Jussi Merikoski

BECKHOFF MODBUS RTU PROTOKOLLA TAAJUUSMUUTTAJAN OHJAUKSESSA (VACON, ABB)

Automaatiotekniikan koulutusohjelma

2014

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

Merikoski, Jussi
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Toukokuu 2014
Ohjaaja: Asmala, Hannu
Sivumäärä:52
Liitteitä:4

Asiasanat: Modbus® RTU, kommunikointi protokolla, taajuusmuuttaja, Beckhoff ®PLC, ABB®, Vacon®

Tämän työn tarkoituksena oli luoda Beckhoff logiikalle ohjelma Modbus-kenttäväylä kommunikointi taajuusmuuttajille (Vacon ja ABB). Eri ohjelmia testattiin SAMK:in automaatiolaboratoriossa ja myöhemmin suoritettiin ABB:n taajuusmuuttajan käyttöönotto uusiutuvien energioiden hybridijärjestelmässä.

BECKHOFF MODBUS RTU FIELDBUS IN FREQUENCY CONVERTER CONTROL (VACON, ABB)

Merikoski, Jussi
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Automation
May 2014
Supervisor: Asmala, Hannu
Number of pages:52
Appendices:4

Keywords: Modbus® RTU, communication protocols, frequency converters, Beckhoff® PLC, ABB®, Vacon ®

This thesis work purpose was to create PLC program for Beckhoff PLC. Program was intended to be utilized in fieldbus communications between PLC and various frequency converters (ABB and Vacon). Program was tested in SAMK automation laboratory and later program was commissioned in renewable energies hybrid system with ABB ACH550 frequency converter.

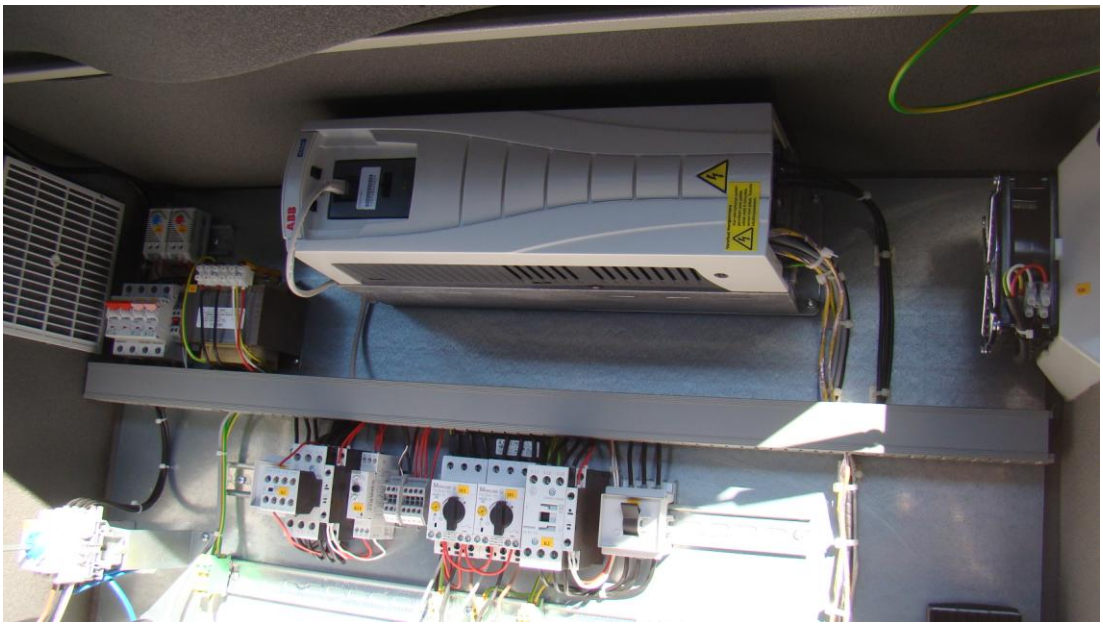
SISÄLLYS

1 JOHDANTO

SAMK:in (Satakunnan Ammattikorkeakoulu) uusiutuvien energioiden hybridijärjestelmään kuuluu mukaan ABB:n taajuusmuuttaja ACH 550, joka säätää jäähdytyspuhaltimen kierrosnopeutta ja näin ollen jäähdytyskennojen läpi virtaamaa ilmamäärää/jäähdytystehoa. Jäähdyttimen avulla estetään hybridijärjestelmän kiertonesteen ylikuumeneminen.

Uusiutuvien energioiden hybridijärjestelmä oli jo osittain otettu käyttöön, mutta kommunikointi ohjausjärjestelmästä (Beckhoff PLC) taajuusmuuttajaan oli vielä käyttöönottamatta. Taajuusmuuttajan ohjaus jouduttiin hoitamaan manuaalisesti aina, kun puhaltimia haluttiin käyttää. Manuaalinen käyttö merkitsi käyntiä rakennuksen katolle ikkunan kautta, jäähdytyskennojen viereen asennetun taajuusmuuttajan sähkökeskuksen avaamista, taajuusmuuttajan käynnistystä ja nopeuden säätöä taajuusmuuttajan ohjauspaneelista. Taajuusmuuttajan käyttö oli tässä muodossaan hankalaa ja epäkäytännöllistä. Kuvissa 1-3 on esitetty sähkökeskus ja keskuksen sisäkuvia.

Kuva.1 Jäähdytysyksikkö, taajuusmuuttaja sijaitsee nuolen osoittamassa sähkökeskuksessa



Kuva.2 Taajuusmuuttaja kuvan 1 sähkökeskuksessa

Kuva.3 Taajuusmuuttajan ohjauspaneeli

Sain tehtäväkseni perehtyä Beckhoff PLC:n ja taajuusmuuttajan väliseen Modbus kommunikointiin. Modbussista minulla oli vähän jo aikaisempia kokemuksia, mutta Beckhoff PLC oli minulle aivan uusi tuttavuus.

Automaatiolaboratorioon rakennettiin pieni testikokonaisuus, jolla testasin Beckhoff Modbus RTU masteria Vacon NXS taajuusmuuttajan kanssa. Projekti osoittautui paljon pitempikestoiseksi ja haastavammaksi kuin alun perin olin ajatellut. Työskentelin

projektin kanssa lähes viikoittain, aina vähintään yhden päivän viikossa, mutta useimpina viikkoina käytin projektiin kaksi päivää viikossa. Edellä mainitulla työskentelyvauhdilla saimme ohjelman ja kommunikointiväylän toimimaan ABB:n taajuusmuuttajan kanssa Maaliskuun lopulla 2014. Projektin aloitus oli tapahtunut Marraskuun 2013 loppupuolella, tosin Beckhoffin moduulien tilaaminen aiheutti parin viikon viiveitä projektiin.

Projektissa ilmeni monia erilaisia vaikeuksia, joista suurin oli tiedonpuute tai paremminkin oikean tiedon löytäminen, johon ensin alkuun käytin paljon ajasta. Aluksi aikaa vei myös ohjelmointiohjelmaan tutustuminen ja käyttöön perehtyminen (Beckhoff TwinCat 3 ja Microsoft Visual)

Pidin projektia mielenkiintoisena, vaikka jossain vaiheessa välillä tuntuikin hankalalta saada kommunikointi Beckhoffin ja taajuusmuuttajan välillä toimimaan halutulla tavalla.
Haluan kiittää Beckhoffin teknistä tukea (Timo Anttila, Teppo Lepistö ja Aapo Vuoristo) saamastani tuesta. Samoin esitän kiitokseni myös Hannu Asmalalle, jolta sain ohjelmointineuvoja. Kiitokset myös Jyrki Mäkiturjalle, jolta sain tukea Vacon Modbus kortin käytössä.

2 MODBUS

2.1 MODBUS TAUSTAA

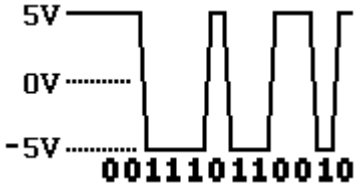
Modbus on Modiconin luoma sarjaliikenneprotokolla, joka alun perin oli tarkoitettu käytettäväksi Modiconin ohjelmoitavien logiikoiden väliseen tiedonsiirtoon. Alun perin Modbus kehitettiin RS-232 (RTU ja ASCII) sarjaliikenteelle, myöhemmin väylätekniikoiden kehittyessä on mukaan tullut myös RS-485 ja Ethernet (TCP protokolla) (Simply Modbus, 2013)

Modbus on isäntä-orja (Master-Slave) tyyppinen kommunikointi protokolla, jossa on väylällä 1 (yksi) isäntä, joka suorittaa kyselyitä tai kirjoittaa dataa tiettyyn orjaan, joiden määrä voi olla jopa 247 yhdessä Modbus väylässä. Jokaisella orjalla on yksilöllinen ID osoite/numero, jonka avulla isäntä kommunikoi orjan kanssa. Orja lähettää dataa vain isännän pyynnöstä.
(Simply Modbus, 2013)

Modbus on lisenssivapaa, joten se on laitevalmistajille vapaasti integroitava kommunikointiprotokolla. Maksuton Modbussin käyttö on tehnyt Modbussista suositun ja yleisesti käytetyn laiteliityntä protokollan teollisten elektroniikka laitevalmistajien keskuudessa, esimerkkinä voisi mainita vaikkapa SymCom RM-2000 moottorin käynnistysvalvontalaitteen, sen keräämän tiedon voi lukea Modbus väylää käyttäen. Modbus on muodostunut eräänlaiseksi laitevalmistajien yleisprotokollaksi.
(Simply Modbus, 2013)

2.2 MODBUS KOMMUNIKOINTI

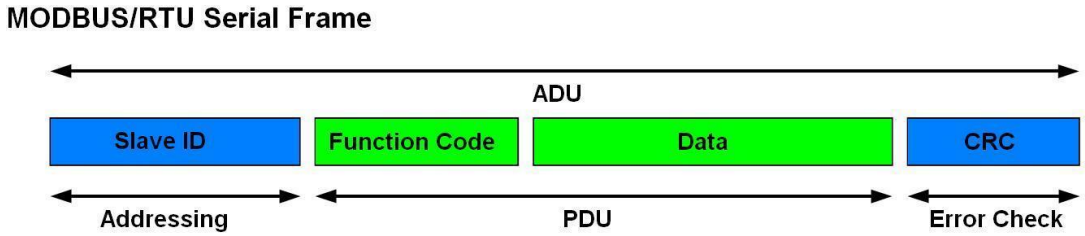
Modbus sarjaliikennekommunikoinnissa data lähetetään sarjana “1” (ykkösiä) ja “0” (nollia), joita kutsutaan biteiksi. Jokainen bitti lähetetään jännitetasona, nolla ”0” lähetetään positiivisena jännitteenä ja ykköset ”1” lähetetään negatiivisena jännitteenä. Kuvassa 4 on esitetty Modbus sarjaliikenteen jännitetasot. Tyypillinen lähetysnopeus on 9600 baudia (bittiä/sekunti), mutta nopeampiakin tiedonsiirtonopeuksia voidaan käyttää. (Simply Modbus, 2013). Tässä projektissa käytin 19200 baud nopeutta, mutta vieläkin suurempia nopeuksia voidaan käyttää, riippuen kommunikointietäisyyksistä ja mahdollisista häiriölähteistä. Suuremmissa etäisyyksissä täytyy nopeutta pudottaa, muutoin sanomat saattavat vääristyä ja kommunikoinnissa saattaa esiintyä paljon virheitä ja pahimmassa tapauksessa kommunikointi ei toimi ollenkaan.



Kuva.4 Modbus sarjaliikenteen jännitetasot

2.2.1 Sanoman rakenne

Modbus sanoma rakentuu orjan osoitteesta, toimintakoodista, datasta ja virhetarkistuksesta. Orja ID:n avulla jokainen verkossa oleva orjalaite pystyy tunnistamaan täytyykö tulevaan viestiin reagoida vai jätetäänkö viesti huomioimatta.
Modbus sanoman rakenne on esitetty kuvassa 5.



Kuva.5 Modbus sanoman rakenne (Grid Connect, 2012)

Alla olevassa listauksessa on esitetty miten sanoma rakentuu numeerisesti. Esimerkkinä on isännän suorittama kysely orjalaitteelta, jonka ID on 17. Esimerkkitapauksessa kyselyssä halutaan analogisen tulo rekisterin 30009 tieto.

Lue analoginen tulo rekisteri (Toimintakoodi FC=04)

Kysely

Tällä kyselyllä luetaan analogisen tulo rekisterin # 30009 tieto orjasta, jonka ID osoite on 17
11 04 0008 0001 B298

11: Orjan osoite (17 = 11 hex)
04: Toimintakoodi (Lue analoginen tulo rekisteri)
0008: Ensimmäisen data rekisterin osoite. (30009-30001 = 8)
0001: Luettavien rekisterien määrä. (lue 1 rekisteri)
B298: CRC (cyclic redundancy check) Virhetarkistus

Vastaus orjalta

11 04 02 000A F8F4

11: Orjan osoite (17 = 11 hex)
04: Toimintakoodi (Lue analoginen tulorekisteri)
02: Tavujen määrä (1 rekisteri x 2 tavua jokaista varten = 2 bytes)
000A: Rekisterin 30009 sisältö
F8F4: CRC (cyclic redundancy check) virhetarkistus

(Simply Modbus, 2013)

2.2.2 Modbus rekisterityypit

Modbusissa on määritelty useita erilaisia rekisterityyppejä. Tarkat määritelmät löytyvät Modbus.org sivuilta löytyvästä pdf dokumentista,
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
(Modbus, 2014)

Monet laitevalmistajat käyttävät termejä coil, analog register. Nämä saattavat ensin alkuun vaikuttaa oudoilta käsitteiltä ja niistä muodostuu helposti väärinkäsityksiä. Seuraavassa on lyhyt selvennys rekisterityypeistä.

Input coil tai output coil tarkoittaa rekisterityyppiä, joka sisältää vain on / off bittitietoa. Modbus määritelmissä on toimintakoodit input ja output coil rekistereille, niistä tarkempi selostus Modbus toimintakoodi kappaleessa 2.2.3.

Analog register input/output ja analog holding register. Tässä rekisterityypissä on numeerista tietoa joko orjalaitteeseen lähtevää tai siihen isännästä kirjoitettavaa tietoa. Holding register tyyppiin voidaan sekä lukea että kirjoittaa.

Taulukko 1. Modbus rekisterityypit (Simply Modbus, 2013)

Coil/Register Numbers	Data Addresses	Type	Table Name
1-9999	0000 to 270E	Read-Write	Discrete Output Coils
10001-19999	0000 to 270E	Read-Only	Discrete Input Contacts
30001-39999	0000 to 270E	Read-Only	Analog Input Registers
40001-49999	0000 to 270E	Read-Write	Analog Output Holding Register

Jokaiselle rekisterityypille varataan 1-9999 rekisteritunnusta, mutta huomattavaa on, että rekistereiden osoitteet alkavat nollasta (0). Tästä johtuen kun laitevalmistajan ensimmäinen osoite on esim. 2001 (esim. Vacon), täytyy kyselyssä antaa osoitteeksi 2000, jotta kysely suoritetaan oikeasta osoitteesta.
Rekisterityyppien määritelmät löytyvät laitevalmistajan ohjekirjoista, niistä ilmenee osoitteiden tyypit esim. Holding register 40002 (ABB).
(ABB Fieldbus, 2007)

2.2.3 Modbus toimintakoodit

Modbussissa käytettävät toimintakoodit (function code) on hyvin kattavasti selitetty jo aiemmin mainitussa Modbus määritelmä dokumentissa, josta löytyvät tarkat selitykset ja määritelmät kullekin eri toimikoodille.
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
(Modbus, 2014)

Taulukko 2. Yleisimpiä Modbus toimintakoodeja

Function Code	Action	Table Name
01 (01 hex)	Read	Discrete Output Coils
05 (05 hex)	Write single	Discrete Output Coil
15 (0F hex)	Write multiple	Discrete Output Coils
02 (02 hex)	Read	Discrete Input Contacts
04 (04 hex)	Read	Analog Input Registers
03 (03 hex)	Read	Analog Output Holding Registers
06 (06 hex)	Write single	Analog Output Holding Register
16 (10 hex)	Write multiple	Analog Output Holding Registers

Käytetty toimintakoodi määräytyy sen mukaan minkälaista toimintaa (FB) Modbus kommunikoinnissa käytetään. Esimerkiksi, jos PLC ohjelmassa käytetään toimintoa *ReadRegs*, kirjoittaa ohjelma kyselyssä toimintakoodiksi 03, tai jos vastaavasti halutaan kirjoittaa useampaan rekisteriin *Writeregs*, saa kysely toimintakoodiksi 16. (Beckhoff)

Toimintakoodeista ei ohjelmoitsijan tarvitse murehtia, vaan PLC ohjelma huolehtii oikean toimintakoodin käyttämisestä kyselyssä, tämä tapahtuu automaattisesti jossain PLC:n systeemikooditasolla. Beckhoffin järjestelmässä ja useilla muilla PLC valmistajilla toiminnot ovat vastaavanlaiset. Ohjelmoitsijan tarvitsee vain huolehtia, että käytössä on oikea toiminta, ettei esim. yhden rekisterin toiminnalla yritetä lukea useaa rekisteriä kerralla.

