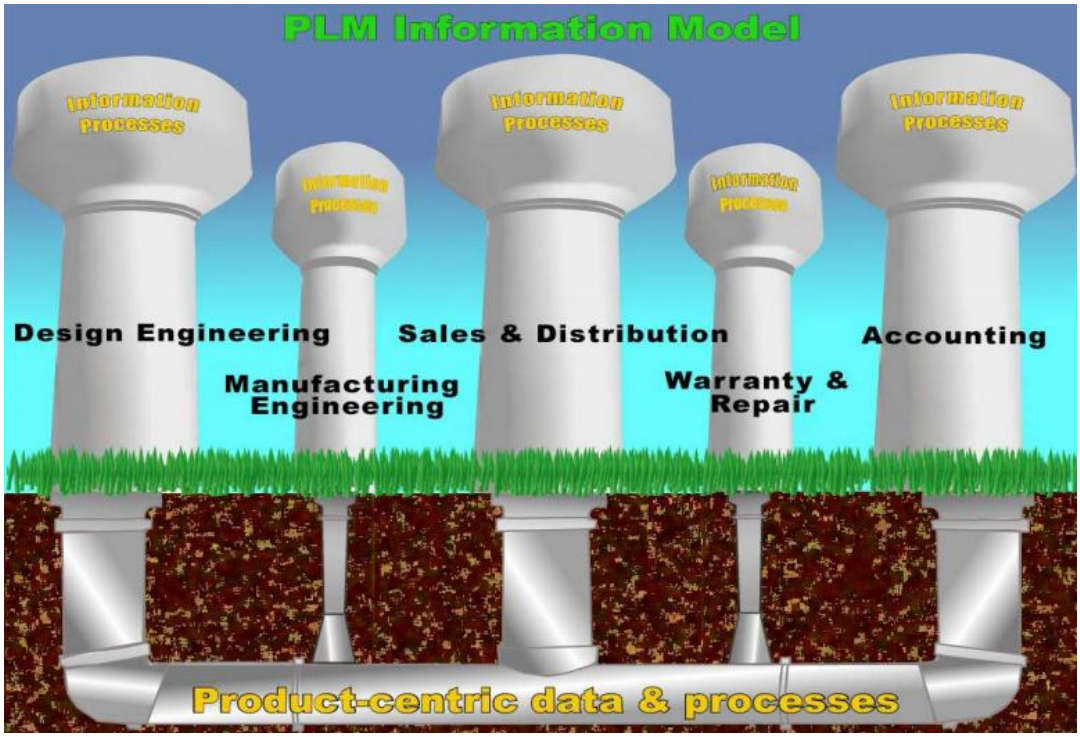
Koventuva maailmanlaajuinen kilpailu tuottaa yrityksille paineita ja vaatii niitä vaihtamaan prosesseja ja toimimaan mahdollisimman tehokkaasti. Useat tuotteet tehdään yhä enemmän asiakkaan vaatimusten mukaan, mikä lisää eroja tuotteiden malleissa. Painetta muutoksiin tuovat esimerkiksi lyhyemmät toimitusajat, tiukemmat laatuvaatimukset ja lainsäädännön tiukkeneminen. Tämän vuoksi on tärkeää toimia tehokkaasti ja välttää tarpeetonta tehotonta työtä. (Saaksvuori & Immonen 2008).

Elinkaari alkaa jo suunnittelusta ja päättyy, kun tuote hävitetään tai siitä luovutaan kokonaan. Tuotteen elinkaaren hallinta, yleisesti PLM eli product life cycle management, on saanut alkunsa lean-ajattelutavasta, joka on systemaattinen tapa vähentää tuottamattoman työn määrää jokaisessa työn vaiheessa menettämättä tuottavuutta. Kuten nimi viittaa, PLM:llä hallinnoidaan tuotetietoja tuotteen koko elinkaaren ajan, alustavasta suunnitelmasta tuotteen käytöstä poistumiseen asti (kuvio 1). Toisin kuin leanissa, PLM:lla eliminoidaan turhuutta ja tehottomuutta kaikissa tuotteen elinkaaren vaiheissa. PLM keskittyy käyttämään tietoa ja leikkaamaan sillä tehottomuutta. Tuotteen elinkaaren hallinta on niin sanotusti vienyt leanin seuraavalle tasolle. (Grieves 2006.)



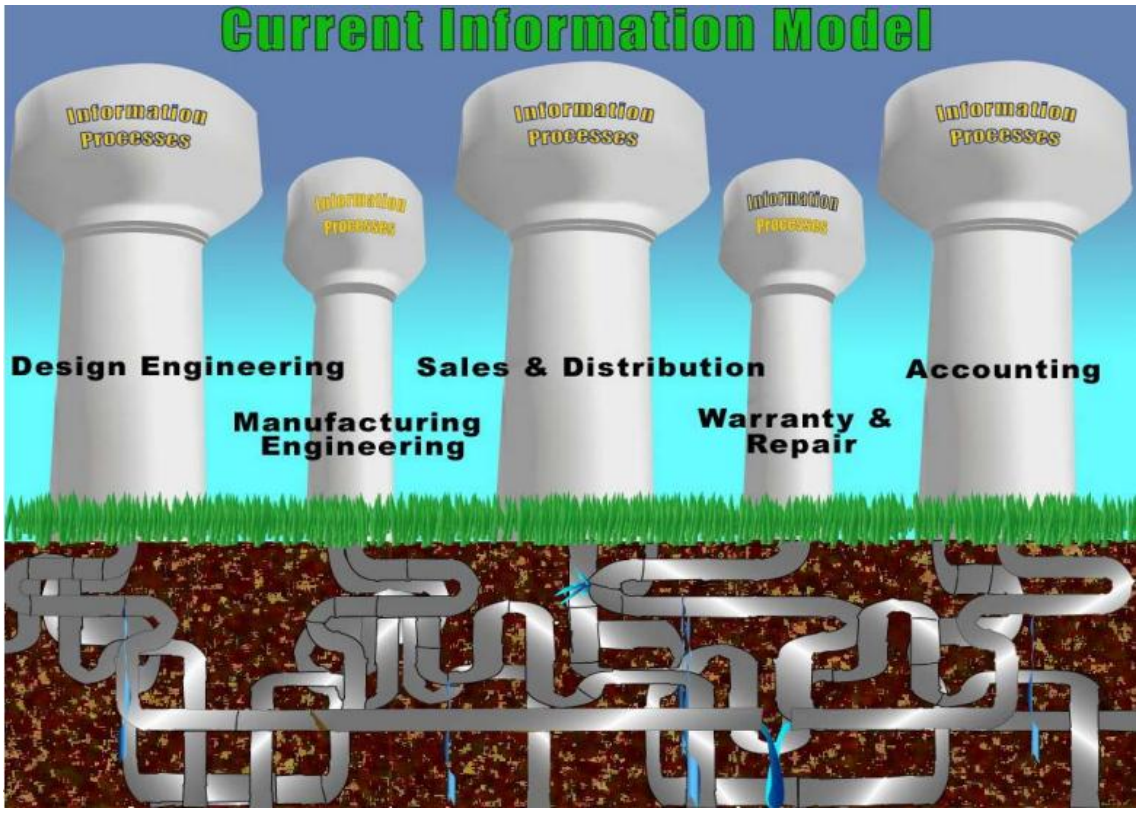
KUVIO 1. Tuotteen elinkaari (Niemann, Tichkiewitch & Westkämper 2009)

PLM:n tiedotusmalli on alusta, jonka tarkoituksena on tunnistaa ja yhdistää kaikki yrityksen toiminnalliset alueet tuotekeskeisiin tietoihin ja prosesseihin. Näin saadaan kehitettyä koko organisaation laajuinen näkymä, jonka avulla tiedot voidaan helposti jakaa organisaation eri prosesseissa (kuva 1). (Grieves 2006.)