Seuraavassa listassa on Beckhoffin käyttämiä toimintakoodeja Modbus RTU protokollalle. Samat toimintakoodit pätevät myös Modbus TCP:lle. Koska TCP:n virhetarkistus on erilainen ja näin ollen sanoma on erilainen, ei samoja toimilohkoja voi käyttää TCP:lle ja RTU:lle.

Beckhoff MB RTU toimintakoodit ja funktiot;

Supported Modbus functions (actions)

- **ModbusMaster.ReadCoils**
  Modbus function 1 = *Read Coils*

  Reads binary outputs (coils) from a connected slave. The data is stored in compressed form (8 bit per byte) starting from address *pMemoryAddr*.

- **ModbusMaster.ReadInputStatus**
  Modbus function2 = *Read Input Status*

  Reads binary inputs from a connected slave. The data is stored in compressed form (8 bit per byte) starting from address *pMemoryAddr*.

- **ModbusMaster.ReadRegs**
  Modbus function 3 = *Read Holding Registers*

  Reads data from a connected slave.

- **ModbusMaster.ReadInputRegs**
  Modbus function 4 = *Read Input Registers*

  Reads input registers from a connected slave.

- **ModbusMaster.WriteSingleCoil**
  Modbus function 5 = *Write Single Coil*
  Sends a binary output (Coil) to a connected slave. The data is supposed to be stored in a compressed form (8 bit pro byte) starting from *address pMemoryAddr*.

- **ModbusMaster.WriteSingleRegister**
  Modbus function 6 = *Write Single Register*
  Sends a single data word to a connected slave

- **ModbusMaster.WriteMultipleCoils**
  Modbus function 15 = *Write Multiple Coils*
  Sends binary outputs (Coils) to a connected slave. The data is supposed to be stored in a compressed form (8 bit pro byte) starting from *address pMemoryAddr*.

- **ModbusMaster.WriteRegs**
  Modbus function 16 = *Preset Multiple Registers*
  Sends data to a connected slave

- **ModbusMaster.Diagnostics**
  Modbus function 8 = *Diagnostics*
  Sends a diagnostics request with a user defined subfunction code to a connected slave. The subfunction code is passed to the function by the *MBAddr* parameter. Additional data can be passed by *pMemoryAddr*.

(Beckhoff)

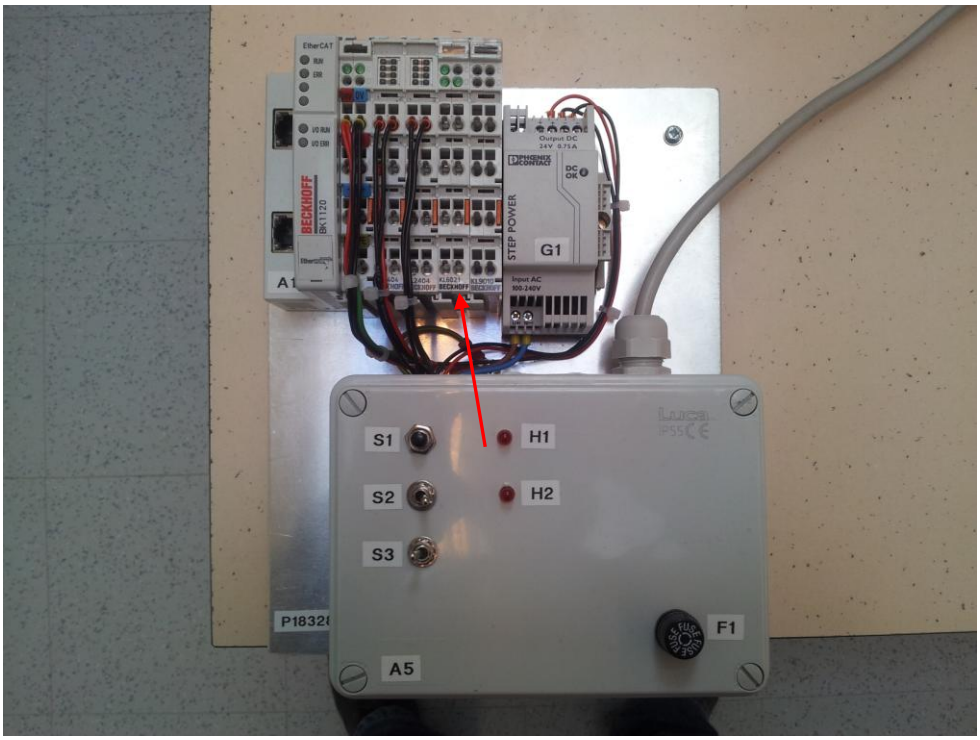
3 LAITEKOKOONPANOT

3.1 BECKHOFF LAITTEISTO

Projektin aikana oli käytössä kolme (3) erilaista Beckhoff laitekokokoonpanoa. Kahdella kokoonpanolla suoritettiin testauksia laboratoriossa Vacon NXS taajuusmuuttajan kanssa. Kolmas laitekokonaisuus oli itse uusiutuvien energioiden hybridijärjestelmää ohjaava Beckhoff laitteisto, jossa erona aikaisempiin testeihin oli taajuusmuuttaja, joka tässä järjestelmässä oli ABB:n 550 ACH.

3.1.1 KL6021 Moduulin laitekokoonpano (Testilaitteisto 1)

Ensimmäisenä laitekokoonpanona laboratoriotesteissä oli olevan kuvan 6 mukainen pieni testilaitteisto, jossa oli KL6021 kortin lisäksi digitaalitulo- ja digitaalilähtökortti. Kuvassa 6 KL6021 korttia ennen ovat tulo- ja lähtökortit. KL6021 jälkeen on vain terminointikortti.



Kuva.6 Beckhoff laitekokoonpano KL6021 kortilla. Nuoli osoittaa KL6021 kortin

Ensimmäiseksi yritin suorittaa kommunikointitestiä em. laitteistolla, mutta KL6021 kortin sijasta, asensin EL6021, joka oli lainattu uusiutuvien energioiden hybridijärjestelmän laitteistosta. Molempien korttien johdotuskytkentä on samanlainen. Johdotuskaavio löytyy Beckhoffin Internet-sivuilta löytyvästä pdf dokumentista KL6021en.pdf, sivulla 6 (Beckhoff, 2006)

RS-485 kommunikoinnissa joudutaan kumpikin kortti johdottamaan half duplex moodiin, mikä tarkoittaa, että kommunikointi toimii vain yhteen suuntaan kerralla. Full duplex pystyy kommunikoimaan molempiin suuntiin samanaikaisesti, eli lähettämään ja vastaanottamaan viestejä.

Yritin jonkin aikaa saada yhteyttä Vacon NXS taajuusmuuttajan ja ensin EL6021 kortin välillä. Kommunikointi ei vaan osoittanut mitään elonmerkkejä, joten otin yhteyttä Timo Suvelan antamaan kontaktiin Beckhoffilla. Teppo Lepistö kertoi minulle hyvin nopeasti kertomani laiteselostuksen jälkeen, ettei EL6021 ole yhteensopiva K-sarjan laitteiston kanssa. Tuo pieni testilaitteisto oli K-sarjaa, mikä selvisi minulle nyt ensimmäistä kertaa.

Tätä pientä laitteistoa varten jouduttiin nyt tilaamaan oma kortti KL6021-S, joka oli viiden (5) tavun eli byten kortti. Tämä tarkoittaa sitä, että kortti ei pysty välittämään kuin korkeintaan 4 tavua eli 2 sanaa samassa datapaketissa. Tosin korttia käytetään vain testauslaitteessa ja tapa millä kommunikoitiin, on 5 tavun kortti riittävä.

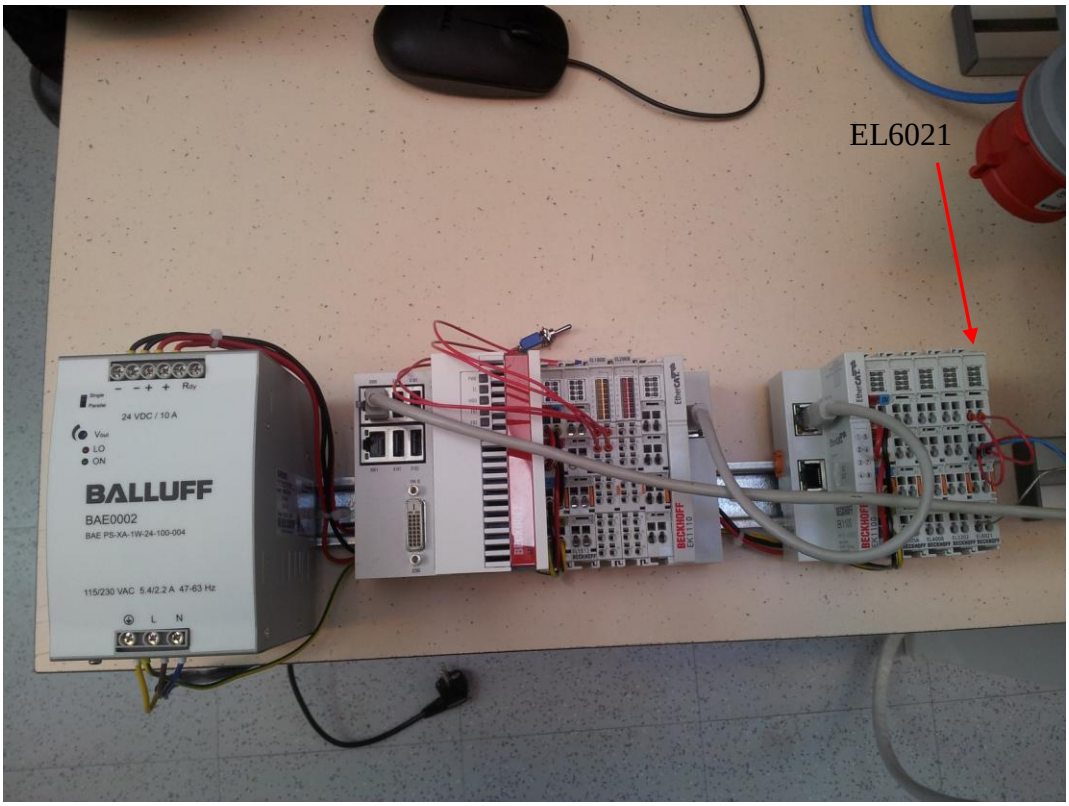
Uusi K-sarjan kortti saatiin jonkin ajan kuluttua ja asensin kortin pieneen testilaitteeseen. Yrityksistäni huolimatta en saanut minkäänlaista elonmerkkiä kommunikointiin, joten otin uudelleen yhteyttä Beckhoffin tekniseen tukeen Lepistölle, joka otti etäyhteyden testilaitteeseen Team Viewer ohjelmalla.
Selvisi, että kortti ei ollut ohjelmoitu 5 tavun kortiksi vaan kortti olikin ohjelmoitu 3 tavun kortiksi. Lepistö sai onneksi uudelleenohjelmoitua kortin 5 tavun kortiksi etäkäytön kautta.

Sain myös lisäapuja Beckhoffin Modbus kommunikointiin myöhemmin Timo Anttilalta Beckhoffilta.
Kuten jo aiemmin olen alussa maininnut, Beckhoff PLC oli minulle uusi ohjelmointiympäristö, joten tässä vaiheessa kaikki mahdollinen tuki oli tervetullutta. Timo Anttila lähetti minulle esimerkkiohjelman Modbus kommunikoinnista Beckhoff PLC:ssä.
Pian minulle selvisi, että ohjelmistosta puuttui kokonaan ohjelmistopaketti TF6340 TC3 serial communication, joka sisältää KL6configuration toimilohkot ja toiminnat. Olin kyllä yrittänyt ladata tuota ohjelmistopakettia, mutta ohjelmiston asennus keskeytyi joka kerta. Timo Anttilan ohjeilla ja avustuksella sain viimein ohjelmistopaketin asennettua PC:lle ja näin ollen lisättyä korttialustukseen tarvittavat toiminnat.
Kuvaus KL6021 kortin alustuksesta löytyy kohdasta 5.1.5

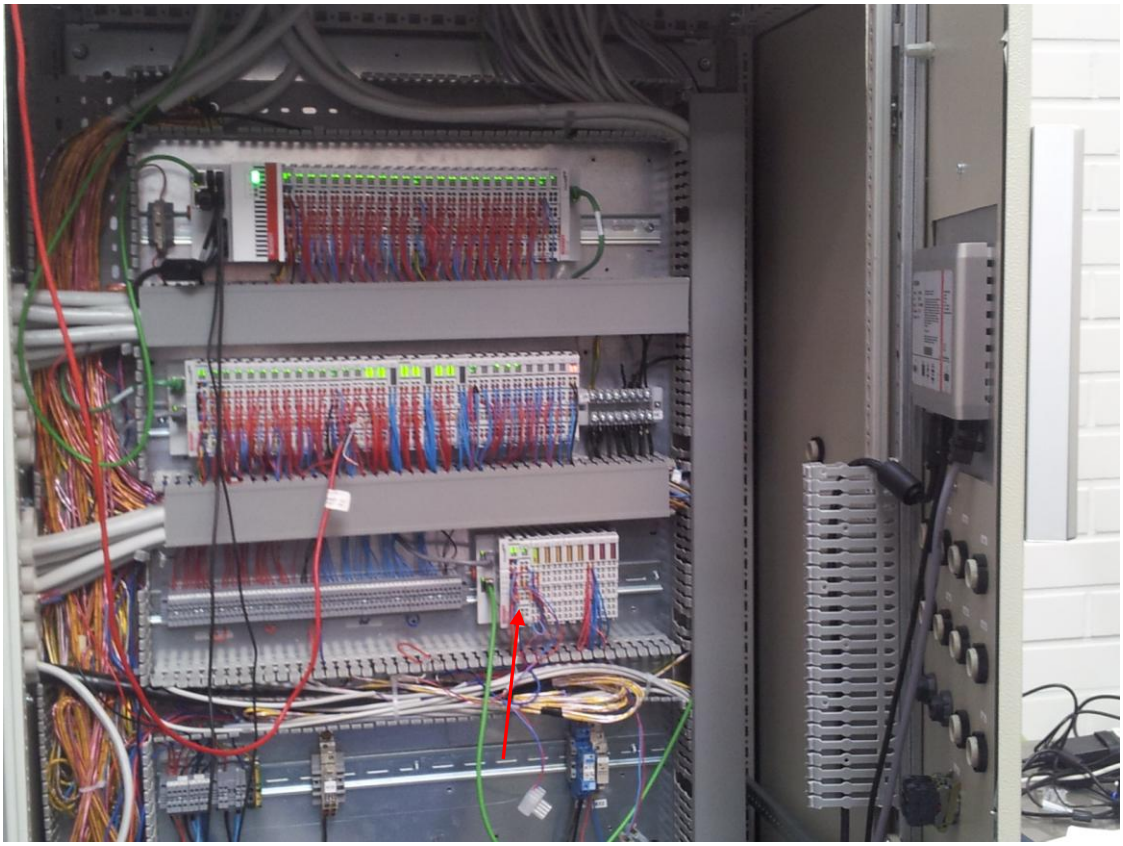
3.1.2 EL6021 Moduulin laitekokoonpano (Testilaitteisto 2)

EL6021 moduulia testattiin vasta kun KL6021 moduuli ja ohjelma oli saatu kommunikoimaan Vacon NXS taajuusmuuttajan kanssa. Kuvassa Kuva.7 on EL6021 kommunikointimoduulille tarkoitettu laitekokoonpano. Muut kuvassa näkyvät kortit ovat pääosin tulo- ja lähtökortteja.

Kuvassa Kuva.7 on varsinainen prosessinohjausjärjestelmä, eli laitteisto, jota varten kommunikointia testattiin.



Kuva.7 EL6021 testilaitteisto. EL6021 on tekstin osoittamassa paikassa



Kuva.8 Uusiutuvien energioiden hybridijärjestelmän ohjauslaitteisto.
EL6021 nuolen osoittamassa paikassa

KL6021 ja EL6021 vaikuttavat ja näyttävät niin samanlaisilta korteilta, että oletin kortin alustus eli konfiguroiti tehtädään molemmilla korteilla samalla tavalla. Ohjelman alussa oli konfigurointi-toimilohko, johon muutettiin Mode tyypin 5 tavusta 22 tavuun EL6021 korttia varten. Aikani yritin saada kommunikointilinjaa heräämään tämän kortin ja Vaconin taajuusmuuttajaan kanssa. Kun homma ei tuntunut lähtevän toimimaan, otin yhteyttä Beckhoffin tukeen. Teppo Lepistö otti Team Viewer ohjelman avulla etäyhteyden testilaitteistoon ja näytti miten EL6021 kortin alustus TwinCat 3:ssa suoritetaan. EL6021 kortille on olemassa ns. korttiparametrit, jotka löytyvät TwinCat 3 valikoista. Tarkempi kuvaus kortin alustuksesta löytyy kohdasta 5.1.6

4 TAAJUUSMUUTTAJAT

4.1.1 Vacon taajuusmuuttajan tyyppi ja ominaisuudet

Laboratoriotesteissä käytetty taajuusmuuttaja oli Vacon NXS, joka on tarkoitettu lähinnä taajuusohjattuihin sovelluksiin, kuten esim. pumput ja puhaltimet. NXS ei omaa yhtä hyviä nopeussäätöominaisuuksia kuin NXP-sarjan taajuusmuuttajat, jotka ovat tehty vaativimpiin nopeussäätösovelluksiin.

Tähän testisovellukseen NXS oli aivan riittävä, koska tarkoituksena oli testata vain Modbus-väylää. Riitti kun taajuusmuuttajan sai käynnistettyä väylän kautta ja luettua oloarvosignaaleita väylän Beckhoffin logiikkaan.

Kuva.9 Laboratoriotestauksissa käytetty Vacon NXS taajuusmuuttaja

4.1.2 Vacon OPT C2 kommunikointikortti

Modbus kommunikointia varten taajuusmuuttajaan asennettiin OPT C2 kommunikointikortti. Kortin asennusohjeet on saatavilla pdf muodossa Vaconin web-sivustolta.
http://www.vacon.com/ImageVaultFiles/id_3118/cf_2/Vacon-NX-OPTC2-C8-Modbus-N2-Board-User-Manual-DPD0.PDF?635223804391930000
(VACON OPTC2 , 2012)

Ohjeissa on kerrottu tarkasti kortin asennus taajuusmuuttajaan. Kuvalliset ohjeet ovat selkeät ja havainnolliset.

4.1.3 Vacon parametrointi

Vaconin taajuusmuuttajan ja lisäkortin parametrointi on hyvin yksinkertainen toimenpide, joka on selostettu Vaconin ohjekirjoissa.
Ensimmäinen toimenpide, jonka itse suoritin oli sovellusohjelman valinta ohjauspaneelin systeemiparametrivalikosta. Valitsin sovellukseksi erikoiskäyttösovelluksen, koska tiesin tuon entuudestaan tukevan useita kenttäväyliä. Tosin muutkin sovellukset olisivat tukeneet Modbus- kenttäväylää. Sovelluksen valinta on selostettu Vacon NXS käyttöohjeissa;
http://www.vacon.com/ImageVaultFiles/id_2784/cf_2/Vacon-NXS-NXP-User-Manual-DPD01218A-FI.PDF?634995835905300000
(VACON NXS User Manual, 2012)

Modbus kenttäväylän parametreja ei ole montaa aseteltavana, mutta koska C2 optiokortti tukee myös Metasys eli N2 kenttäväylää, on yksi parametreista protokollan valinta.

Muut asteltavat parametrit ovat slave address (orjan ID osoite), joka asetettiin arvoon=1, Baudrate (baudinopeus) = 19200, Parity type (pariteetti)= 1 eli Even, ja communication timeout (kommunikointikatkoksen kesto) jolle annoin arvoksi 10 s. Jos taajuusmuuttaja ei havaitse kommunikointia 10 sekuntiin, tehdään vika. Tämä viive toimii ohjauspaikan ollessa valittuna fieldbus eli kenttäväyläohjaukselle, vaikka ohjekirja ei mainitse mitään ohjauspaikasta, testauksen aikana ei kommunikointivikaa ilmennyt muulloin kuin ohjauspaikan ollessa valittuna kenttäväylälle.
Parametrien selostukset löytyvät Vacon OPTC2 käyttöohjeista, jotka löytyvät Vaconin web-sivustolta;
http://www.vacon.com/ImageVaultFiles/id_3118/cf_2/Vacon-NX-OPTC2-C8-Modbus-N2-Board-User-Manual-DPD0.PDF?635223804391930000
(VACON OPTC2 , 2012)

4.1.4 ABB taajuusmuuttaja tyyppi ja ominaisuudet

Uusiutuvien energioiden hybridijärjestelmän taajuusmuuttajana on ABB:n ACH550. Se on suunniteltu erityisesti HVAC (Heat Ventilation and Air Conditioning) eli LVI sovelluksille. ABB:n taajuusmuuttajassa on sisäänrakennettu kenttäväyläkommunikointi, joka tukee tyypillisimpiä HVAC puolen kenttäväyliä. Sisäänrakennettu kommunikointi tukee seuraavia kenttäväyliä; Modbus, Metasys N2 ja APOGEE FLN. Erillistä lisämoduulia ei siis tarvinnut asentaa.

4.1.5 ABB parametrointi

ABB taajuusmuuttajan parametrointi osoittautui hieman monimutkaisemmaksi kuin Vaconin taajuusmuuttajan. ABB:n taajuusmuuttajassa on enemmän parametreja, jotka täytyy asetella kenttäväyläohjausta varten. Pelkän kommunikointi parametrien asettelut ovat jotakuinkin samat kuin Vaconin taajuusmuuttajassa., esim. Slave address, Baudrate, Parity, jne. Kenttäväyläohjauksen asetuksista lisää kappaleessa BECKHOFF-ABB TESTAUS.
ABB:n kenttäväylälle löytyy ohjekirja Internet-sivustolta. Ohjekirjasta löytyvät niin parametriselostukset, kuin myös askelkaaviot taajuusmuuttajan ohjaukseen kenttäväylän kautta.
http://www05.abb.com/global/scot/scot201.nsf/veritydisplay/25ba8ab3f04e2266c12572e9004ffafe/$file/en_ach550_efb_d.pdf
(ABB Fieldbus, 2007)

Edellä mainitun oppaan lisäksi tarvitsin usein tietoa ACH550 käyttäjän käsikirjasta (User’s manual), josta löytyi tarkempia selostuksia monitoroitavista signaaleista ja parametreista. Lisäksi käsikirjasta löytyy selostus ohjauspaneelin käytöstä, joka aluksi oli minulle vierasta.
Ohjekirjan voin ladata ABB:n verkkosivuilta;
http://www05.abb.com/global/scot/scot201.nsf/veritydisplay/7ccd70ad6b24df53c1257607003d1969/$file/en_ach550_01_um_f_a4_loscreenres.pdf
(ABB ACH550, 2009)

5 OHJELMOINTI

5.1 BECKHOFF LOGIIKAN OHJELMOINTI

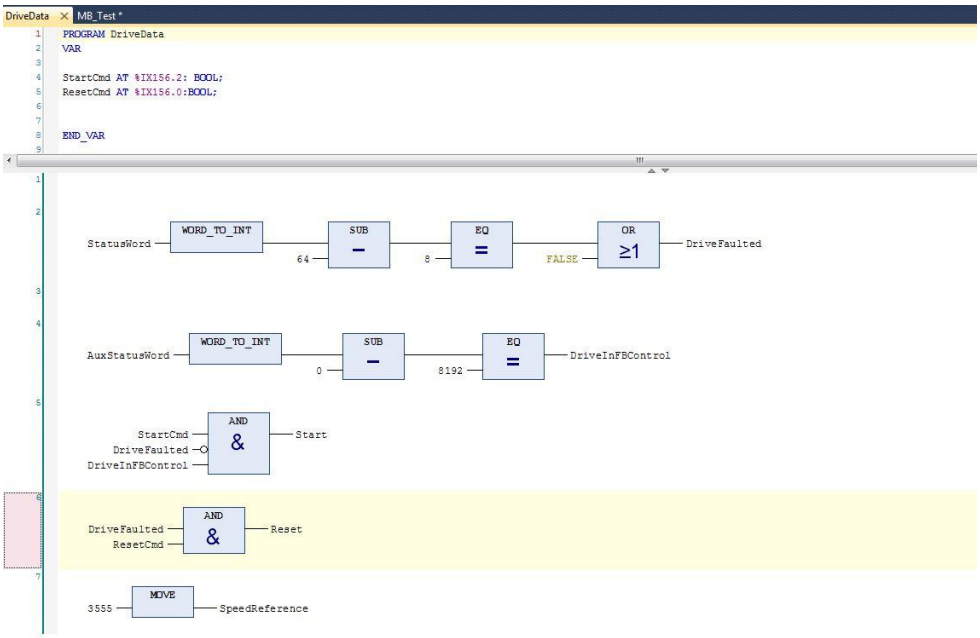
Kuten jo aiemmin mainitsin, ennen tätä projektia ei minulla ollut minkäänlaista kokemusta Beckhoffin PLC ohjelmoinnista tai laitteista, joten oppimiskäyrä oli aluksi melko jyrkkä. Onneksi IEC 61131-3 standardin myötä logiikoiden ohjelmointi on yhtenäistynyt. Toisin sanoen logiikoiden ohjelmointi on useilla valmistajilla hyvin samanlainen, mutta pieniä eroavaisuuksia tietysti löytyy ohjelmointityökaluissa.

Tämän ohjelman päällimmäisenä tehtävänä oli tutkia ja ohjelmoida kenttäväylä-kommunikointi. Ohjelmassa ei puututtu lopullisen järjestelmän prosessiohjauksen ohjelmasovellukseen. Tärkeintä oli saada luotua ohjelma, jonka avulla prosessiohjaus voisi kommunikoida ja kontrolloida taajuusmuuttajaa Modbus kenttäväylää käyttäen.

5.1.1 TwinCat 3 ohjelmointityökalu

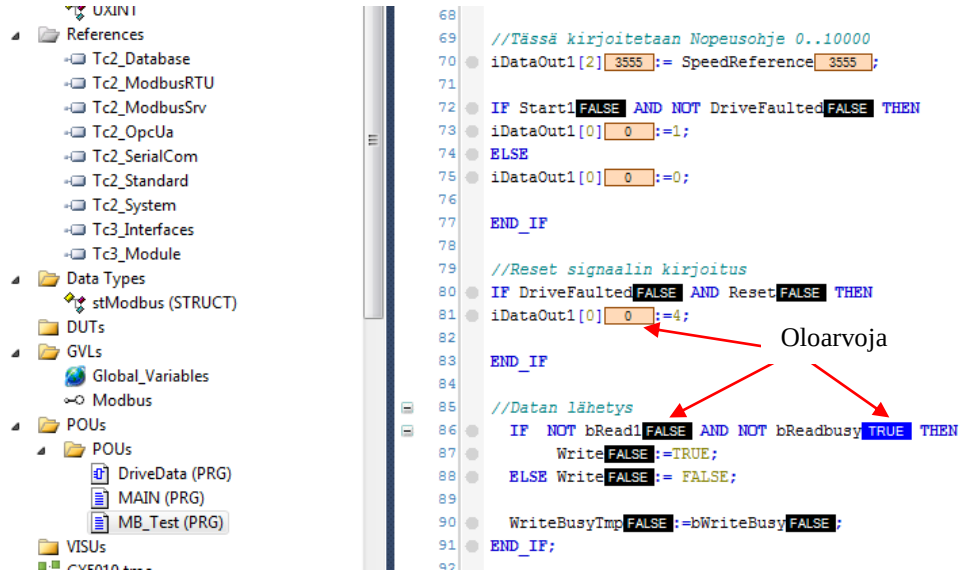
Beckhoffin TC3 (Twincat 3) ohjelmointiympäristö pyörii Microsoftin Visual Studio ohjelmassa. TwinCat on IEC 61131-3 standardin mukainen, joten se tukee IL (Instruction List), ST (Sturtured Text), LD (Ladder Diagram), FBD (Function Block Diagram), SFC (Sequential Function Chart) ja CFC (Continuous Function Chart) ohjelmointikieliä. Itse käytin ohjelmoinnissa ST ja FBD ohjelmointia.

Pääohjelman eli varsinaisen kommunikointiohjelman kirjoitin käyttäen ST ohjelmointia, johon oli hyvä syykin. Esimerkkiohjelma, jonka sain Beckhoffin tekniseltä tuelta, Timo Anttilalta, oli kirjoitettu tuolla ohjelmointikielellä. En nähnyt tarpeelliseksi alkaa muuttamaan ohjelmaa toiselle kielelle, joten ohjelmaa muokattiin ST käskylistauksessa. Käytin FBD ohjelmointikieltä yhdessä ohjelman osassa, joka liittyi taajuusmuuttajan ohjaukseen.



Kuva.10 FBD ohjelmakaavio

Ohjelmointityökalussa on paljon toimintoja, joita en edes käyttänyt, kuten nk. Scope ikkuna, jonka avulla voi suorittaa signaalien graafista monitorointi samalla tavoin kuin oskilloskoopilla. Tuolle toiminnolle olisi voinut olla varmasti jotakin hyödyllistä käyttöä, mutta tyydyin käyttämään perustoimintoja kuten signaalien monitorointia numeerisessa tekstieditointikentissä, kuten kuvassa 11 on esitetty.



Kuva.11 Ohjelman monitorointi. Siniset, oranssit ja mustat kentät ovat oloarvoja

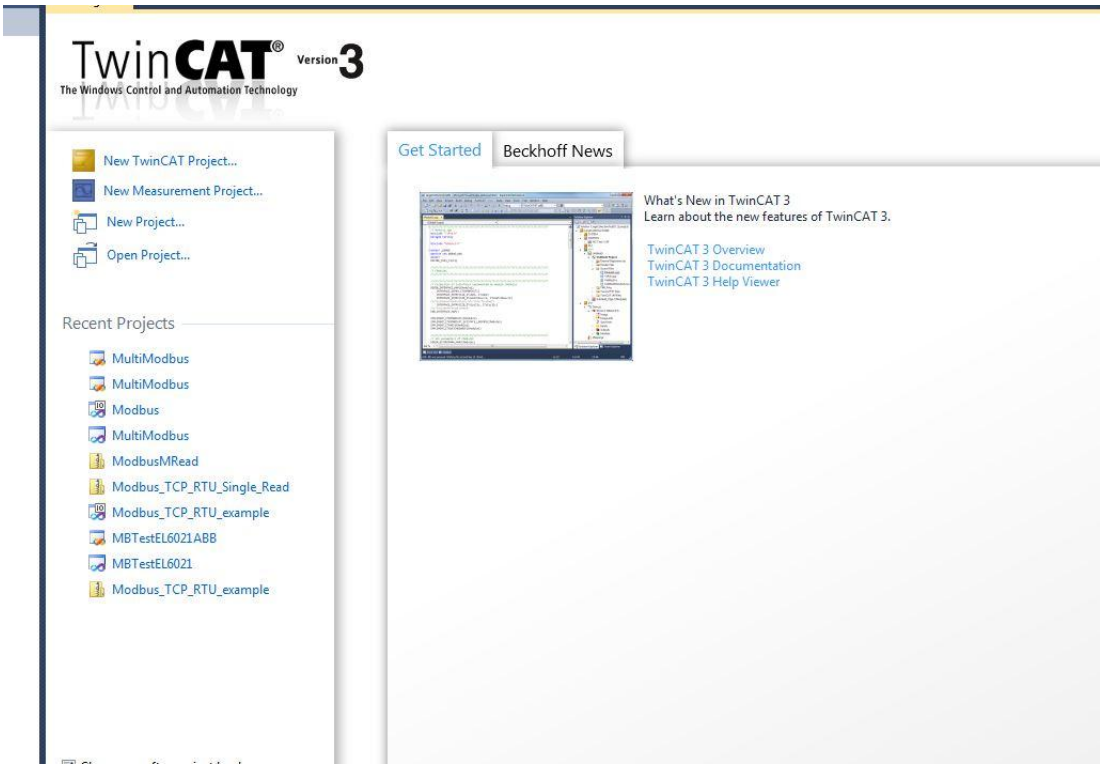
Eräs toiminto josta erityisesti pidin Beckhoffin TC3 ohjelmointityökalussa oli signaalien pakottaminen suoraan ruudulta. Erittäin helppo ja nopea käyttää.

Arvon pakottaminen tapahtui yksinkertaisesti klikkaamalla hiirellä kuvassa 11 näkyvää oloarvokenttä ja sen jälkeen syöttämällä uusi arvo, joka aktivoitiin pakotus toiminnolla. (ei näy kuvassa, painike on yläosan työkalupalkissa)
Aikaisempien ohjelmointityökalujen kanssa signaalien pakottaminen toiseen arvoon on ollut jokseenkin hankalaa ja vaivalloista, mutta TC3:lla toiminto oltiin tehty käyttäjäystävälliseksi.