KUVA 1. Tuotteen elinkaaren hallinnan tiedotusmalli (Grieves 2006)

Ennen tuotteen elinkaaren hallinan tiedotusmallia on tuotetietojen keräys ollut hyvin epäselvää ja tehotonta (kuva 2). Kuvassa nähdään hyvin, että tiedot eivät aina kulje tarvittaville osille ja niitä tarvittaessa etsimiseen kuluu aikaa. (Grieves 2006.)



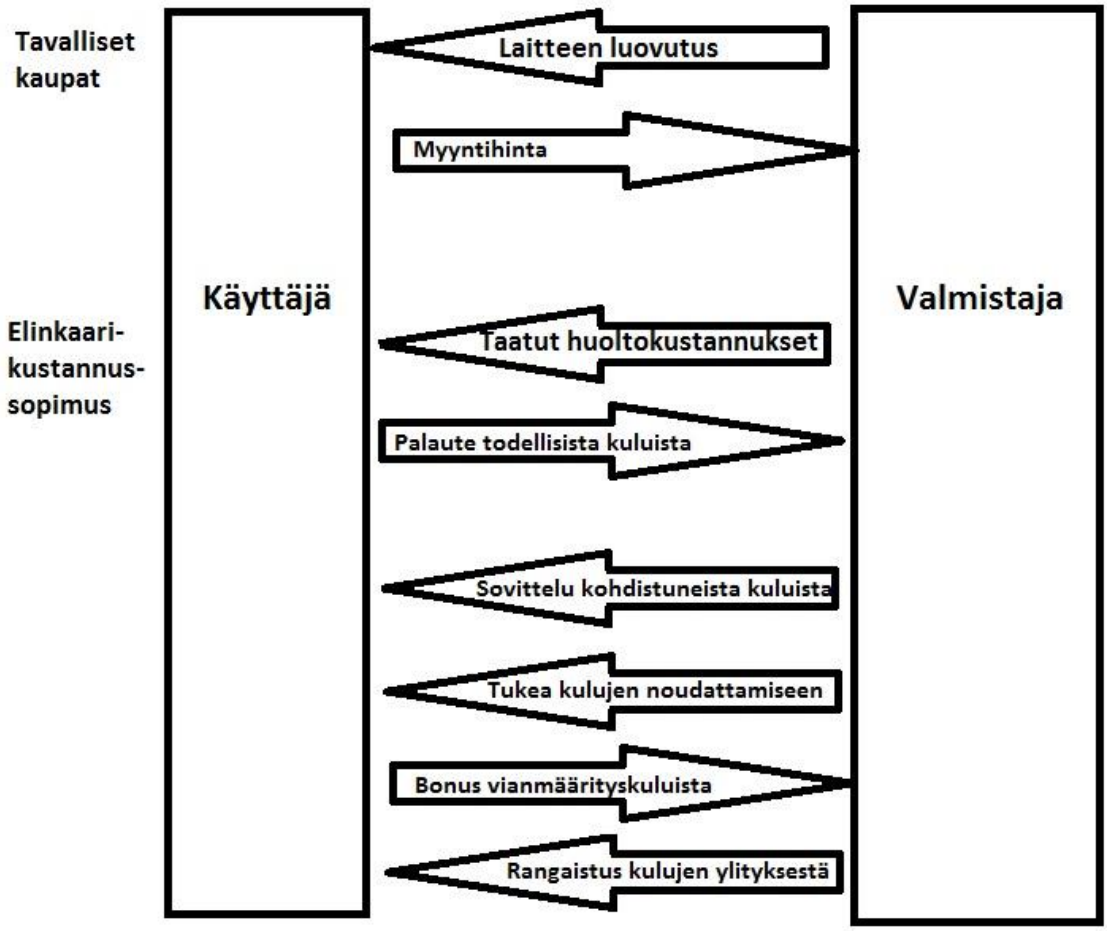
KUVA 2. PLM-tiedotusmallia edeltävä tiedotusmalli (Grieves 2006)

2.2 Tuotteen elinkaarikustannukset

Tuotteen elinkaarikustannuksilla kuvataan elinkaaren aikana tulevia kuluja. Kuluihin lasketaan esimerkiksi suunnittelun aikaisia hankinta- ja markkinointikuluja, tuotteen valmistuksen aiheuttamia valmistuskuluja sekä tuotteen elinkaaren lopussa tuotteen hävitykseen liittyviä kuluja.

Koska elinkaaren kustannukset ovat oleellinen osa teollista toimintatapaa, on tärkeää pitää kirjaa, analysoida ja optimoida kyseisiä kustannuksia. Tämän avulla voidaan saada merkittäviä taloudellisia etuja tuotteen elinkaaren aikana. (Niemann, Tichkiewitch & Westkämper 2009)

Yhä useammin asiakkaat vaativat varmuutta siitä, millaisia kustannuksia heidän hankkimiinsa tuotteisiin liittyy niiden elinkaaren aikana. Varmuutta voidaan taata tekemällä elinkaaren kustannuksista sopimus, jossa tuotteen myyjä takaa tuotteelle esimerkiksi teknistä tukea ongelmiin, varaosia tai tuotteen korjausta (kuvio 2). (Niemann, Tichkiewitch & Westkämper 2009)



KUVIO 2. Esimerkki elinkaarikustannussopimuksesta (Niemann, Tichkiewitch & Weskämper 2009, muokattu)

3 ALUSTAVAT TOIMENPITEET

Tämän opinnäytetyön lähtöpisteenä oli vanhemman sukupolven porauslaitteiden käyttämän Epec 2024 -ohjausyksikön elinkaaren päättyminen. Ohjausyksikön elinkaaren päättyminen aiheuttaa tarpeen ohjausyksikön päivittämiselle. Ohjausyksiköiden päivittäminen uudempaan Epec 3724 -ohjausyksikköön mahdollistaa, että porauslaitteiden elinkaari ei pääty eikä laitteiden ohjauksille tarvitse tehdä niin suuria muutoksia.

Työssä käsitellään XML-tiedostoja, jotka mahdollistavat laajojen tietomassojen selkeämmän käsittelyn. Muokattavien tiedostojen lisäksi työssä käytetään Python-pohjaisia massaeditointiskriptejä, joita ei ole tarkoitus työn aikana muokata. Tiedostoja voidaan käsitellä esimerkiksi Notepad++:n avulla.

XML-tiedostojen lisäksi työssä käsitellään plc-ohjelmia CoDeSyS-ohjelmointiympäristön avulla. Koska työssä käsitellään vanhempia Epec 2024- ja hieman uudempia Epec 3724 -ohjausyksiköitä, tarvitaan PLC-ohjelmille kaksi eri CoDeSyS-versiota. Vanhempia Epec 2024 -ohjausyksiköitä voidaan ohjelmoida CoDeSyS 2.1 -versiolla ja uudempia 3724 -ohjausyksiköitä 2.3-versiolla.

3.1 Epec

Ohjausyksiköt 2024 ja 3724 (kuva 3) ovat IP-luokitukseltaan IP67. IP-luokitus on sähkölaitteiden ja laitekoteloiden tiiviydestä kertova luokitus. IP67 ilmaisee laitteen olevan täysin pölytiivis (6) ja kestävän hetkellisen upotuksen veteen (7). Tämä tarkoittaa sitä, että laitetta voidaan käyttää samanlaisissa olosuhteissa kuin aiemminkin.