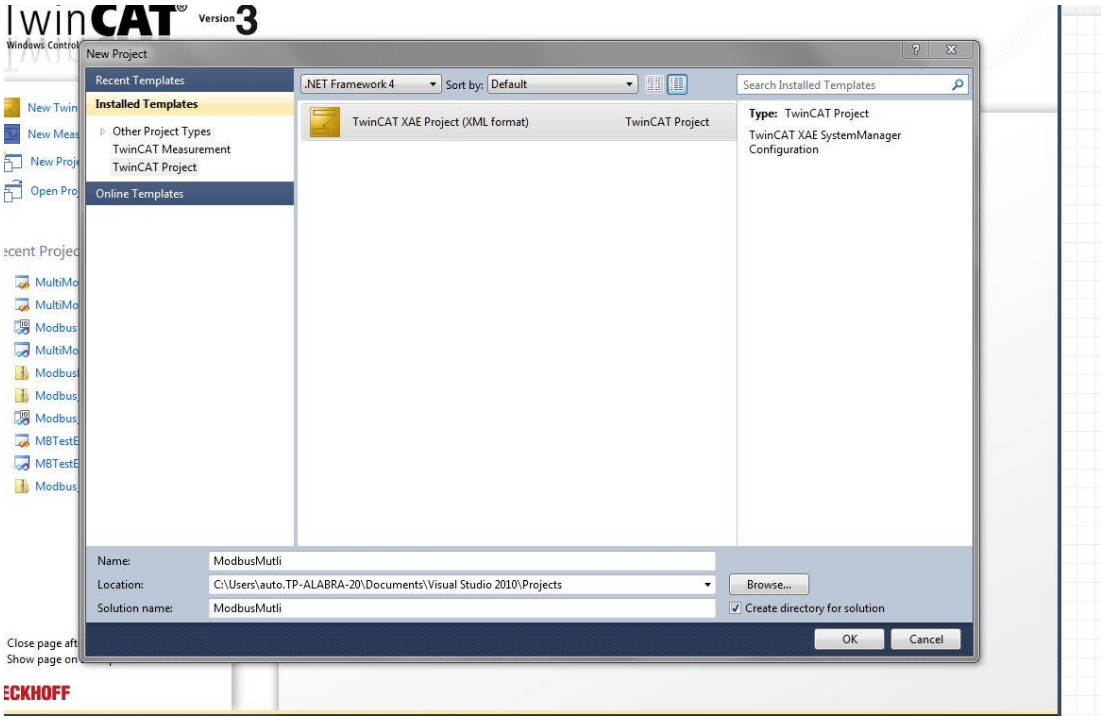
5.1.2 Visual studion projektin luominen

Itselläni oli käytössä ohjelmapohja, josta aloitin muokkaamaan omaa ohjelmaversiota, joten ihan alusta ei minun tarvinnut projektia alkaa tehdä. Puolivalmiin ohjelman käytössäkin on omat pienet hankaluutensa, esim. jo olemassa olevien signaalien ja muuttujatyyppien tutustumisessa. Tämä vie jonkin verran aikaa ennen kuin on saanut selvitettyä itselleen, miten signaalit on luotu ja missä niitä käytetään tai tarvitaan.

Seuraavissa kuvissa on lyhyt esimerkki uuden projektin luomisesta TC3:lla.

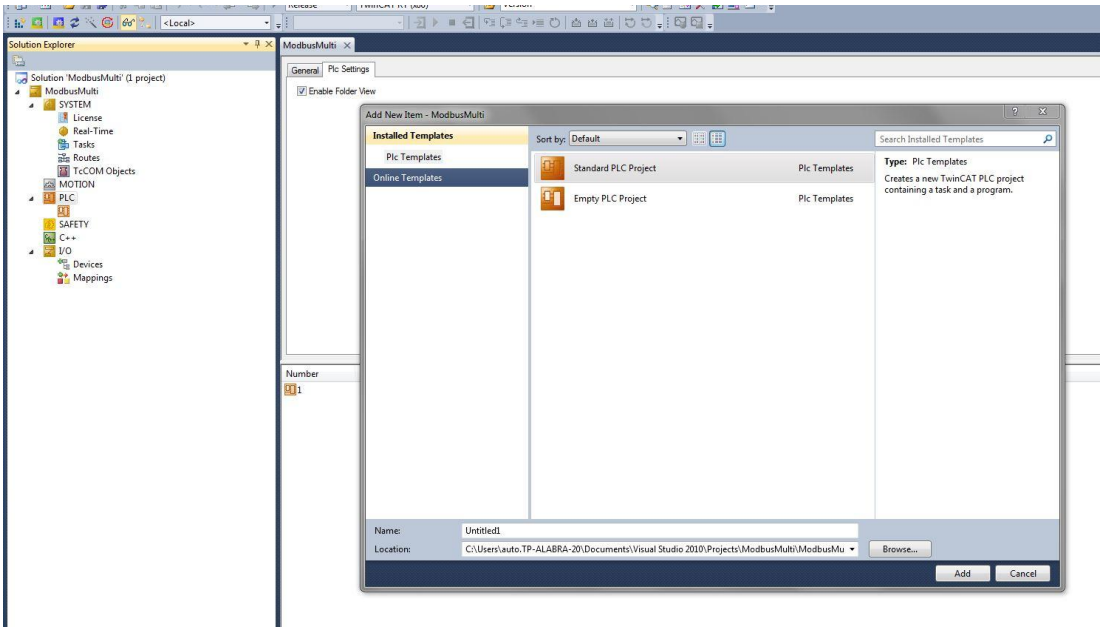


Kuva.12 TwinCat 3 aloitusvalikko. Tässä valitaan vasemmalla oleva New project..



Kuva.13 Projektin nimeäminen

Uuden projektin valinnan jälkeen avautuu yllä oleva näkymä. Annetaan projektille nimi ja klikataan OK



Kuva.14 PLC projektityypin valinta

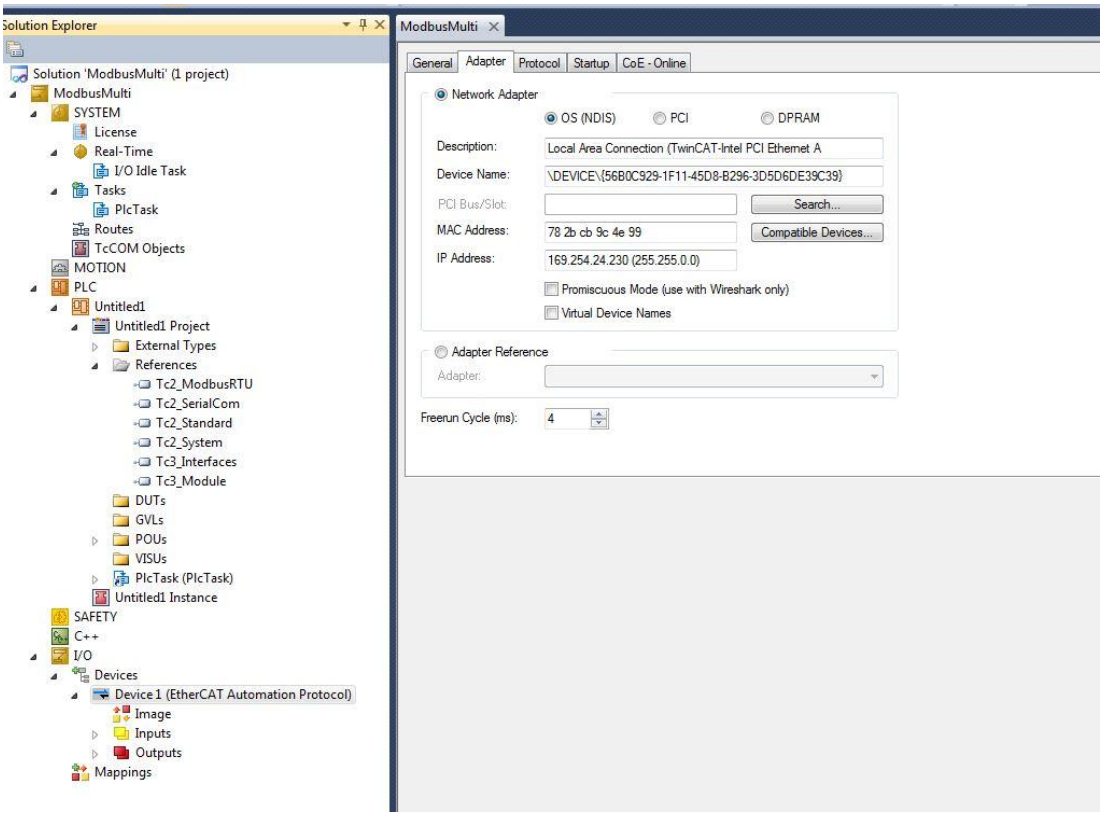
Projektin luonnin jälkeen, täytyy klikata PLC kuvaketta projektipuussa. Ohjelma kysyy minkäläisen projektin käyttäjä haluaa luoda, tässä valitaan Standard PLC Project, jonka myötä ohjelma lisää projektiin tiettyjä PLC kirjastoja. Kaikki Modbus ohjel-

moinnissa tarvittavat kirjastot eivät sisälly perusprojektiin, vaan niitä täytyy lisätä, josta myöhemmin tässä dokumentissa on oma selostus

Perusprojekti on nyt luotu, mutta tästä on vielä joitakin askeleita siihen pisteeseen, että varsinaista ohjelmointia voi alkaa suorittamaan.

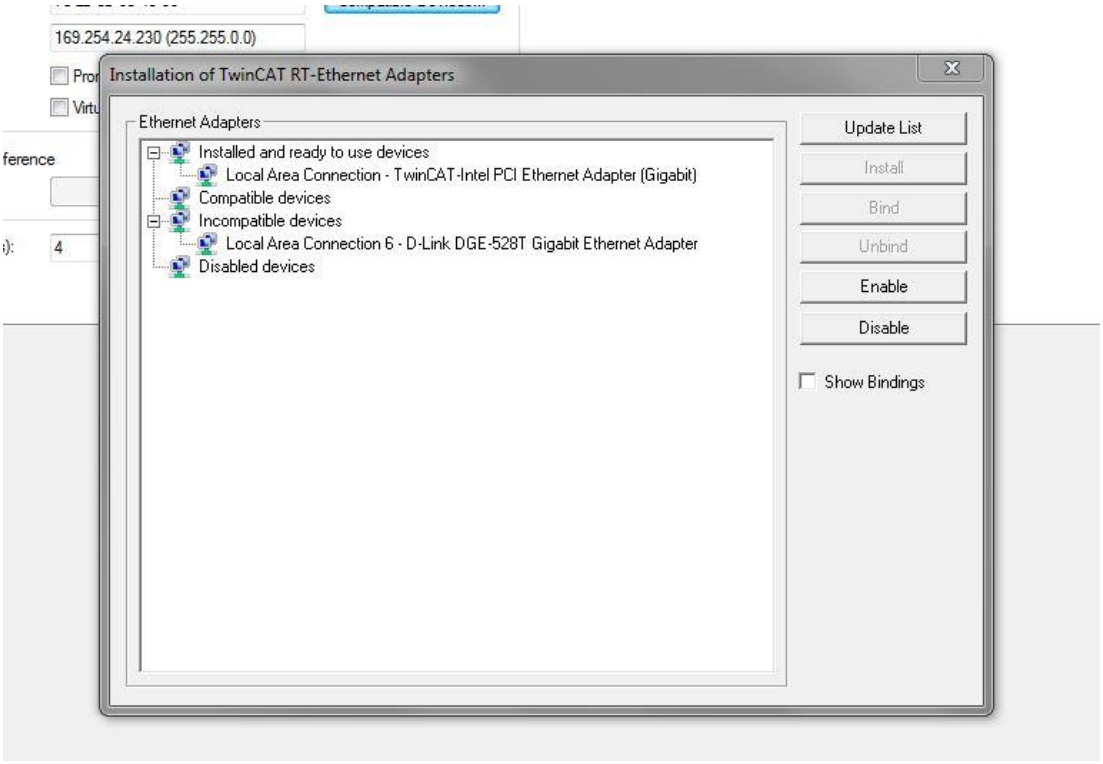
5.1.3 Laitekokoonpanon skannaus

Perusprojektin jälkeen kannattaa suorittaa laiteskannaus, jolla TC3 tutkii siihen liitetyn laitekokoonpanon. Skannaus suoritetaan klikkaamalla taikasauvan sisältävää painiketta ohjelman ylälaidassa oleavasta työkaluvalikosta Kuvassa 18
Seuraavissa kuvissa on skannauksen kuvia



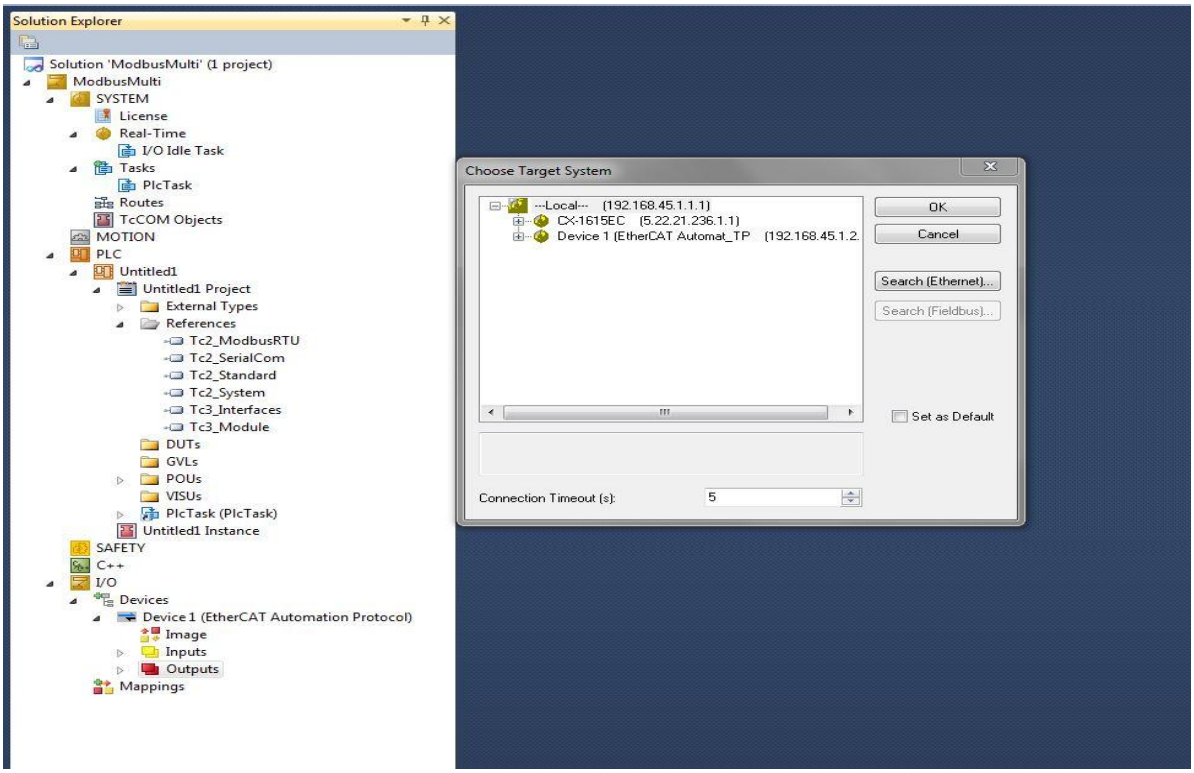
Kuva.15 Ensimmäinen skannaus

Jos kuvan 15 ikkunassa painaa Search toimintoa, listaa ohjelma kaikki löytyvät verkkokortit, myös ei-yhteensopivat kts. Kuva.16



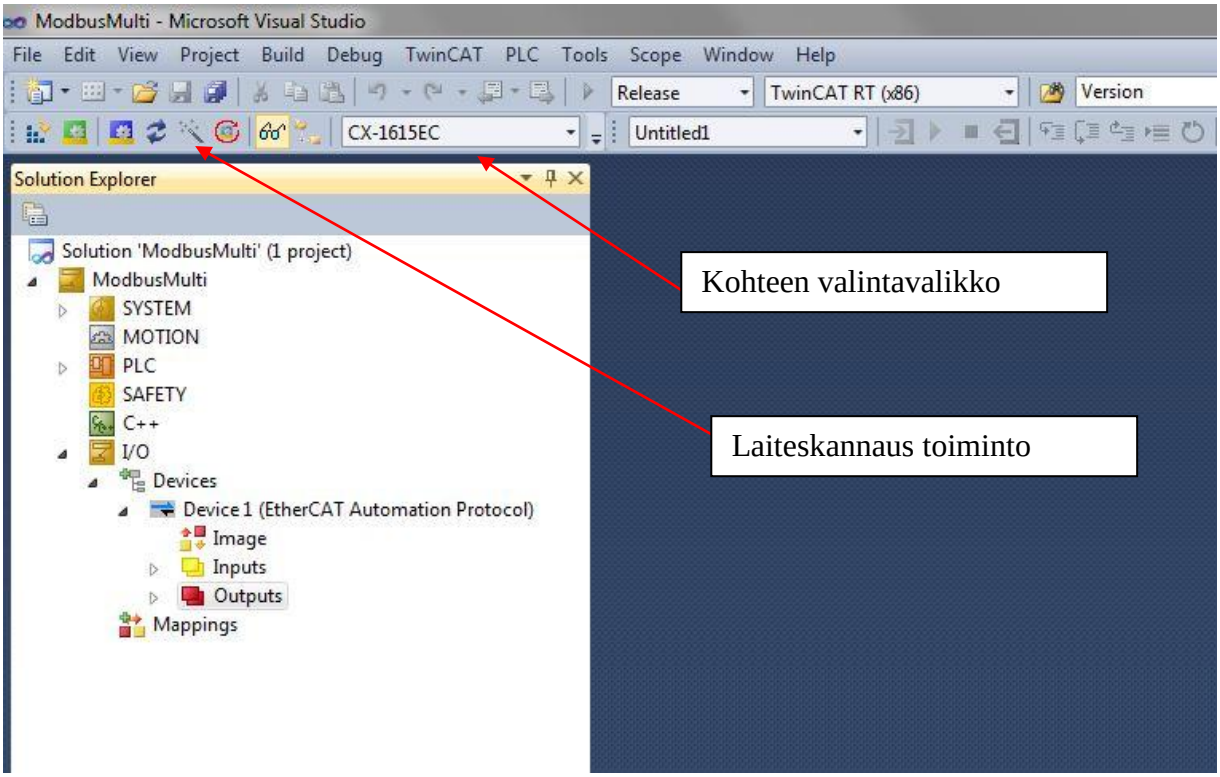
Kuva.16 Verkkokorttien listaus

Ensimmäisellä skannauksella ohjelma löysi vain PC:n verkkokortit. Näin voi käydä jos kohteeksi ei ole valittu ulkoista laitetta. Local eli paikallinen kohde on esim. testi-laitteisto 1, jossa ohjelma suoritetaan PC:ssä. Testilaitteisto 2, jossa mukana oli EL6021, tuossa laitteistossa on oma prosessori, joten kohteeksi täytyy valita ko. laite. Seuraavissa kuvissa on kohteen valinta esimerkki kohteen valinnasta.



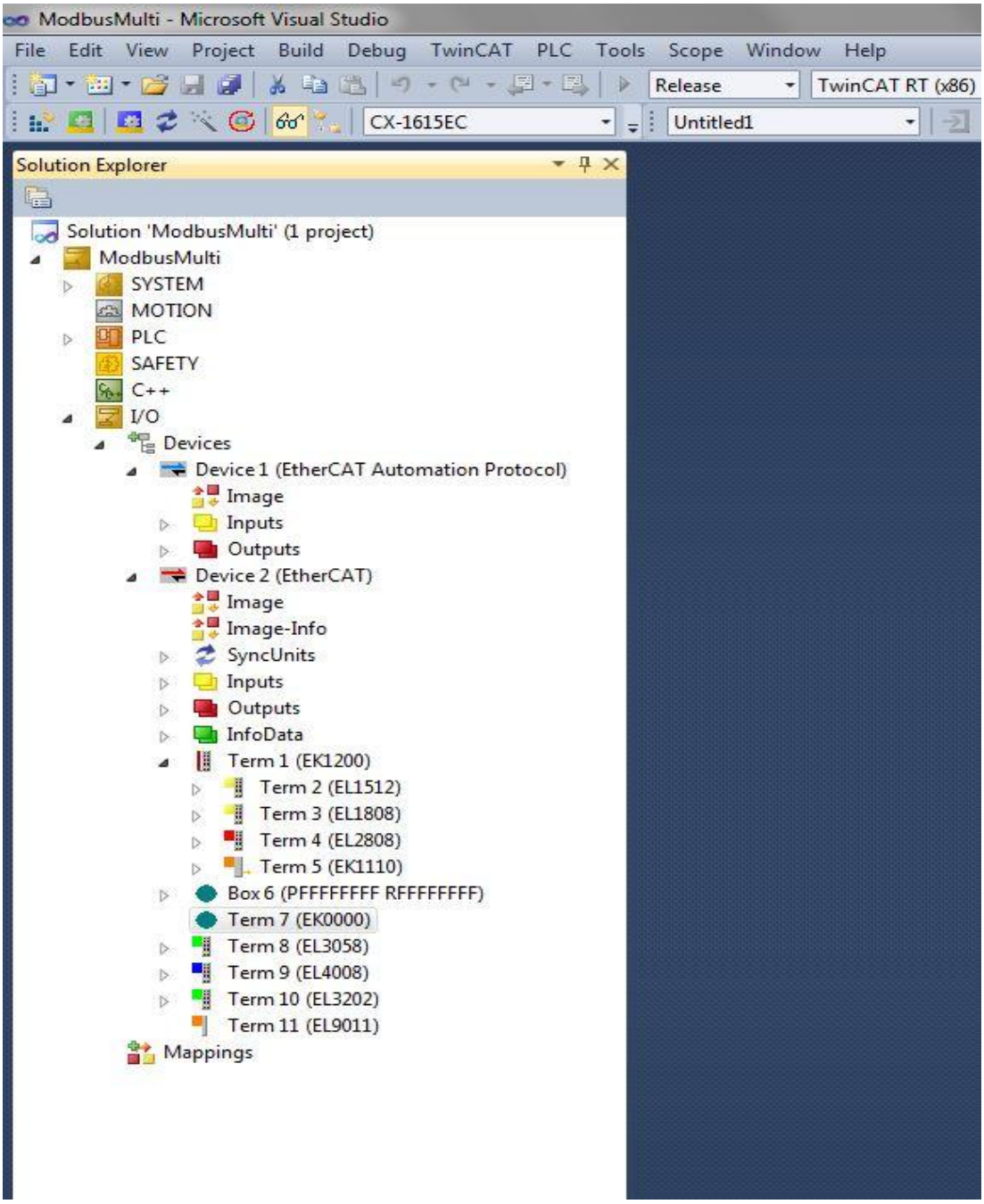
Kuva.17 Kohteen valinta

Kuva.18



Kuva.19 Valittu kohde

Oikean kohteen valinnan jälkeen skannaustulos näytti kuvan 20 mukaisen laitekokoonpanon.



Kuva.20 Laiteskannaus suoritettu.

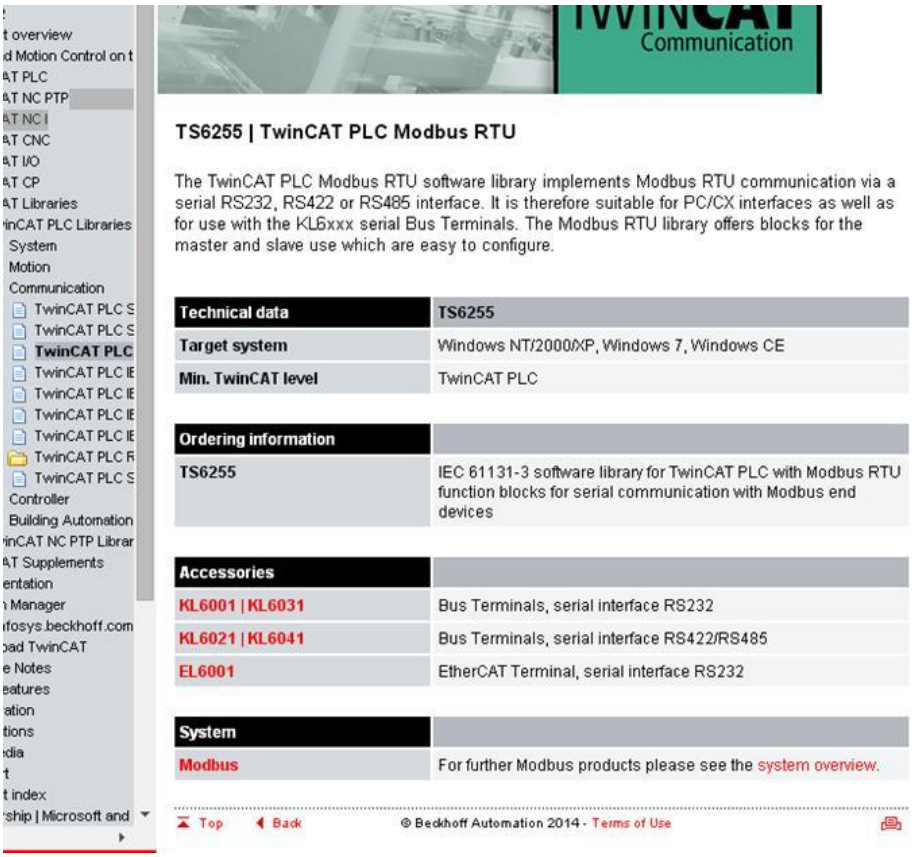
Valikoista ei löydy EL6021, se tässä poistettu, mutta sen paikka olisi juuri ennen Term 11(EL9011) päätekorttia

5.1.4 Tarvittavat ohjelmamoduulit

Projektin alussa minulla ei ollut juurikaan tietoa siitä minkälaisia toimilohkoja tai toimintoja ohjelmassa tarvittaisiin Modbus kommunikointia varten. Aikani selattuani Beckhoffin Internet-sivuja minulle selvisi, että joitakin lisäkirjastoja täytyi TC3:een ladata, jotta Modbus kommunikoinnin voisi toteuttaa. Mainitsin aikaisemmin tässä dokumentissa, että minulla oli vaikeuksia asentaa TF6340 TC3 Serial communication kirjasto PC:lle. Linkki tuonne kirjastoon on TF6255 Modbus RTU sivun kautta. Kun avaa TF6255 dokumentoinnin ja alkaa tutkia KL6configuration toimilohkoa, niin huomaakin, ettei se kuulukaan TF6255 kirjastoon vaan se on osa TF6340 kirjaston alla.

Itselleni kävi aluksi sellainen virhe, että latasin TF6255 kirjaston Beckhoffin sivuilta. Jälkeenpäin etsiessäni noita KL6Configuration toimintoja huomasin, ettei TF6255 sisälläkään ko. toimintoja. Sen jälkeen latasin asennusohjelman TF6340 toimilohkokirjastolle, jonka asentamisen kanssa minulla paljon vaikeuksia. Asennus ei onnistunutkaan ilman Beckhoffin teknisen tuen apua.

Pienenä negatiivisena palautteena Beckhoffille antaisin tarpeen internet sivujen selkeyttämiselle, mainittakoon esimerkkinä TF6255 ja TF6340 toimilohkokirjastojen selostukset ja linkitys internet-sivuilla. Nykyisessä muodossa kirjastojen yhteenkuuluvuus ei ole kovin selvä.
Sivut saisivat olla siinä mielessä selkeämmät, että hakusana ei aina löydä tarvittavia asioita, jos sattuu olemaan väärässä kohtaa sivuja. Toisaalta sivuilta puuttuu myös informaatiota kuten esim. kuvassa Kuva.21, ei ole mainintaa EL6021 kortista. Samoin TF6255 kirjastoon voisi olla suora linkki, kuten KL6021 kohdalla on. Sivut tuntuvat puutteellisilta ja tiedon etsiminen on vähän hidasta.

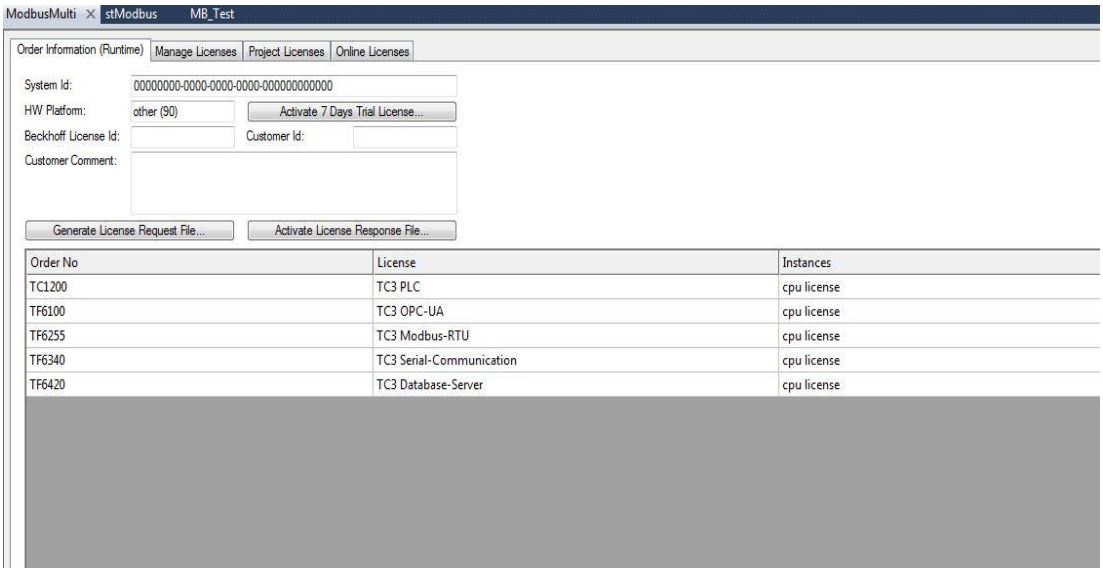


Kuva.21 Beckhoffin web-sivu

Modbus ohjelman luomista varten lisäosia täytyi ladata Beckhoff Automation web-sivustoilta. Tarvittavat lisäosiokirjastot olivat:

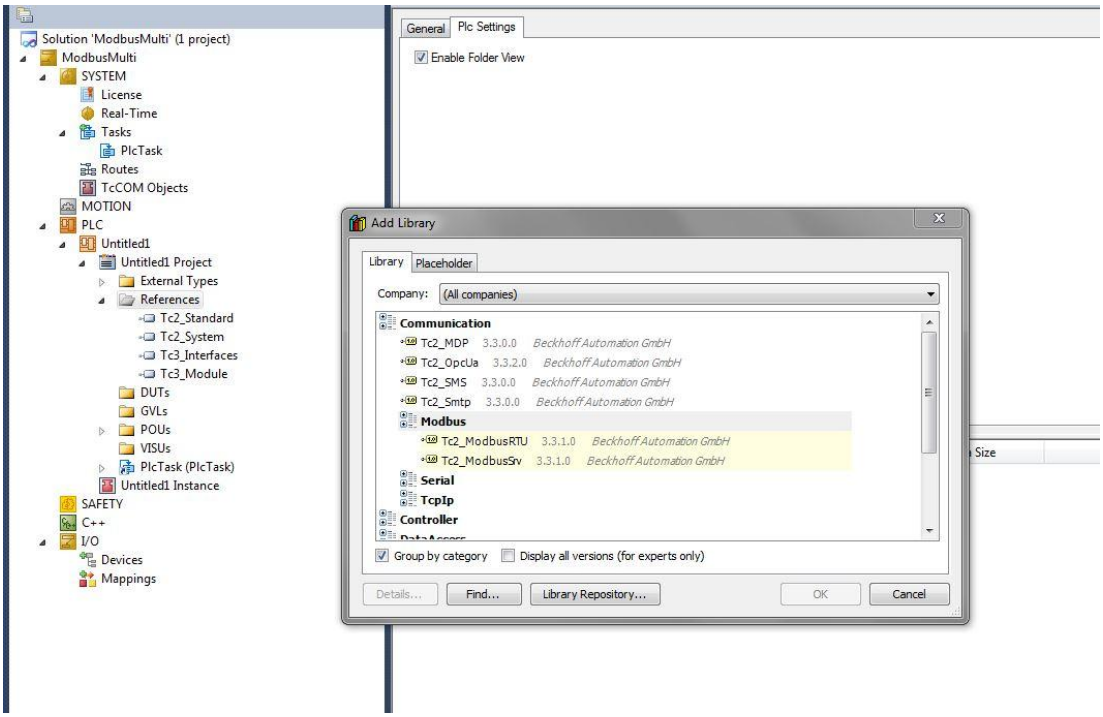
- TF6340 TC3 Serial communication
- TF6255 TC3 Modbus RTU

Lisäosat voi ladata Beckhoffin web-sivuilta;
http://www.beckhoff.com/english.asp?twincat/te1000.htm
(Beckhoff PLCLib, 2014)
Sivuilta löytyy myös tarvittaessa ohjeita PLC kirjastoista ja niiden asentamisesta. Kuvassa 21 TC3 asennetut ohjelmakirjastot TF6255 ja TF6340 olivat RTU:lle tärkeät lisäosat

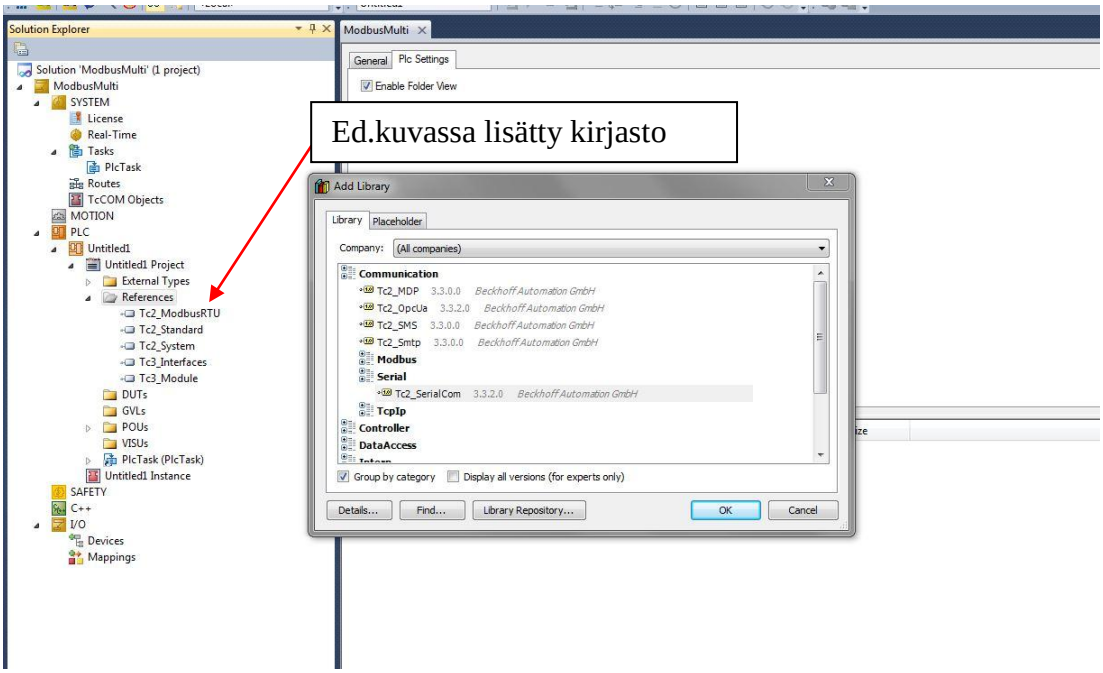


Kuva.22 TC3:een asennetut ohjelmakirjastot

Vaikka lisäosat näkyvätkin jo lisenssivalikossa, täytyy noiden kirjastojen sisältämät toimilohkot ja muut toiminnot vielä lisätä PLC kirjastoihin. Se tapahtuu seuraavien kuvien mukaisesti.