KUVA 3. Epec 3724 -ohjausyksikkö (Epec Oy: Epec 3724 Control Unit Technical Document)

Uudemmassa että vanhemmassa ohjausyksikössä on yhtä monta sisään- ja ulostuloa (taulukko 1), yhteensä 52 sisään- ja ulostulopinniä. Lisäksi ohjausyksiköiden langoitukset ovat sisään- ja ulostulojen kannalta samoissa paikoissa. Digitaalisten ja analogisten sisään- ja ulostulojen sijainnit nähdään Epec 2024 -ohjausyksikön IEC-mapista (liite 1).

Outputs	24 x	PWM/DO/DI (sourcing, up to 3 A, PWM frequency by application)
	4 x	DI/DO (sinking)
Inputs	4 x	DI (pull-down to GND)
	4 x	FB/AI (current measuring feedback, 0-1 A)
	8 x	AI/DI (0-5 V / 0-22 mA selection by application)
	8 x	DI/PI (pull-down to GND)

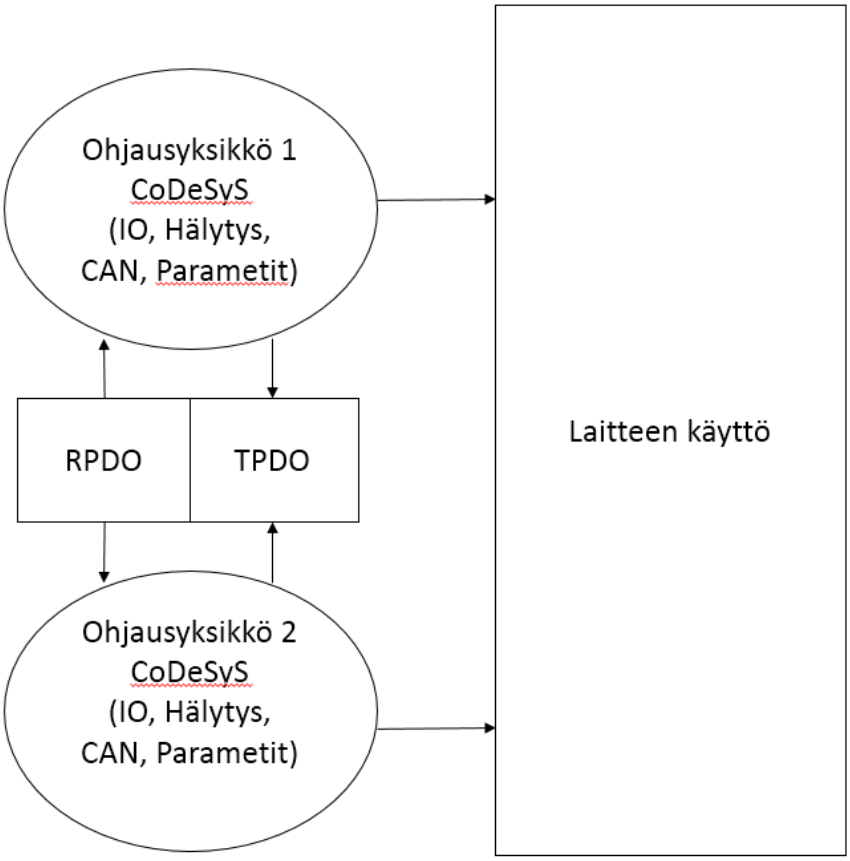
Taulukko 1. Epec 3724:n sisään- ja ulostulojen käyttömahdollisuudet (Epec Oy: Epec 3724 Control Unit Technical Document)

Uuden version on pinnien XM3.2 ja XM3.10 maksimivirta, joka on 270 mA. Vanhemmassa ohjausyksikössä kyseisten pinnien maksimivirta on 250 mA. Lisäksi

verrattaessa PWM/DI/DO_Type051- ja AI/DI_Type064_3 -pinnejä vastaaviin Epec 2024 -pinneihin on sisääntuloresistansseissa eroja.

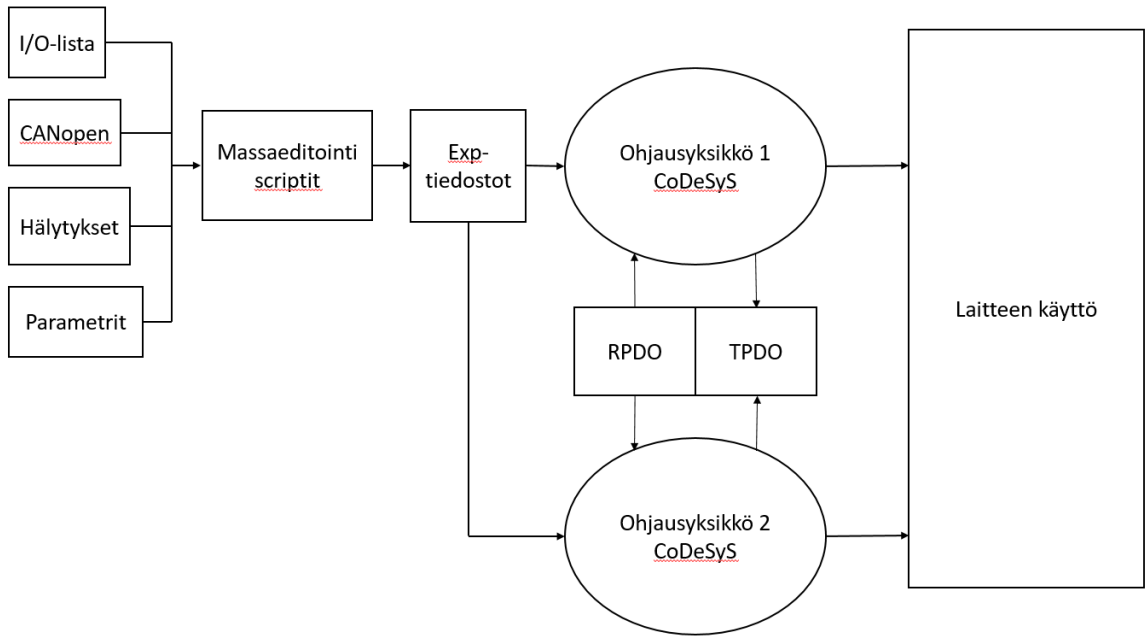
3.2 Konfiguraation rakenne

Aiemmin ainoastaan ohjausyksiköiden PLC-ohjelmat ovat olleet osa konfiguraation rakennetta (kuvio 3). Tällöin can-väylän tiedot, I/O ja muut tiedot on muutettu suoraan ohjelmassa. Kuviosta 3 nähdään myös, että ohjausyksiköt keskustelevat muiden ohjausyksiköiden kanssa lähettämällä PDO:ita, joita toinen ohjausyksikkö ottaa vastaan.



KUVIO 3. Vanha konfiguraatiorakenne

Uudessa konfiguraatiorakenteessa I/O-lista yms. muutetaan massaeditointiskriptien avulla export-tiedostoiksi (kuvio 4). Export-tiedostot voidaan ladata suoraan plc-ohjelmaan. Tämä tapa helpottaa ja nopeuttaa muutoksien tekoa esimerkiksi I/O- tai CAN-puolella.



KUVIO 4. Uusi konfiguraatiorakenne

Hälytykset, I/O-listat ja muut projekteissa tarvittavat tiedostot on lajiteltu kansioihin eri puolelle projektia. Massaeditointiskripteissä on määritelty, mistä tiedostot löytyvät. Skriptien, mallipohjien ja täytettyjen tietojen avulla voidaan luoda export-tiedostot, jotka sen jälkeen ladataan ohjausyksiköiden plc-ohjelmiin.

3.2.1 Devpacks

Devpacks-kansio sisältää projektiin tarvittavia skriptejä ja skriptien käyttämiä tiedostojen mallipohjia. Kansio ladataan SVN:stä (Subversion), joka on versionhallintaohjelma. Versionhallintaohjelma mahdollistaa ajantasalla olevien tiedostojen ja skriptien käytön. Skriptit ovat Python-kielellä tehtyjä ohjelmia, jotka mahdollistavat tehokkaamman ja selkeämmän tavan tehdä projektin lopullinen CoDeSyS-ohjelma.

IOConfigTemplate on tekstipohjainen XML-tiedosto, jota massaeditointiskriptit käyttävät DM-tiedostoa tehtäessä. Tiedostoon on merkitty eri IO-tyypit, niiden hälytykset ja miten eri IO:t muodostuvat XDC-tiedostoon.

3.2.2 Config

Config-kansio sisältää tiedostot ohjausyksikön PDO:ista, hälytyksistä, laitteen ohjausyksiköiden IO-listat ja tiedot laitteen väylistä. PDO:lla (Process Data Object) on tiedon välittämistä noodilta toiselle. Noodilla tarkoitetaan väylässä olevia laitteita. PDO:ita on lähettäviä ja vastaanottavia.

Machine Type Description -tiedosto on XML-tiedosto, johon lisätään laitteella käytettävät väylät ja kyseisten väylien tiedot, kuten siirtonopeudet ja noodien tunnisteet. Tiedostossa ilmoitetaan myös mikä väylä toimii masterina ja mitkä slaveina. Väylätekniikassa masterilla kuvataan noodia, joka lähettää tai pyytää dataa slaveilta. Master voi myös vaihtaa slaven tilaa, kuten käynnistys, pysäytys ja nollaus.

IO-lista on Excel-muodossa oleva lista, johon kootaan tarvittavat tiedot niille tarkoitetuille paikoille. Listaan tulee täyttää toiminto, sensori tai käyttö, sen ohjausyksiköllä oleva liittimen ja pinnin paikka, I/O-tyyppi, sisään- ja ulostulon arvo ja laitteen mahdollinen toiminta-alue. I/O-tyypillä tarkoitetaan onko ohjausyksikön kyseisellä pinnillä käytössä esimerkiksi maattava digitaalinen ulostulo (DOLowSideSwitching). Sisääntulon toiminta-alueella kuvataan CoDeSyS-projektissa kyseisen toiminnon aluetta. Jokaiselle laitteella käytettävälle ohjausyksikölle luodaan oma IO-lista omille lehdilleen samaan Excel-tiedostoon.

Ohjausyksikön CANopen-tiedot kootaan omaan XDC-pohjaiseen tiedostoon, joka nimetään ohjausyksikön ohjelman toiminnasta riippuen. Tiedostoon kootaan mm. ohjausyksikön lähettävät ja vastaanottavat PDO:t.

3.2.3 MC

MC-kansio sisältää laitteen ohjelmointiin liittyvät PLC-ohjelmat ja niihin tarvittavat export-tiedostot. PLC-ohjelmiin tarvittavat export-tiedostot on saatu skriptien avulla, kun laitteen speksit on lisätty tarvittaviin tiedostoihin.

CoDeSyS-projekti aloitetaan valmiiseen mallipohjaan, joka sisältää Sandvikilla käytettävän SICA-alustan. Alusta toimii projektin pohjana ja projektiin ohjelmoidaan laitteelta vaaditut toiminnot.

DataManager-tiedosto (DM) on tekstipohjainen XML-tiedosto, joka sisältää listan ohjausyksikön inputeista ja outputeista. Tiedosto voidaan massaeditointiskriptin avulla kääntää EXP-muotoon. Muunto tehdään, jotta IO-lista voidaan tuoda CoDeSyS-ohjelmaan mahdollisimman tehokkaasti.

4 EXPORT-TIEDOSTOT

4.1 Kehitysympäristö

Projekti aloitettiin asentamalla kehitysympäristö ja tarvittavat devpackit. Kehitysympäristö on joukko ohjelmia, jonka avulla ohjelmistoa suunnitellaan ja tehdään. Kehitysympäristöön voidaan integroida mm. versionhallintatuki ja profilointi.

Kehitysympäristössä on yleensä kolme tasoa, jotka voidaan jakaa kehitykseen, testaukseen ja tuotantoon. Kehitystasolla luodaan koodia ja varmistetaan, että applikaatio toimii kyseisellä koodilla. Applikaation toiminnan varmistamisen jälkeen ohjelmoija siirtää koodin testaustasolle. Testaustasolla koodin toiminta varmistetaan, luotettavuus testataan ja varmistetaan, että se ei vikaannu tuotantotasolla. Kun applikaatio saa testaustasolla hyväksynnän, se voidaan siirtää tuotantotasolle. (Technopedia: Development Environment)

4.2 DataManager -export

Kehitysympäristön asentamisen jälkeen voitiin täyttää MachineTypeDescription -tiedostoon laitteen väylät sekä niiden masterit ja slavet (kuva 4).

```
14      <!--#
15          #  BackBone bus
16          #                 -->
17
18      <node xsi:type="canopenNodeType" key="TRI1" presence="always" name="HMI Master">
19      <!-- HM - HMI Master EPEC -->
20          <alternativeNodeType nodeType="EPEC_TCM3724" binaryRelationKey="hm_bin" xddFile="TRI1.XDC">
21              <nodeConfig xsi:type="xdc" filename="backbone/TRI1.XDC"/>
22          </alternativeNodeType>
23          <nodeId buskey="BackBone" type="normal" value="0x14" prefix="TRI1" port="CANA" mconFlags="DisableSDOForPDOSig"/>
24          <nodeId buskey="BackBone" type="debug" value="0x54" prefix="TRI1" port="CANA" mconFlags="DisableSDOForPDOSig"/>
25          <nodeId buskey="BackBone" type="sdo_gw" value="0x74" prefix="TRI1" mconFlags="DisableSDOForPDOSig"/>
26      </node>
27  </nodeList>
28
29  <busList>
30      <bus xsi:type="canopenBusType" key="BackBone" baudrate="500k" backbone="true" configFlags="">
31          <nmtMaster nodeRef="TRI1"/>
32          <gateway nodeRef="TRI1"/>
33      </bus>
34  </busList>
```

KUVA 4. Ensimmäisen ohjausyksikön TRI1 väylätietojen lisäys MTD

Projektin rakenteen luonnin jälkeen laitteelle aiemmin tehty IO-lista tulee päivittää skriptien vaatimukset täyttäväksi. IO-lista tehtiin uuteen muotoon ja tarvittavat tiedot lisättiin. Tarvittavia lisäyksiä olivat pinnin toiminto, ohjausyksikön pinnien I/O tyyppi, analogisten sisääntulojen toiminta-alueet, sisään- ja ulostuloissa olevat jännitteet ja virrat.