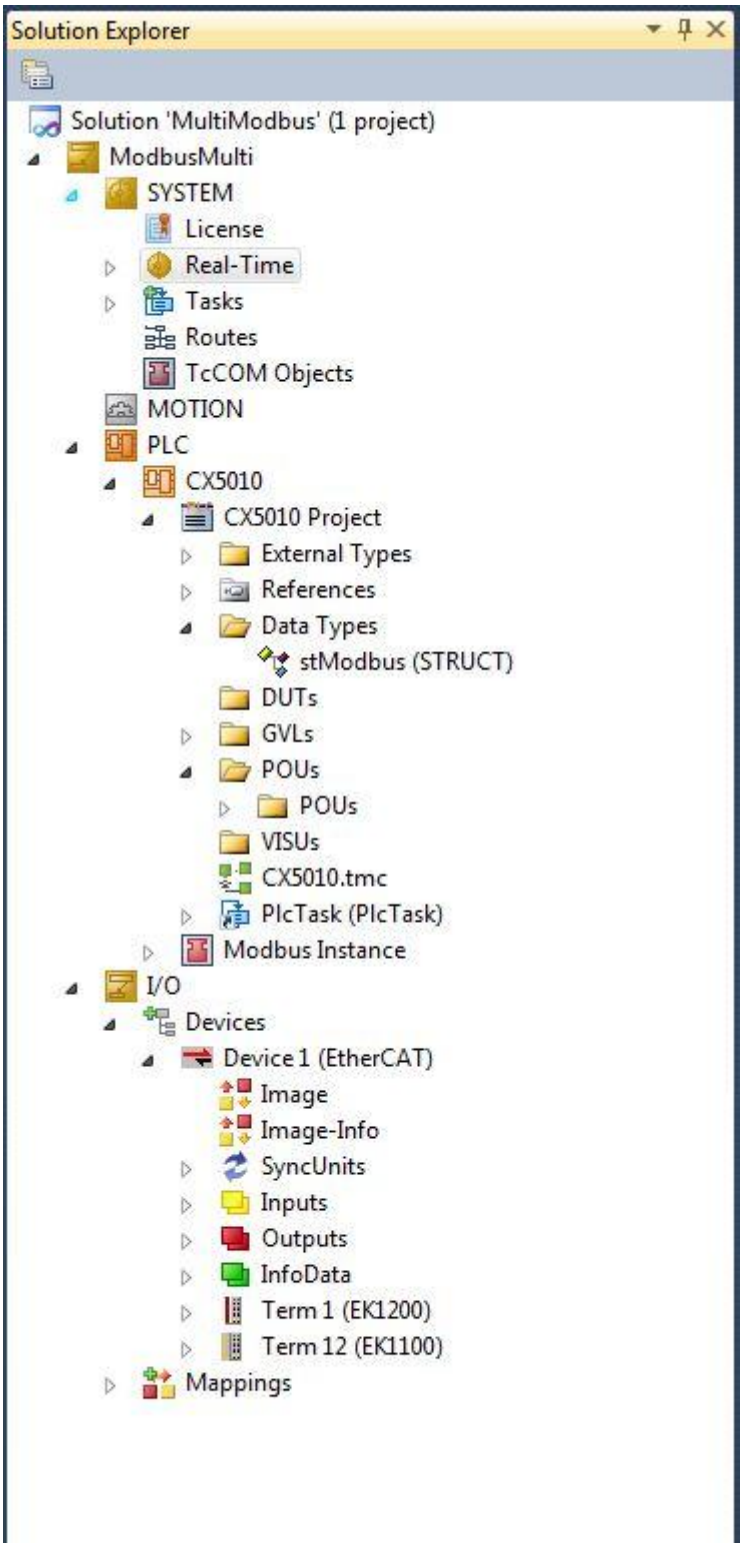


Kuva.23 Modbus toimilohkokirjaston lisäys



Kuva.24 Sarjaliikenne toimilohkojen lisäys.

Projektipuun tulisi näyttää edellä mainittujen suoritusten jälkeen suurin piirtein kuvan 24 mukaiselta.



Kuva.25 Projektipuu esimerkki

5.1.5 KL6021 Kommunikointimoduulin alustus (configurointi)

Kohdassa 3.1.1 on kuvattu KL6021 kortin alustusta. Kertauksen vuoksi esitetään tässä kohdassa uudelleen KL6021 kortin alustus.

Kortin alustus tapahtuu Modbus kommunikointiohjelman alussa sitten, että KL6Configuration toimilohko suoritetaan ohjelman alussa vain kerran. Tähän tarkoitukseen käytetään ohjelmasta saatavaa sisäistä signaalia, joka on päällä vain ensimmäisen ohjelmakierron ajan. Seuraavassa ohjelmalistauksessa on KL6021 kortin alustus;

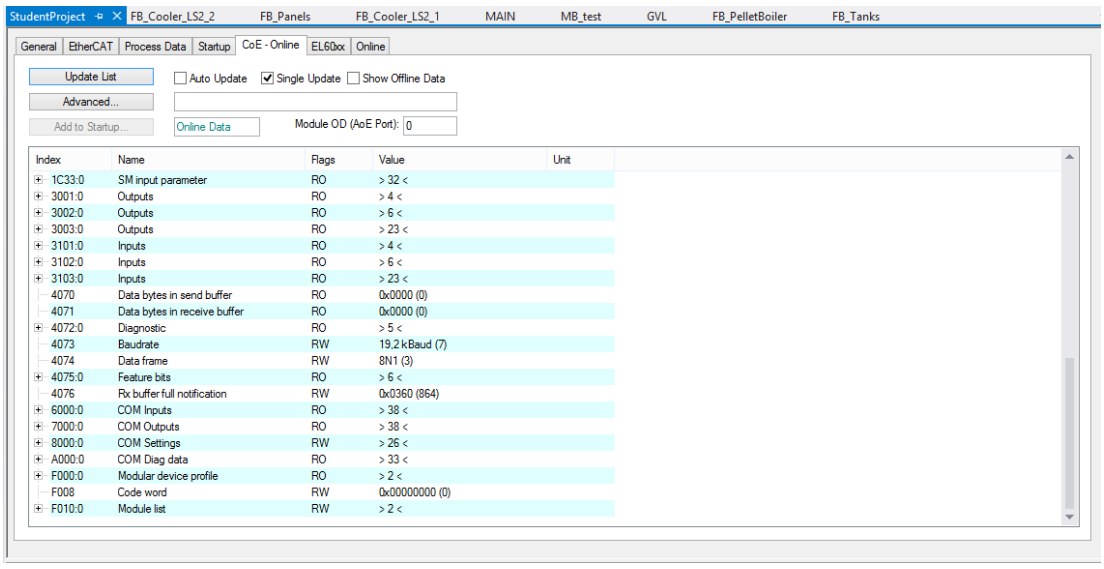
```
IF _TaskInfo[1].FirstCycle THEN (Tieto siitä, että ohjelma kierto on ensimmäinen)
        bConfig:=TRUE; (alustetaan bitti päälle)
END_IF

KL6configuration1(
        Execute:=bConfig , (Suorita alustus)
        Mode:=SERIALLINEMODE_KL6_22B_STANDARD, (tässä voisi olla myös
        SERIALLINEMODE_KL6_5B_STANDARD, riippuu kortin tyypistä)
        Baudrate:=19200 , (Baudinopeus)
        NoDatabits:=8 , (databittien lukumäärä)
        Parity:=PARITY_EVEN , (Pariteetti)
        Stopbits:=1 , (Stop bittien määrä)
        Handshake:=RS485_HALFDUPLEX , (Kättely)
        ContinousMode:=TRUE , (Tämä bitti poistaa viiveet viestien välistä, esim. Vacon ei toiminut jos viestissä oli viiveitä)
        pComIn:=ADR(MB.InData) , (Muuttuja jonka kautta viestit luetaan sisään)
        pComOut:=ADR(MB.OutData) , (Muuttuja jonka kautta viestit lähetetään)
        SizeComIn:=SIZEOF(MB) , (sarjaliikenne laittieston käyttämän process imagen koko)
        Done=>ConfigDone , (alustuksen suorituksen tarkistus)
        Busy=> ConfigBusy, (Config tila päällä, kun bitti on päällä, poistuu kun config suoritettu)
        Error=> , (Virhe)
        ErrorId=> ); (Virheen koodi)

IF ConfigDone=TRUE THEN
        ConfigFinished:=ConfigDone; (kun alustus suoritettu ConfigFinished saa arvon TRUE)
        bConfig:= FALSE; (Resetoidaan bConfig signaali)
END_IF
```

KLConfiguration toimilohkon englanninkielinen selotus löytyy Beckhoffin websivustolta;
http://infosys.beckhoff.com/english.php?content=../content/1033/tf6255_tc3_modbus_rtu/html/tcplclibmodbusrtu_overview.htm&id=
(Beckhoff)

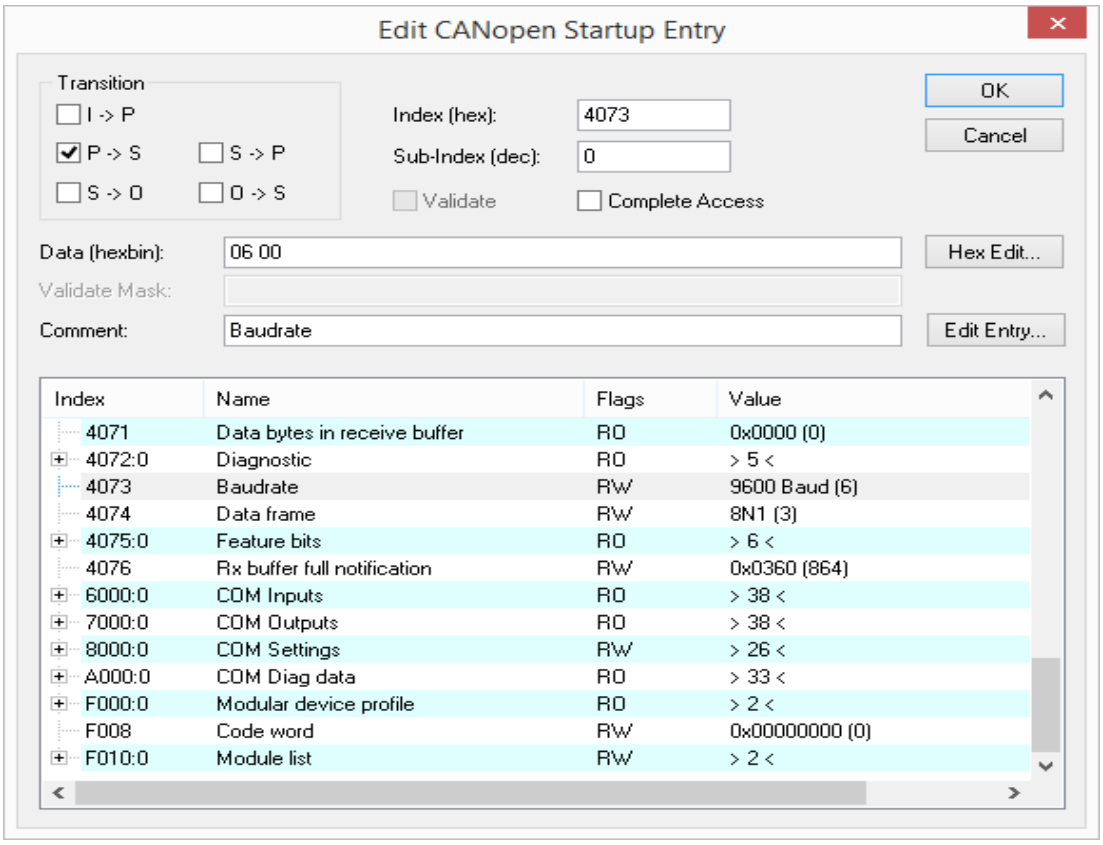
5.1.6 EL6021 Kommunikointimoduulin alustus (configurointi)

EL6021 kortin alustus tapahtuu kortin sisäisiä parametreja muuttamalla. Oletin aluksi, että kortti alustettaisiin samalla tavalla kuin KL6021 ja ensin alkuun yritinkin alustaa kuin KL-kortin. Myöhemmin Teppo Lepistö Beckhoffilta opasti Team Viewer ohjelman välityksellä EL6021 kortin oikean alustamisen. EL6021 ei vaadi erillistä alustustoimintaa, koska EL6021 kortin alustus suoritetaan korttiparametreja käyttämällä. KL6021 toimilohko sai jäädä kommenttilausekkeeksi ohjelmaan mukaan, eli KL6Configuration otettiin pois ohjelmasuorituksesta.

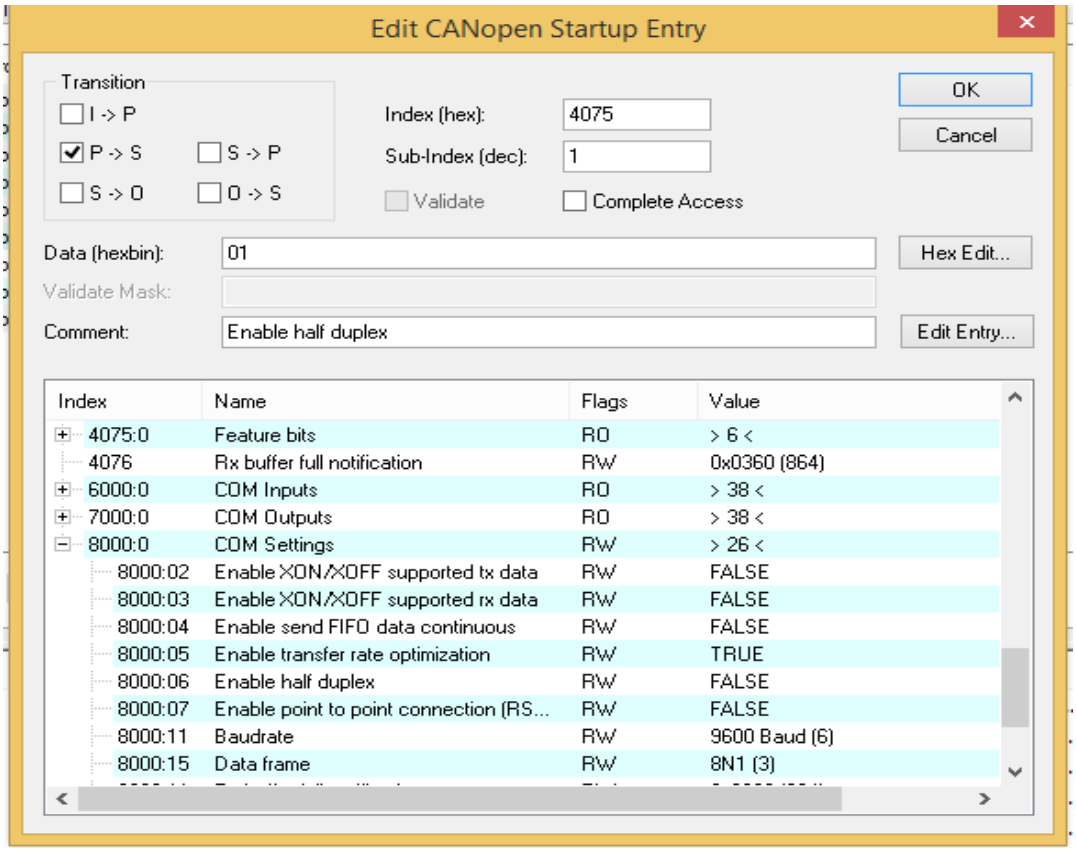
EL6021 korrtiparametreja on esitetty seuraavissa kuvissa.



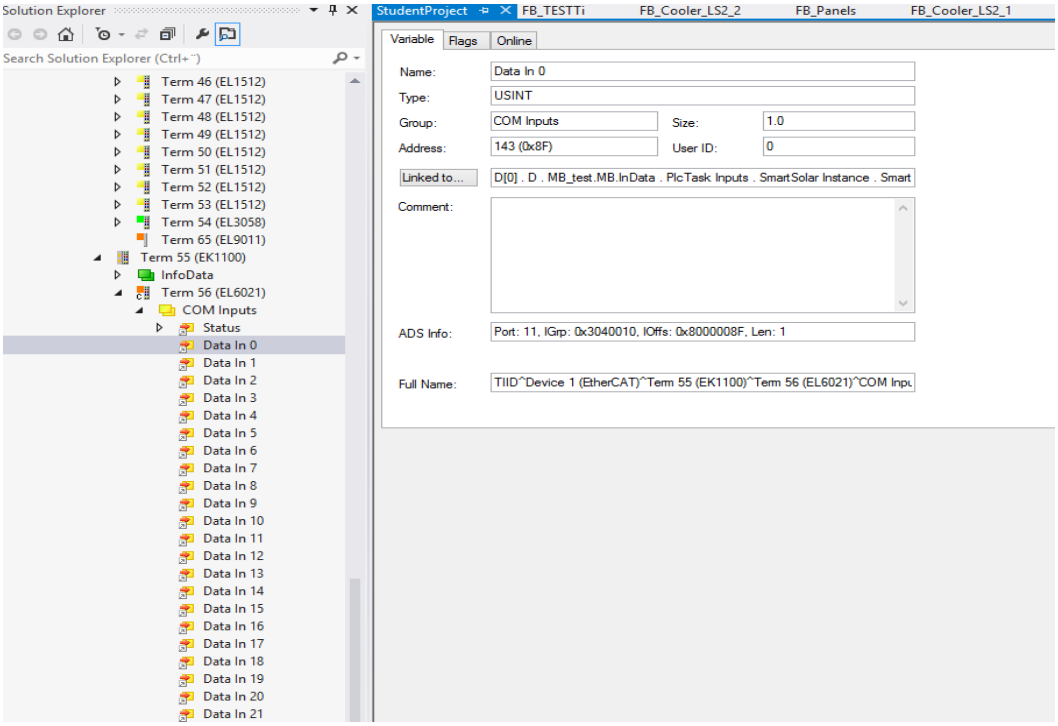
Kuva.26 Kortin alustusparametrit 4073 ja 4074 täytyi muuttaa.