IO-listan muokkausten lisäksi devpacks-kansiossa olevan Epec-ohjausyksikön IOConfigTemplate-pohjaan lisättiin DOLowSideSwitching ja AIFeedBack-IO-pinnit ja niiden hälytykset (kuva 5). Lisäys tehtiin IOConfigTemplaten puutteellisuuden vuoksi, eikä välivaihetta tarvitse tehdä enää uudestaan esimerkiksi uusia moduuleita lisättäessä.

```
250     <IOPin type="DOLowSideSwitching">
251     <Template type="DM">
252         <output-signal sourcetype="bool" source="S_OUT_@name_camelcase@" destinationtype="DOLowSideSwitching" destination="@connector@_@pin@_channel"/>
253     </Template>
254
255     <Template type="AlarmConfig"/>
256
257     <Template type="XDC">
258         <CANopenObject index="*" name="IO_DO" objectType="8" accessType="rw">
259             <CANopenSubObject subIndex="*" name="DO_@function@_bits" objectType="7" dataType="06" accessType="rw" defaultValue="0" SICAMcIecBitVariable="true">
260                 <SICAMcIecBitSignal bit="*" name="DO_@name_camelcase@"/>
261             </CANopenSubObject>
262         </CANopenObject>
263     </Template>
264
265     <Template type="IOView">
266         <Group name="@connector@">
267             <Pin name="@pin@" type="DO" uiDescription="@Name@ (@ioname@)" inputFunction="" valueMin="0" valueMax="1"/>
268         </Group>
269     </Template>
270
271 </IOPin>
```

KUVA 5. DOLowSideSwitching lisäys IOConfigiin

Tämän jälkeen IO-listasta voitiin tehdä skriptien avulla kullekin ohjausyksikölle DM-tiedosto, joka myös generoitiin export-tiedostoksi plc-ohjelmaa varten. Generoitu export-tiedosto ladattiin kehitysympäristöön, jonka jälkeen plc-ohjelmaan oli saatu lisättyä ohjausyksikön sisään- ja ulostulot (kuva 6).

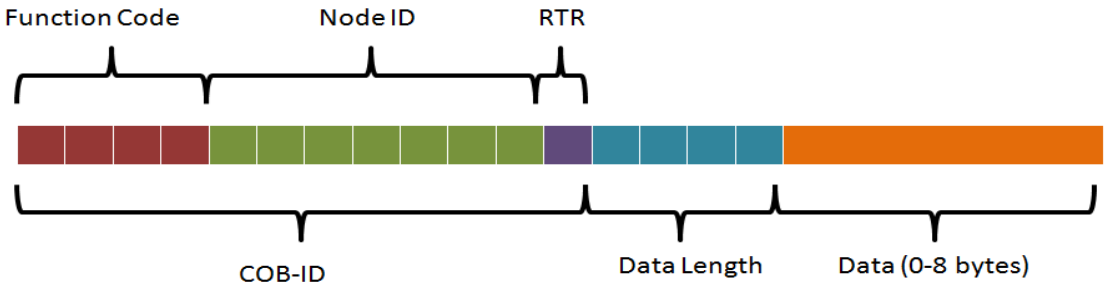
```
0001 PROGRAM GENERATED_DMin
0002 VAR
0003     initialized: BOOL := FALSE;
0004     canEnableAlarms: BOOL := TRUE;
0005     GENERATED_initializationSucceeded: BOOL := FALSE;
0006
0007     DataManager_input_XM1_19_channel_S_IN_compressorAirTemperatureWarning100C: IOControl_DI_HighSide;
0008     DataManager_input_XM1_20_channel_S_IN_compressorAirTemperature115C: IOControl_DI_HighSide;
```

KUVA 6. Esimerkki sisääntuloista plc-ohjelmassa

4.3 Generoitu CANOpen -export

Ohjausyksikön vastaanottamat ja lähettämät PDO:t koottiin omaan XDC-tiedostoon. Jokaiselle PDO:lle lisättiin konfigurointiasetuksiin COB-ID, lähetystyyppi ja rajoitusaika (NationalInstruments: The Basics of CANopen). COB-ID koostuu lähettäjän node ID:stä ja tiedosta, kuinka mones TPDO on. (CiA: CiA 301 version 4.2.0, CANopen application layer and communication profile).

CAN-kehys muodostuu mm. laitteen COB-ID:stä ja datasta. PDO-lähetyksen todellinen datan pituus on 0-8 byteä, joka on 64 bittiä tai 40h (kuvio 5).



KUVIO 5. CAN-kehyksen rakenne (NationalInstruments: The Basics of CANopen)

PDO:iden konfiguroinnin lisäksi niille luotiin signaalien mappaus. Signaalien mappauksessa PDO:lle asetetaan perusarvo, joka on esimerkiksi muodossa 0x30160110 (kuva 7).

```
<!-- CANA fifth TPDO CAN ID = secondary Node id + 180 + 20h -->
<CANopenObject index="1804" name="TPDO4_CommunicationParameters" objectType="9" dataType="20">
    <!-- subindex parameters Identified by name in export -->
    <CANopenSubObject subIndex="01" objectType="7" dataType="07" accessType="xo" defaultValue="0x1D9" name="TPDO4_COB_ID"/>
    <CANopenSubObject subIndex="02" objectType="7" dataType="05" accessType="xo" defaultValue="0xFF" name="TPDO4_TransmissionType"/>
    <CANopenSubObject subIndex="03" objectType="7" dataType="06" accessType="xo" defaultValue="800" name="TPDO4_InhibitTime"/>
    <!-- value in 100us -->
    <CANopenSubObject subIndex="04" objectType="7" dataType="05" accessType="xo" defaultValue="0" name="TPDO4_Reserved"/>
    <CANopenSubObject subIndex="05" objectType="7" dataType="06" accessType="xo" defaultValue="200" name="TPDO4_EventTimer"/>
    <!-- value in ms -->
</CANopenObject>
<CANopenObject index="1A04" name="TPDO4_Mapping" objectType="9" dataType="21" accessType="xo">
    <CANopenSubObject subIndex="01" objectType="7" dataType="07" accessType="xo" defaultValue="0x10160120" name="TPDO4_Signal_1" denotation="CANO_EpecBeat"/>
    <CANopenSubObject subIndex="02" objectType="7" dataType="07" accessType="xo" defaultValue="0x50120108" name="TPDO4_Signal_2" denotation="CANO_CommonEnableForOutputs"/>
    <CANopenSubObject subIndex="03" objectType="7" dataType="07" accessType="xo" defaultValue="0x30160110" name="TPDO4_Signal_3" denotation="CANO_SequenceAndFrameMovement"/>
    <CANopenSubObject subIndex="04" objectType="7" dataType="07" accessType="xo" defaultValue="0x30160208" name="TPDO4_Signal_4" denotation="CANO_DrillingModeCageAndFeedPressure"/>
</CANopenObject>
```

KUVA 7. Esimerkki TPDO:n kommunikointiparametreista ja mappauksesta

Luku voidaan jakaa osiin, joista ensimmäinen ”3016” viittaa signaalin indexiin tiedostossa, ”01” sub indexiin 01 ja ”10” kertoo bittien määrän heksalukuna. PDO:n

signaali on siis tällöin mapattuna [3016sub01] (kuva 8) ja viestin pituus on 16 bittiä. (CANopenSolutions: CANopen Basics - Process Data Exchange).

```
970   <!-- TPDO -->
971   <CANopenObject index="3016" name="TPDO4" objectType="7" dataType="07" accessType="rw">
972       <CANopenSubObject subIndex="01" objectType="7" dataType="06" accessType="rw" name="TPDO4_Sequence">
973           <SICAMcIecBitSignal bit="0" name="EndAutoSequence_HMI1"/>
974           <SICAMcIecBitSignal bit="1" name="EndSemiAutoSequence_HMI1"/>
975           <SICAMcIecBitSignal bit="2" name="FrameMovementRight_HMI1"/>
976           <SICAMcIecBitSignal bit="3" name="FrameMovementLeft_HMI1"/>
977           <SICAMcIecBitSignal bit="4" name="SemiAutomaticDrillingRotationCW_HMI1"/>
978           <SICAMcIecBitSignal bit="5" name="SetAutoSequenceLeftLimit"/>
979           <SICAMcIecBitSignal bit="6" name="SetAutoSequenceRightLimit"/>
980           <SICAMcIecBitSignal bit="7" name="AutoSequenceStartup_HMI1"/>
981           <SICAMcIecBitSignal bit="8" name="SemiAutoSequenceStartup_HMI1"/>
982           <SICAMcIecBitSignal bit="9" name="KompressorAirTemperatureAlarm_POWER"/>
983           <SICAMcIecBitSignal bit="10" name="Drill4PowerRequestAck_POWER"/>
984       </CANopenSubObject>
985       <CANopenSubObject subIndex="02" objectType="7" dataType="05" accessType="rw" name="TPDO4_Misc">
986           <SICAMcIecBitSignal bit="0" name="MinFeedPressure_HMI1"/>
987           <SICAMcIecBitSignal bit="1" name="MaxFeedPressure_HMI1"/>
988           <SICAMcIecBitSignal bit="2" name="SafetyCageClosed"/>
989           <SICAMcIecBitSignal bit="3" name="SafetyCageOptionActive"/>
990           <SICAMcIecBitSignal bit="4" name="NRR_OptionActive"/>
991           <SICAMcIecBitSignal bit="5" name="LasersOn_HMI1"/>
992           <SICAMcIecBitSignal bit="6" name="ManualDrillingMode"/>
993           <SICAMcIecBitSignal bit="7" name="AutoDrillingMode"/>
994       </CANopenSubObject>
995       <CANopenSubObject subIndex="03" objectType="7" dataType="05" accessType="rw" name="TPDO5">
```

KUVA 8. PDO-signaalien sisällön lisäys XDC-tiedostoon

Lopulta XDC-tiedostosta saatiin generoitua export-tiedosto CoDeSyS-ohjelmaan. Tällöin XDC-tiedostoon lisätyt CANopen-tiedot muuttuivat PLC-ohjelman muotoon (kuva 9).

```
0001  initOk := C_PDO_MEMALLOC(bus := 1, numOfRpdos := 6, numOfTpdos := 16);
0002  RegisterTPDO(busId:=1, cobId:=16#199,transmissionType:=16#FF,inhibitTime:=6,eventTimer:=20,mapping1:=16#50060110,mapping2:=16#50060210,mapping3:=16#50060310);
0003  RegisterTPDO(busId:=1, cobId:=16#1D9,transmissionType:=16#FF,inhibitTime:=80,eventTimer:=200,mapping1:=16#10160120,mapping2:=16#50120108,mapping3:=16#30160110,mapping4:=16#30160208);
0004  RegisterTPDO(busId:=1, cobId:=16#219,transmissionType:=16#FF,inhibitTime:=80,eventTimer:=200,mapping1:=16#30170208);
```

KUVA 9. Esimerkki TPDO:sta CoDeSyS-ohjelmassa, TPDO4 rivillä 3

4.4 Hälytykset

Ohjausyksikön toimintojen hälytykset esimerkiksi oikosuluille tai jännitteen puuttumiselle muodostuvat ohjausyksikön IOConfigin avulla erilliseen XML-tiedostoon. Kyseiset hälytykset tulee tämän jälkeen lisätä alarmconf-tiedostoon, että hälytykset muodostuvat export-tiedostoon. Lisättäessä hälytyksiä tiedostoon tulee muistaa mainita OrigComponent IO-hälytyksille. Hälytysten lisäys voidaan suorittaa kopioimalla ohjausyksikön IO-hälytykset alarmconf-tiedostoon.

5 PLC-OHJELMA

Epec 3724 -ohjausyksikön RAM-muisti on noin 1024 kilobittiä ja Sandvikilla ohjelmoinnissa käytettävä SICA-alusta vie muistista noin 5,5% (kuva 10).

KUVA 10. Projektin SICA-alustan jälleenrakennus

Tällä hetkellä käytössä olevilla uudemman sukupolven moduuleilla on toimiva massaeditointiskripti, joka tuottaa DMi_IO- ja DMo_IO-objektit PLC-ohjelmaan. Ohjelma luodaan, jotta ohjausyksikön sisään- ja ulostulot saadaan koottua applikaatiopuolella helpommin käytettäviksi. Kyseisiin objekteihin kirjataan myös PDO:issa tulevat ja lähtevät signaalit, joita ohjausyksikkö käyttää.

Epecin ohjausyksiköiltä puuttuu vielä tällainen skripti, jonka avulla sisään- ja ulostulot sekä PDO-signaalit saataisiin helposti käyttöön plc-ohjelmoinnissa. Tällä hetkellä käytössä olevilla uudemman sukupolven moduuleilla tällainen skripti jo on ja sitä voi olla mahdollista muuttaa Epecin ohjausyksiköihin sopivaksi.

Skriptin muunto Epecin ohjausyksiköiden kanssa sopivaksi vaikuttaa järkevältä jatkotoimenpiteeltä, koska tämä on tarvittava toimenpide jokaiselle projektissa käytettävälle ohjausyksikölle. Lisäksi tulevaisuuden kannalta, jos Epecin ohjausyksiköitä tullaan käyttämään enemmän on tällainen massaeditointiskripti oletettavasti hyvä tehdä.

Ohjelman siirron selkeyttämiseksi on hyvä säilyttää aiemmassa versiossa käytetyt globaalit muuttujat. Selkeyttämisen lisäksi samojen muuttujien käyttö helpottaa mahdollisten massaeditointiskriptien tekoa.

6 MASSAEDITOINTISKRIPTIEN SPESIFIKAATIOT

6.1 IO-lista

Työn tuloksiksi koottiin massaeditointiskripteille tarvittavat spesifikaatiot. Vanhemmassa CoDeSyS-versiossa olevan IO-listan (kuva 11) muuntoa ei ole järkevä tehdä massaeditointiskriptin avulla uuteen.

```
0022        CONNECTOR XM1
0023 ------------------------------------------------------------------------------*)
0024    DIDO_Sensor24VDCSupply_DIAG          AT %IX1.4: BOOL;      (* XM_1_1 can also have project depending usage! *)
0025    DO_Sensor24VDCSupply_DIAG            AT %QX1.4: BOOL;      (* XM_1_1 *)
0026 (* PWM_Name_Function                    AT %QW104: UINT;      (* XM_1_1 *)
0027
0028    DIDO_TurnBrakeRelease_DIAG           AT %IX1.5: BOOL;      (* XM_1_2 *)
0029    DO_TurnBrakeRelease_BOOM             AT %QX1.5: BOOL;      (* XM_1_2 *)
0030 (* PWM_Name_Function                    AT %QW105: UINT;      (* XM_1_2 *)
0031
0032 (* DI_Name_Function                     AT %IX1.7: BOOL;      (* XM_1_3 *)
0033 (* DO_Name_Function                     AT %QX1.7: BOOL;      (* XM_1_3 *)
0034    PWM_BoomDown_BOOM                    AT %QW107: UINT;      (* XM_1_3 *)
0035
0036 (* DI_Name_Function                     AT %IX1.6: BOOL;      (* XM_1_4 *)
0037 (* DO_Name_Function                     AT %QX1.6: BOOL;      (* XM_1_4 *)
0038    PWM_BoomUp_BOOM                      AT %QW106: UINT;      (* XM_1_4 *)
0039
0040 (* DI_Name_Function                     AT %IX3.1: BOOL;      (* XM_1_5 *)
0041    FB_BoomVerticalMotion_BOOM           AT %IW101: UINT;   (* XM_1_5 *)
0042
0043 (* DI_Name_Function                     AT %IX3.0: BOOL;      (* XM_1_6 *)
0044    FB_BoomTilt_BOOM                     AT %IW100: UINT;   (* XM_1_6 *)
```

KUVA 11. Osa vanhan CoDeSyS-version IO-listasta

Tämä johtuu siitä, että uuteen versioon IO-lista tulisi muuttaa Excel-muotoon ja suora muunto ei anna kaikkea tarvittavaa tietoa Excel-tiedostoon (kuva 12). Tarvittavia lisätietoja ovat esimerkiksi ulos- ja sisääntulon jännite ja virta, funktio sekä laitteen toiminta-alue. Lisäksi IO-listan täyttö on melko nopeaa valmiiseen Excel-pohjaan.