Kuva.27 Korttiparametreja 4073 ja 4074 samat kuin ed.kuvassa Tässä arvojen asetus valikko.



Kuva.28 8000:06 Half Dublex valintaparametri



Kuva.29 Kortin signaalien linkitys ohjelmaan (tämä tehdään molemmilla 6021 korteilla samalla tavalla)

5.1.7 Ohjelman kuvaus

Ohjelmalla testattiin kahta tapaa kommunikoida taajuusmuuttajan kanssa. Ensimmäisessä testausohjelmassa tutkittiin kommunikointitapaa, jossa globaalitaulukkomuuttajilla luettiin dataa taajuusmuuttajasta ja kirjoitettiin dataa taajuusmuuttajaan. Tätä toimintoa varten täytyi ohjelmaan luoda tietuemuuttujataulukko. Taulukoon syötetyt Modbusrekisterien osoitteet käytiin läpi yksi kerrallaan laskurin osoittaman osoitteen perusteella.

Taulukon sisäinen osoitteen hallinta suoritettiin laskurin avulla, siten että laskuri osoitti tietuetaulukosta luettavan Modbusrekisterin osoitteen, josta data luettiin. Laskurin arvolla 1 luettiin siis ensimmäinen tietuetaulukon osoite jne. Laskuri askelsi globaalimuuttujissa määritellyn arvon (iArrayLenght) verran, eli 8:aan asti. Kun viimeinen luettava taulukon osoite oli luettu, siirryttiin dataa kirjoittavan taulukon käsittelyyn. Samaa tietuerakennetta (stModbus) pystyttiin käyttämään sekä lukemisessa

että kirjoittamisessa. Luettava taulukko oli 8 muuttujaa käsittävä rakenne [1..8] ja kirjoitettava 2 muuttujaa käsittävä [0..1]

Toinen testattava kommunikointitapa oli lukea ja kirjoittaa data ns. pakettina. Toisin sanoen Beckhoffin toimilohkoilla ReadRegs (Modbus FC 3 read analog registers) ja WriteRegs (Modbus FC16 write multiple registers) suoritettiin kommunikointi. Datan luku toiminnossa toimilohkolle annetaan osoite, mistä datan lukeminen alkaa. Toimilohkon parametreissa määritellään myös, montako sanaa eteenpäin aloitusosoitteesta luetaan. Luettavien sanojen määrä riippuu kortin tyypistä. Käytössä ollut KL6021 ei pystynyt lukemaan kuin enintään 2 sanaa. EL6021 pystyi suoriutumaan 11 sanasta. Datan kirjoitus tapahtuu samoin kuin lukeminen, käyttäen WriteRegs toimilohkoa. Kirjoitettavien sanojen määrä riippuu samalla tavalla korttityypistä kuin datan lukemisessa. Testissä sanoja kirjoitettiin vain kolme.

Mainittavaa on myös että jälkimmäistä kommunikointitapaa ei suoritettu kuin Vaconin taajuusmuuttajan kanssa laboratoriossa. ABB:n taajuusmuuttaja oli jo toiminnassa joten sillä ei testejä enää suoritettu.

5.1.8 Struct rakenne

Globaalille muuttujalle stModbus täytyi määritellä tietuerakenne, joka oli seuraavanlain;

```
TYPE stModbus : tietueen nimi ja alla on tietueessa käytettävät kentät
STRUCT
    FunctionType    :INT;
    Address         :INT;
    Name            :STRING;
    value           :INT;
    Error           :BOOL;
    ErrorID         :INT;
END_STRUCT
END_TYPE
```

Ohjelman listauksessa näkyy tietueen käyttö ohjelmassa. Tätä rakennetta käytettiin siis vain ohjelmassa missä signaalit käsiteltiin yksi kerrallaan. Ohjelman listauksen yhteydessä on tarkempi selostus.

5.1.9 Modbus muuttujat

Modbus kommunikointia varten täytyy luoda seuraavat rajapintamuuttujat.

```
TYPE KL6DataIn22B (Tämä on tärkeä tietue. Tämän avulla tulosignaalit saadaan tuotua ohjelmaan)
STRUCT
        Status: WORD;
        D: ARRAY[0..21] OF BYTE; (Tavujen määrä riippuu käytettävästä kommunikointkortista)
END_STRUCT
END_TYPE

TYPE KL6DataOut22B (Tämä on tärkeä tietue. Tämän avulla lähtösignaalit saadaan lähetettyä)
STRUCT
        Ctrl: WORD;
        D: ARRAY[0..21] OF BYTE; (Tavujen määrä riippuu käytettävästä kommunikointkortista)

END_STRUCT
END_TYPE
```

5.1.10 Globaalimuuttujat

Kommunikointi varten täytyi luoda globaalimuuttujalista, joka oli seuraavanlainen;

```
VAR_GLOBAL CONSTANT
                iArrayLenght      :INT:=8; (Luettavan taulukon pituus, signaalien määrä)
                iWriteArrayLength: INT := 1; (kirjoitettavan taulukon pituus, signaalien määrä)
        END_VAR
VAR_GLOBAL
        DriveDataIn :WORD;      (word tyypinen muuttuja)
        DriveFaulted: BOOL;     (Boolean tyyppinen muuttuja)

stModbusComm: ARRAY [1..iArrayLenght] OF stModbus:=[ (Taulukkotyyppinen muuttuja)
        (FunctionType:=4,Address:=2100,Name:='Status Word '), (Ensimmäinen luettava signaali)
```

```
        (FunctionType:=4,Address:=2101,Name:='Aux Status Word '),
        (FunctionType:=4,Address:=2102,Name:='FB Actual Speed 0..10000 '),
        (FunctionType:=4,Address:=2103,Name:='Output Frequency Hz '),
        (FunctionType:=4,Address:=2104,Name:= 'Motor Speed rpm'),
        (FunctionType:=4,Address:=2105,Name:='Motor Current A '),
        (FunctionType:=4,Address:=2106,Name:='Motor Torque %'),
        (FunctionType:=4,Address:=2107,Name:='Motor Power % ')];
```

Tarvittaessa tähän taulukkoon lisätään signaaleja, jos haluaa lukea enemmän kuin yllä olevat.

FunctionType ei ole Modbus toimintakoodi, nimi vähän erheellinen, parempi nimi olisi RegisterType

Signaali nimi tulee ’ ’ kirjoitetusta tekstistä. Nimeä ei lueta väylän kautta vaan annetaan taulukkoon

```
stModbusWrite: ARRAY [0..iWriteArrayLength] OF stModbus:=[ (kirjoitettavat signaalit)
        (FunctionType := 4, Address := 2000, Name := 'Control Word'),
        (FunctionType := 4, Address := 2002, Name := 'Speed Ref  ')];

        StatusWord: INT;  (Ohjelmassa käytettäviä yksittäisiä muuttujia tästä eteenpäin)
        DriveInFBControl: BOOL;
        SpeedReference: INT;
        AuxStatusWord: INT;
        StartCmd: BOOL;
        Reset: BOOL;
        ResetCmd: BOOL;
        Start: BOOL;
END_VAR
```

ABB:n taajuusmuuttajaa varten globaali muuttajalista näytti kuvan 30 mukaiselta. Muut määritelmät pysyivät samoina.

```
stModbusComm: ARRAY [1..iArrayLenght] OF stModbus := [
    (FunctionType:=4,Address:=0003,Name:='Status Word '),
    (FunctionType:=4,Address:=0303,Name:='Aux Status Word '),
    (FunctionType:=4,Address:=0101,Name:='FB Actual Speed 0..10000 '),
    (FunctionType:=4,Address:=0102,Name:='Output Frequency Hz '),
    (FunctionType:=4,Address:=0101,Name:= 'Motor Speed rpm'),
    (FunctionType:=4,Address:=0103,Name:='Motor Current A '),
    (FunctionType:=4,Address:=0104,Name:='Motor Torque %'),
    (FunctionType:=4,Address:=0106,Name:='Motor Power % ')];

stModbusWrite: ARRAY [0..iWriteArrayLength] OF stModbus:=[
    (FunctionType := 4, Address := 0000, Name := 'Control Word'),
    (FunctionType := 4, Address := 0001, Name := 'Speed Ref  ')];
```

Kuva.30 ABB:n taajuusmuuttajalle luotu globaali taulukkomuuttuja

5.1.11 Ohjelmalistaukset

Taulukon avulla suoritettava kommunikointi:

Ohjelmalistauksessa on Vaconin taajuusmuuttajan osoitteet. ABB:n taajuusmuuttajalle ohjelma on muuten saman mutta CW (control word) kirjoitus poikkeaa Vaconista ja taulukon osoitteet ovat erit. Control Wordin käytöstä enemmän tarkempi kuvaus Testaus osiossa. Ohjelmalistaus taulukkomuuttujilla toimivalla kommunikoinilla on esitetty liitteessä 2.

Usean rekisterin käsittely samanaikaisesti:

Tästä alkaa ns. multiread ja multiwrite ohjelman listaus. Tämä ohjelma osia ei siis käyttää taulukkorakennetta joka on määritelty globaalimuuttujissa, vaan kaikki tiedot luetaan yhteen word tyyppiseen taulukkoon. Ohjelmalistaus on lyhyempi. Ohjelmalistaus tätä toimintoa varten on esitetty liitteessä 4

Local muuttujiin täytyi tehdä seuraavat määritelmät

iData2	: ARRAY [1..14] OF WORD; datan luku muuttuja
iDataOut1	: ARRAY [0..2] OF WORD; Datan lähetysmuuttuja

6 TESTAUS

6.1 BECKHOFF-VACON TESTAUS

Vacon NXS:n kanssa pystyin testaamaan molemmat kortit sekä KL- ja EL6021. Tämä olikin hyvä, koska itse hybridijärjetelmän kanssa ei ollut mahdollisuuksia testauksiin. ohjelman piti käyttöönotossa olla jo jotakuinkin toimiva.

Molempien korttien ja testilaitteistojen testaus Vacon taajuusmuuttajan kanssa oli lähes identtinen. Laitteiston vaihtuminen ei taajuusmuuttajan päässä aiheuttanut mitään muutoksia. Kaikki tarvittavat muutokset koski ainoastaan Beckhoff ohjelmistoa ja laitteistoa.

6.1.1 KL6021 Moduuli

Ensimmäinen testi, jonka suoritin Beckhoffin testilaitteistolla 1 ja Vacon taajuusmuuttajalla, oli yksinkertaisesti lukea yksi signaali taajuusmuuttajasta. Beckhoffin PLC ohjelma suoritus tapahtui PC:ssä, koska laitteistossa ei ollut omaa suoritinta (CPU).

Aivan ensimmäisissä testeissä minulta puuttui kortin alustuslohko, joten tuo alkeellinen ohjelma ei missään tapauksessa olisi toiminut. Beckhoffin sivuilta löytämäni alustuslohkot ja Beckhoffin teknisen tuen avulla sain asennusohjelman suoritettua, joka asentaa toimilohkot toimilohkokirjastoon.

Uusi yritys alustuslohkon kanssa, edelleen yrittäen lukea vain taajuusmuuttajan tilasanaa, Vaconin osoite 2101 FB Status Word (SW). Tilasana on siitä hyvä signaali lukea, että sen tila on helppo muuttaa käynnistämällä ja pysäyttämällä taajuusmuuttaja sen ohjauspaneelista.

Monitoroin tilasanaa Beckhoffin logiikassa ja samalla monitoroin taajuusmuuttajan ohjauspaneelilla OPT C2 kortin kenttäväylältä vastaanottamia sanoja. Monitorointi

näyttö on selostettu Vacon NX OPT C2 käsikirjassa sivulla 17. (VACON OPTC2 , 2012)
Aluksi vain virheellisten sanomien määrä kasvoi, joten väylällä liikkui jotain sanomia, mutta taajuusmuuttaja näki ne virheellisinä. Tarkistin johdotukset useampaan kertaan. Modbus-osoitteet, kortin parametroinin, kaikki näytti olevan OK. Päätin kokeilla alustuslohkossa olevaa Continuous bittiä, ja asetin sen päälle. Väylä lähti toimimaan tuon asetuksen jälkeen. Enää ei virheellisten sanomien määrä kasvanut, sen sijaan hyväksyttyjen sanomien laskurin arvo kasvoi.

Yksinkertainen kommunikointitesti toimi. Seuraava vaihe oli saada ohjelma lukemaan kaikki globaali muuttujataulukossa olevat arvot. Ensin alkuun lukeminen taulukon kanssa ei toiminut. Virheeksi osoittautui tietueen alustuksen puuttuminen. Olin kopioinut tietueen Beckhoffin minulle lähettämästä ohjelmasta, mutta en ollut luonut tai alustanut tarvittavaa signaalia stModbusComm, eli ohjelmassa oli pelkkä nimi, millä ei ollut mitään toimintoa. TC3 antoi aina ohjelmaa käännettäessä virheen, Init missing, ja jonkin aikaa kesti löytää puuttuva Init signaali ohjelmasta.

Saatuani virheet poistettua ohjelmakäännöksestä, kohtasin seuraavan ongelman. Ohjelma luki yhden signaalin, tilasanan sisään, mutta kun piti lukea koko taulukko, niin kommunikointi loppui. Syy tähän selvisi melko nopeasti, mutta korjaus kesti jonkin aikaa. Syy kommunikoinnin loppumiseen oli ohjelmasekvenssissä. Ohjelman piti toimia niin, että jokaisella ohjelmakierrolla luetaan sen hetkisen laskurin osoittaman taulukkopaikan osoite taajuusmuuttajasta ja odotetaan niin kauan kuin ReadRegs-lohkon busy tieto on päällä. Kun busy tieto on poistunut, suoritetaan seuraava askellus, eli laskurin arvoa kasvatetaan yhdellä ja luetaan seuraava taulukkopaikan osoite. Liitteessä 1 on vuokaavio sekvenssin suoritukselle.

Tuon toiminnan tahdistuksessa minulla oli pitkään vaikeuksia. Yritin soveltaa kaikenlaisia ehtoja ilman hyväksyttävää tulosta. Olen tehnyt paljon ohjelmointia FBD ohjelmakaaviolla ja koetin soveltaa sen sääntöjä tähän. Ohjelma oli tehty lausekielellä ja siinä ei ohjelmakiertosäännöt pätenytkään samalla tavalla kuin FBD:ssä. Hannu Asmalan opastuksella ohjelman ajoitus saatiin toimimaan ja ohjelma lukemaan koko taulukko.

Kun edellä mainittu vaihe oli saatu toimimaan, seuraava vaihe oli lisätä tuohon sekvenssiin myös datan kirjoittaminen. Sekvenssi on samanlainen kuin lukusekvenssi. Kirjoittamisen aloittamisen ehtona on, että luku on suoritettu loppuun. Kirjoittamisen jälkeen siirrytään takaisin lukemaan data.

Ongelmaksi lukemisen ajoittamiseksi tuli Busy signaalin käyttö. Tuo signaali on sama luku- ja kirjoitustoimilohkoille, eli kun suoritetaan datan lukemista, on kirjoituslohkossa busy signaali samanaikaisesti päällä, vaikka kirjoitusta ei suoritettaisikaan. Signaali kertoo siis väylän tilan, ei niinkään toiminnan tilaa.

Sekvenssi lähti lopulta toimimaan ohjelmamuutoksilla. Suurin vaikeus oli löytää sellaiset ehdot ja ajoitus, ettei kirjoitus ala ennen kuin koko luku oli suoritettu.

Kun sekä luku- että kirjoitussekvenssi toimivat, aloin testaamaan taajuusmuuttajan käynnistämistä väylän kautta. Vaconissa tuo on yksinkertainen logiikka, Control Wordin (CW) eli ohjaussanan ensimmäinen bitin arvoksi asetetaan arvo 1.
Ennen kuin taajuusmuuttajaa voi ohjata kenttäväylän kautta, täytyy taajuusmuuttajan ohjauspaikaksi valita paneelista Fieldbus (FB) eli kenttäväylä. Valinta suoritetaan ohjauspaneelin valikosta M3 parametri 3.1 ohjauspaikka / Control place. Valintoina on IO, Paneeli ja kenttäväylä.
Ohjaus toimi heti ensi yrittämällä. Testilaitteiston 1 testi oli näin ollen valmis. Pieniä ohjelmamuutoksia jouduin tekemään, liittyen lähinnä käynnistyslogiikkaan esim. jos vikatieto tilasanassa oli päällä, ei käy-käskyä lähetty lainkaan. Tällaisten ehtojen ohjelmointi on enemmän sovellusohjelmointia, kuin Modbus-väylän kommunikointiin liittyvää, joten en keskittynyt ehtojen ohjelmointiin kovin intensiivisesti.