Function		Sensor / use	I/O name	Connection typ	Conn	Pin	Signal Type	I/O Type	Output function	Input function	Type detail	onnected devi
XXX		Supply Voltage for TCM1 inputs			XM1	01	DO	DOHighSideSwitching	3A 24V			
XXX		Turn brake release			XM1	02	DO	DOHighSideSwitching	3A			
XXX		Boom DOWN			XM1	03	PWM	AOPWM	3A			
XXX		Boom UP			XM1	04	PWM	AOPWM	3A			
XXX		Boom DOWN/UP			XM1	05	FB	AIFeedback		0-1A		0-10000
XXX		Tilt BEHIND/FRONT			XM1	06	FB	AIFeedback		0-1A		0-10000

KUVA 12. Uuteen versioon tehty IO-lista Excelissä

6.2 PDO -> XDC

PDO:ita tuodessa PLC-ohjelmasta XDC-tiedostoon tulee muuntaa COB-ID PDO:iden kommunikaatioparametreihin.

```
235     <!-- CANA first RPDO CAN ID = according to transmitter TPDO -->
236     <CANopenObject index="1400" name="RPDO0_CommunicationParameters" objectType="9" dataType="20">
237         <!-- subindex parameters Identified by name in export -->
238         <CANopenSubObject subIndex="01" objectType="7" dataType="07" accessType="rw" defaultValue="0x498" name="RPDO0_COB_ID"/>
```

KUVA 13. RPDO:n COB-ID “0x498” XDC-tiedostossa

Ensimmäisen vastaanotettavan PDO:n COB-ID 498h (kuva 13) muodostuu ID:stä ja PDO:sta. Tässä tapauksessa ID on 24 ja PDO on neljäs (kuva 14), joten ID:seen lisätään 480h.

$$498\text{h} - 480\text{h} = 18\text{h} => 24$$

```
0009 (* Receive PDO init parameters: ENABLE, PDO_TYPE_SELECT: false = tpdo, Node-ID [, Predefined set PDO-nr] *)
0010 (* Predefined set identifiers for PDO1..4 without node-ID which has to be added to the number in hex format 180-280-380-480 *)
0011     (* IW200..203 TRI 2 PDO 4 *)
0012     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=24, PDO:=4);
0013     (* IW204..207 TRI 2 PDO 7 *)
0014     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=56, PDO:=3);
0015     (* IW208..211 TRI 3 PDO 4 *)
0016     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=23, PDO:=4);
0017     (* IW212..215 TRI 3 PDO 7 *)
0018     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=55, PDO:=3);
0019     (* IW216..219 TRI 4 PDO4 *)
0020     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=22, PDO:=4);
0021     (* IW220..223 TRI 4 PDO7 *)
0022     CANOPEN_ADD_RECEIVE_PDO(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=54, PDO:=3);
```

KUVA 14. RPDO:n COB-ID PLC-ohjelmassa

Vanhassa PLC-ohjelmassa lähetettävien PDO:iden COB-ID saadaan vastaanotettavien lailla myös ID:stä ja PDO:sta (kuva 15). Lähetettävät PDO:t ovat neljän PDO:n sarjoissa, joissa ID pysyy samana neljälle peräkkäiselle PDO:lle. Ensimmäiset neljä PDO:ta (1-4) ovat ohjausyksikön omalla ID:llä

```
0029 (* Transmit PDO init parameters: ENABLE, PDO_TYPE_SELECT: false = tpdo, Node-ID [, Predefined set PDO-nr] *)
0030     (* QW200..203 QW204..207 QW208..211 QW212..215 included in CANopen initialisation *)
0031
0032     (* QW216..219 QW220..223 QW224..227 QW228..231 *)
0033     CANOPEN_ADD_TRANSMIT_NODE_ID(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=57);      (*39 TCM PDO 5-8 *)
0034     (* QW232..235 QW236..239 QW240..243 QW244..247 *)
0035     CANOPEN_ADD_TRANSMIT_NODE_ID(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=89);      (*59 TCM PDO 9-12 *)
0036     (* QW248..251 QW252..255 QW256..259 QW260..263 *)
0037     CANOPEN_ADD_TRANSMIT_NODE_ID(ENABLE:=TRUE, PDO_TYPE_SELECT:=FALSE, ID:=121);     (* 79 TCM PDO 13-16 *)
```

KUVA 15. Lähetettävien PDO:den ID

COB-ID:n lisäksi PDO:ssa olevat tiedot (kuva 16) kootaan subeihin. Kuvassa nähdään mitä PDO:ssa vastaanotetaan. Lähettävien ja vastaanottavien PDO:iden tiedot on merkitty samalla tavalla PLC-ohjelmaan.

```
0003 (*TRI 2 *)
0004     (*IW200*)
0005     CANI_Zoom2EPECbeat_DIAG                          AT %IX200.0: BOOL;
0006
0007     CANI_FrameSidewayshift_DISP                      AT %IX200.6: BOOL;
0008
0009     (*IW204*)
0010     CANI_TrimmerMovementsPowerRequest_BOOM_trimm     AT %IX204.0: BOOL;
0011
0012     CANI_FilterCleaningCleanTime_DISP                AT %IW205:UINT;
0013     CANI_FilterCleaningPulseUpTime_DISP              AT %IW206:UINT;
0014     CANI_FilterCleaningNumberOfCleanings_DISP        AT %IW207:UINT;
0015
0016 (*TRI 3*)
0017     (*IW208*)
0018     CANI_Zoom3EPECbeat_DIAG                              AT %IX208.0: BOOL;
0019
0020     CANI_Drill3SetToPauseByJamRelease_AutoSequence   AT %IX208.3:BOOL;
0021     CANI_Drill4SetToPauseByJamRelease_AutoSequence   AT %IX208.4:BOOL;
0022     CANI_Drill3JamReleaseActive_AutoSequence         AT %IX208.5:BOOL;
0023     CANI_Drill4JamReleaseActive_AutoSequence         AT %IX208.6:BOOL;
0024     CANI_WrongDrillsSelected_Autosequence            AT %IX208.7: BOOL;
```

KUVA 16. PDO:ssa olevat tiedot

Plc-ohjelmasta PDO:iden tiedot kootaan indexeihin kuvan 17 osoittamalla tavalla. Tiedot on hyvä tuoda käyttäen datatyyppejä 05 (8bittiä), 06 (16bittiä) ja 07 (32bittiä).

```
883     <CANopenObject index="3012" name="RpmAutomatics_RPDOs" objectType="8" accessType="rw">
884         <CANopenSubObject subIndex="01" objectType="7" dataType="05" accessType="rw" name="RpmAutomatics_Tri3">
885             <SICAMcIecBitSignal bit="0" name="RpmAutomaticsSelectFeedExtension_Tri3"/>
886             <SICAMcIecBitSignal bit="1" name="RpmAutomaticsSelectFeed_Tri3"/>
887             <SICAMcIecBitSignal bit="2" name="RpmAutomaticsSelectFrameMove_Tri3"/>
888             <SICAMcIecBitSignal bit="3" name="RpmAutomaticsSelectRetainer_Tri3"/>
889             <SICAMcIecBitSignal bit="4" name="RpmAutomaticsSelectRotation_Tri3"/>
890         </CANopenSubObject>
```

KUVA 17. PDO:den tiedot koottu XDC-tiedostossa indexiin ja subindexeihin

Kun PDO:iden tiedot on koottu, voidaan ne mapata omille PDO:lleen osoittamalla defaultValue-kohtaan index, subindex ja bittien määrä (kuva 18).

```
<CANopenObject index="1A04" name="TPDO4_Mapping" objectType="9" dataType="21" accessType="ro">
    <CANopenSubObject subIndex="01" objectType="7" dataType="07" accessType="ro" defaultValue="0x10160120" name="TPDO4_Signal_1" denotation="CANO_EpecBeat"/>
    <CANopenSubObject subIndex="02" objectType="7" dataType="07" accessType="ro" defaultValue="0x50120108" name="TPDO4_Signal_2" denotation="CANO_CommonEnableForOutputs"/>
    <CANopenSubObject subIndex="03" objectType="7" dataType="07" accessType="ro" defaultValue="0x30160110" name="TPDO4_Signal_3" denotation="CANO_SequenceAndFrameMovement"/>
    <CANopenSubObject subIndex="04" objectType="7" dataType="07" accessType="ro" defaultValue="0x30160208" name="TPDO4_Signal_4" denotation="CANO_DrillingModeCageAndFeedPressure"/>
</CANopenObject>
```

KUVA 18. PDO:iden mappaus (esim. defaultValue=0x30160110)

6.3 Parametrit -> XDC

PDO:den lisäksi XDC-tiedostoon on hyvä tuoda vanhan PLC-ohjelman parametrit (kuva 19). PLC-ohjelmassa parametreille on annettu oletusarvo ja viereen on kerrottu myös miten arvo muodostuu.

```
(* PAR_ParameterName_ModuleName      AT %MW0:Type:=Value *)

PAR_ExcavatorControlTimeLimitInUse_Tri1      AT %MW0:UINT:=1;        (*defines if excavator control time is limited with parameter, 1=true*)
PAR_MaxExcavatorControlTime_Tri1          AT %MW1:UINT:=60;       (*time in seconds, ( min=1 / max=65535 )*)
PAR_MaxFeedPressure_Tri1                  AT %MW2:UINT:=1250;     (*bar with one decimal, 1500=150,0bar, ( min=0 / max=65535 )*)
PAR_MinFeedPressure_Tri1                  AT %MW3:UINT:=100;      (*bar with one decimal, 300=30,0bar, ( min=1 / max=65535 )*)
```

KUVA 19. Parametrit vanhassa plc-ohjelmassa

Vanhat parametrit luodaan omiin subindexeihin ja annetaan PLC-ohjelmassa oleva oletusarvo myös XDC-tiedostoon (kuva 20).

```
<CANopenObject index="480A" name="Global_Parameters" objectType="8" SICAMcIecParamSetNr="04" accessType="rw">
    <CANopenSubObject subIndex="01" name="PAR_ExcavatorControlTimeLimitInUse" objectType="7" dataType="04" accessType="rw" defaultValue="1" groupName=""/>
    <CANopenSubObject subIndex="02" name="PAR_MaxExcavatorControlTime" objectType="7" dataType="04" accessType="rw" defaultValue="60" groupName=""/>
    <CANopenSubObject subIndex="03" name="PAR_MaxFeedPressure" objectType="7" dataType="04" accessType="rw" defaultValue="1250" groupName=""/>
    <CANopenSubObject subIndex="04" name="PAR_MinFeedPressure" objectType="7" dataType="04" accessType="rw" defaultValue="100" groupName=""/>
```

KUVA 20. Parametrit XDC-tiedostossa

7 POHDINTA

Työn tavoitteena oli tuoda vanhemmasta ohjelmointiympäristöstä ohjelmoinnin kannalta tarvittavat tiedot, kuten CAN-rajapinta, laitteen sisään- ja ulostulot sekä parametrit. Ohjelmointiympäristön tarvittavien tietojen lisäksi työn aikana kirjattiin ylös työn eri vaiheet ja selvitettiin spesifikaatiot massaeditointiskripteille.

Opinnäytetyöaihe vastasi odotuksia ja sille asetetut tavoitteet saatiin suoritettua. Työn aihe on hyvin keskeisessä osassa alaa. Työ paransi osaamista etenkin CAN-väylistä, tuotteiden elinkaaresta ja syvensi hieman ymmärrystä plc-ohjelmoinnista.

Opinnäytetyö selvitti Sandvikille tarvittavat toimenpiteet, joilla vanhempien kivenporauslaitteiden ohjausyksiköt voidaan päivittää uudempiin. Toimenpiteiden lisäksi työssä käytiin läpi mitä tarvitaan mahdollisiin massaeditointiskripteihin, jotta vanhojen ohjausyksiköiden päivitys uusiin tapahtuisi mahdollisimman helposti. Lisäksi työn aikana todettiin jo olemassa olevien skriptien toiminta Epecin ohjausyksiköillä ja tarvittavien korjausten jälkeen mallipohjat päivitettiin ajantasaisiksi.

Jatkotoimenpiteenä opinnäytetyölle ehdotetaan massaeditointiskriptiä, jonka avulla voidaan siirtää vanhemmasta PLC-ohjelmasta parametrit, PDO:iden signaalit sekä COB-ID XDC-tiedostoon. Lisäksi DMi/o_IO-objektin luontiin voi olla hyvä muokata aiempaa uudemman sukupolven moduulilla käytössä olevaa massaeditointiskriptiä.

LÄHTEET

CANopenSolutions. CANopen Basics - Process Data Exchange. Luettu 25.2.2019. https://www.canopensolutions.com/english/about_canopen/pdo.shtml

CiA. CiA 301 version 4.2.0, CANopen application layer and communication profile. Luettu 16.4.2019. https://www.can-cia.org/groups/specifications/

Epec Oy. Epec 2024 Control Unit Technical Document. Luettu 28.2.2019. http://www.bram-engineers.nl/wp-content/themes/arigato/extensions/Manuals_Epec/2024_M1/2024G08.pdf

Epec Oy. Epec 3724 Control Unit Technical Document. Luettu 28.2.2019. http://bram-engineers.nl/wp-content/uploads/2013/12/3724_MAN000553.pdf

Grieves, M. 2006. Product Lifecycle Management. USA: The McGraw-Hill Companies, Inc.

NationalInstruments. The Basics of CANopen. Luettu 25.2.2019. http://www.ni.com/white-paper/14162/en/

Niemann, J., Tichkiewitch, S. & Westkämper, E. 2009. Design of Sustainable Product Life Cycles. Germany: Springer-Verlag Berlin Heidelberg

Saaksvuori, A. & Immonen, A. 2008. 3. painos. Product Lifecycle Management Germany: Springer-Verlag Berlin Heidelberg

Technopedia. Development Environment. Luettu 15.4.2019 https://www.techopedia.com/definition/16376/development-environment

LIITTEET

Liite 1. Epec 2024 IEC Map (Epec Oy: Epec 2024 Control Unit Technical Document)