6.1.2 EL6021 Moduuli

EL6021 sarjaliikenteen testaus tapahtui samankaltaisesti kuin KL6021 testaus. KL6021 testauksesta oli monella tapaa apua esimerkiksi Beckhoffin ja taajuusmuuttajan johdotus saatiin varmennettua KL6021 testauksen aikana.

EL6021 moduulin testilaitteisto oli erilainen kuin KL6021:lle tarkoitettu, koska moduulit eivät ole yhteensopivia. Merkittävimpänä erona testilaitteistoissa oli suoritin eli CPU (Central Processing Unit). KL6021 testilaitteiston ohjelma ajettiin tietokoneessa, kun taas EL6021 testissä ohjelma ladattiin keskusyksikköön, joka oli pienoistietokone sisältäen Windows ® XP käyttöjärjestelmän.

Toinen merkittävä ero moduulien välillä oli alustus, joka on selostettu jo aiemmin tässä dokumentissa.

Ohjelmaan ei tullut suuria muutoksia. Joidenkin tietueiden tavujen (Byte) tai sanojen (Word) määrää piti kasvattaa, mutta varsinainen kommunikointiohjelma pysyi entisellään.

Suurimmat vaikeudet läpi käytyäni KL6021 testauksessa, EL6021:n testaus sujuikin jo paljon nopeammin.

EL6021 testauksesta Vaconin taajuusmuuttajan kanssa ei ole suurempaa huomionarvoista mainittavaa. Piti vain muistaa sarjaliikennemoduulin alustuksen ero.

Myöhemmässä vaiheessa tällä laitteistolla testattiin usean rekisterin samanaikainen luku ja rekisterien kirjoitus. Tuota testausta varten ei laiteasetuksissa tehty mitään muutoksia, kaikki tarvittavat muutokset tehtiin ohjelmallisesti. Tästä ohjelmasta on kerrottu tarkemmin ohjelmointi-osiossa. Tämän ohjelman testauksesta suoriuduin melko lailla nopeasti, johtuen ohjelman yksinkertaisesta rakenteesta.
Liitteessä 2 on esitetty sekvenssikaavio datan lukemiseen ja kirjoittamiseen tätä ohjelmaa käyttäen.

6.2 BECKHOFF-ABB TESTAUS

EL6021 laboratoriotestauksen jälkeen siirryttiin kokeilemaan ohjelman toimivuutta Uusiutuvien Energioiden Hybridijärjestelmän ohjauslogiikalla ja siihen Modbus-kenttäväylällä liitetyllä ABB:n ACH 550 taajuusmuuttajalla. Tein testauksen yhdessä

Tommi Lehtisen kanssa, joka oli ohjelmoinut järjestelmän logiikkaa, ja tunsi järjestelmän hyvin.

EL6021 moduulin asetukset ja Modbus-ohjelma säilyvät samanlaisina kuin laboratoriotestauksessa yhtä poikkeusta lukuun ottamatta, globaalimuuttujataulukko täytyi ohjelmoida ABB:tä varten uudestaan, koska ABB:n rekisterien osoitteet ovat erit.
ABB:n taajuusmuuttajan kommunikointiparametrien asettelut ovat hyvin samanlaiset kuin Vaconin; ID, Baudnopeus, pariteetti jne. Parametriosoitteisto poikkeaahuomattavasti toinen toisistaan. ABB:ssa myös parametrien määrä on jonkin verran suurempi kuin Vaconissa käyttämässäni sovelluksessa. Molemmissa löytyy samoja parametreja, mutta niiden löytäminen toisesta voi vaatia kestää hetken parametrien nimien eroavaisuuksien vuoksi. ohjauspaneelin valikoiden navigoinnissa totuttelua on myös jonkin verran opeteltavaa Vaconin käytön jälkeen.

Kommunikointiparametrien asettelujen jälkeen monitoroimme Beckhoffin logiikasta taajuusmuuttajalta tulevia signaaleja. Saimme tilasanan (StatusWord) luettua heti logiikkaan. Tästä tiesimme, että ainakin vastaanotto eli luku toimi.

Seuraava vaihe oli yrittää taajuusmuuttajan käynnistämistä Modbus-väylän kautta. Parametrit eivät kaikki olleet aseteltu kenttäväyläohjausta varten. Parametrit 1001 ja 1002 olivat vielä asettelematta. Samoin nopeusohjeen valinta 1103 piti vaihtaa kenttäväylälle.

Yrityksistä huolimatta emme saaneet taajuusmuuttajaa käymään kenttäväylän kautta. Soitimme ABB:n tekniseen tukeenkin, kysyäksemme lisäohjeita käynnistykseen. Teknisen tuen soittoa odotellessa, selailin käyttöohjeita lisää. Löysin selostuksen CW:n (Control Word) ohjaussanan käytöstä. Toisin kuin Vaconin taajuusmuuttajassa, jossa riittää Start bitin ohjaus päälle/pois, ABB:n taajuusmuuttajassa on tietty sekvenssi miten ohjaussanan bitit täytyy laittaa päälle. Ohjaussanalla täytyy taajuusmuuttaja saattaa ensin Ready eli Valmius tilaan. Vasta tuon jälkeen voidaan antaa käy-komento. Jos bitit laitetaan päälle väärässä järjestyksessä, ei taajuusmuuttaja käynnisty. Oikea bittien ajoitus ja käyttö on selostettu ABB:n kenttäväylä EFB (Embedded Fieldbus) ohjekirjassa. (ABB Fieldbus, 2007)

Ohjaussanan oikean sekvenssin lähettämisellä saimme vihdoin taajuusmuuttajan käymään. Kaikki luettavat rekisterit päivittyivät ja pystyimme käynnistämään ja pysäyttämään taajuusmuuttajan väylältä, mutta eteen tuli vielä yksi pienimuotoinen ongelma. Taajuusmuuttaja ei vaikuttanut seuraavan väylältä lähetettyä nopeusohjetta, erityisesti hidas nopeus ei ollut mitä väylältä lähetettiin. Vaikutti siltä kuin taajuusmuuttaja olisi ajanut vakionopeutta. Lopulta tähän löytyi selitys. Taajuusmuuttajan minimi taajuusraja astui voimaan kun kenttäväylältä annettu taajuusohje alitti minimirajan, parametri 2007 MIN.FREQ. Jos taajuusohje kenttäväylältä oli vähemmän kuin ko. minimiraja, taajuusmuuttajan lähtötaajuus jäi minimirajaan.

Tuon pienen virheen jälkeen kommunikointi ja ohjaus toimivat niin kuin laboratoriotesteissä. Tämän jälkeen ohjauksen viimeistely oli sovellusohjelmointia, jonka kehittely jäi Tommi Lehtiselle.

7 YHTEENVETO

Projektissa testattiin kahta erilaista tapaa kommunikoida Modbus-väylällä. Molemmilla ovat omat hyvät ja huonot puolensa. Yksittäisten rekisterien luku ja kirjoitus kommunikoinnissa hyvänä puolena voitaisiin mainita globaalitaulukon helppo muuttaminen ja signaalien lisäys. Periaatteessa taulukko voisi olla hyvin laaja, haittapuolena taulukon kasvaessa, seuraa kommunikoinnin hidastuminen. Monitoroitaessa arvoja nopeudella ei ole suurta merkitystä, mutta jos signaaleita on tarkoitus käyttää säätöjen takaisinkytkennässä, silloin globaalitaulukon käyttö ei ole tarkoitukseen soveltuva. Ohjauskäskyt voivat olla myös hitaita tätä tapaa käytettäessä. Tähän voi tosin vaikuttaa sovelluksen ohjelmoinnilla siten, että ohjaussana saa aina korkeimman prioriteetin, eli vaikka datan luku olisikin kesken, ohjaussanan kirjoitus suoritettaisiin aina sanan muuttuessa. Tällä tavoin ohjauksen vasteesta saataisiin nopeampi.
Etuna voisi mainita, että luettavat rekisterit voivat olla hajallaan, joten käyttäjä saa juuri haluamansa tiedot. Toinen positiivinen asia on väylän kevyt kuormitus.

Usean rekisterin samanaikaisessa lukemisessa ja kirjoittamisessa suurimmaksi eduksi nousee nopeus. Tätä datasiirtotapaa voisi käyttää jo ohjauksiin ja jossain määrin myös säätämiseen. Tämän tavan käytön negatiivisena puolena täytyy mainita seuraavat asiat; Väylän kuormittuminen ja mahdollinen ylimääräisen datan luku, jolla tarkoitan sitä, että kun luetaan useita rekistereitä kerralla saattaa rekistereiden välissä olla tarpeetonta dataa. Taajuusmuuttajavalmistajat pyrkivät asettelemaan luettavat rekisterit siten, että välissä ei olisi nk. turhaa dataa. Mikä sitten on turhaa dataa, se riippuu sovelluksen käyttäjän ja ohjelmoijan näkökannasta.

Projekti antoi myös pienen mahdollisuuden vertailla eri taajuusmuuttajien eroavaisuuksia ja yhteneväisyyksiä.

Pidin projektin haasteellisuudesta. Erityisesti Beckhoff-ympäristöön tutustuminen oli minulle projektin haasteellisin osuus. Opin minulle uuden valmistajan ohjelmointiympäristöstä pienen osan. Taajuusmuuttajamaailma minulle oli jo entuudestaan tuttu, joten tuo osio ei antanut kovin paljon uutta, ainakaan Vaconin osalta. ABB:stä opin uutta, kuten ohjauspaneelin käytön, ja varsin mieleenpainuvana ohjaussanan oikean käytön.

Näen projektin antaman tiedon ja opin hyvin hyödylliseksi, jolle toivottavasti löytyy käyttöä tulevaisuudessa.

LÄHTEET

ABB ACH550. (7. Heinäkuu 2009). *http://www05.abb.com/*. Haettu 10. Toukokuu 2014 osoitteesta ABB: http://www05.abb.com/global/scot/scot201.nsf/veritydisplay/7ccd70ad6b24df53c1257607003d1969/$file/en_ach550_01_um_f_a4_loscreenres.pdf

ABB Fieldbus. (31. Toukokuu 2007). *http://www05.abb.com/*. Haettu 10. Touko 2014 osoitteesta ABB: http://www05.abb.com/global/scot/scot201.nsf/veritydisplay/25ba8ab3f04e2266c12572e9004ffafe/$file/en_ach550_efb_d.pdf

Beckhoff. (24. Lokakuu 2006). *http://download.beckhoff.com/*. Haettu 8. Toukokuu 2014 osoitteesta Beckhoff : http://download.beckhoff.com/download/Document/BusTermi/BusTermi/KL6021en.pdf

Beckhoff. (ei pvm). *http://infosys.beckhoff.com/*. Haettu 8. Toukokuu 2014 osoitteesta http://infosys.beckhoff.com/english.php?content=../content/1033/tcmodbussrv/html/tcmodbussrv_fb_mbwriteregs.htm&id=

Beckhoff PLCLib. (2014). *http://www.beckhoff.com/*. Haettu 10. Toukokuu 2014 osoitteesta Beckhoff: http://www.beckhoff.com/english.asp?twincat/te1000.htm

Grid Connect. (2012). *http://gridconnect.com/*. Haettu 7. 5 2014 osoitteesta Grid Connect .

http://www.modbus.org/. (26. Huhtikuu 2012). Haettu 7. Toukokuu 2014 osoitteesta Modbus Organization.

Modbus. (2014). *http://www.modbus.org/*. Haettu 7. Toukokuu 2014 osoitteesta Modbus organization: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Simply Modbus. (2013). *http://www.simplymodbus.ca/*. Haettu 7. Toukokuu 2014 osoitteesta Simply Modbus.

VACON NXS User Manual. (2012). *www.vacon.com*. Haettu 9. Toukukuu 2014 osoitteesta Vacon : http://www.vacon.com/ImageVaultFiles/id_2784/cf_2/Vacon-NXS-NXP-User-Manual-DPD01218A-FI.PDF?634995835905300000

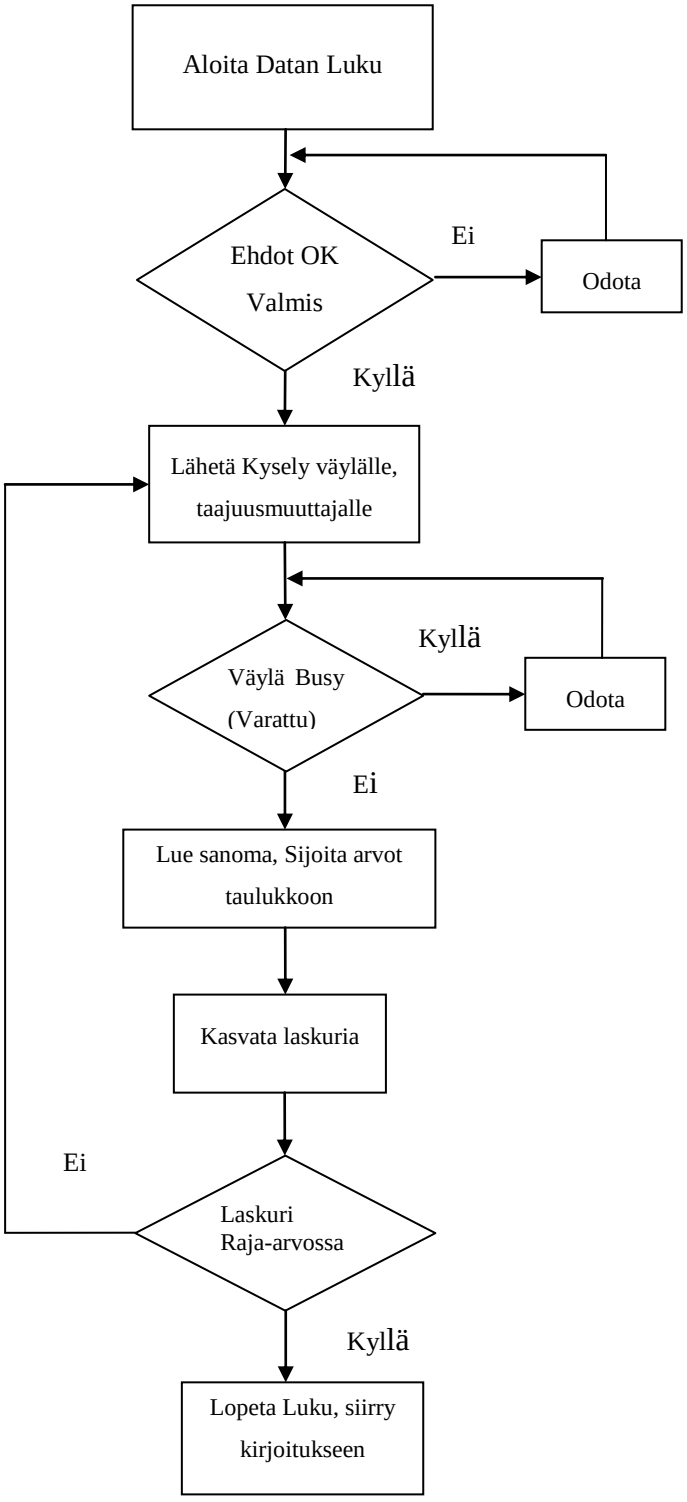
VACON OPTC2 . (2012). *www.vacon.com*. Haettu 9. Toukokuu 2014 osoitteesta Vacon: http://www.vacon.com/ImageVaultFiles/id_3118/cf_2/Vacon-NX-OPTC2-C8-Modbus-N2-Board-User-Manual-DPD0.PDF?635223804391930000

LIITTEET

LIITE 1

Sekvenssikaavio datan lukemiseen Modbus-väylällä tietuerakennetta käyttäen

Kirjoitussekvenssi on samanlainen



LIITE 2

Ohjelmalistaus Liiteen 1 mukaisen sekvessin toimintaan

(* TÄTÄ LOHKOA EI TARVITA EL6021, KOMMUNIKONTIPARAMETRIT ASETELLAAN SUORAAN KORTILLE, KLCONFIGURATION TARVITAAN VAIN
KL6021 KORTILLE*) **Kommnettilauseke kortin alustuksesta**

```
IF _TaskInfo[1].FirstCycle THEN
        bConfig:=TRUE;
END_IF

KL6configuration1( Tämä on kommunikoinnin alustus toimilohko
        Execute:=bConfig ,
        Mode:=SERIALLINEMODE_KL6_22B_STANDARD , (tämä riippuu kommunikointikortin tyypistä)
        Baudrate:=19200 , Datasiirtonopeus
        NoDatabits:=8 , Databittien määrä
        Parity:=PARITY_EVEN , Pariteetti
        Stopbits:=1 , Stop bittien lukumäärä
        Handshake:=RS485_HALFDUPLEX , (kommunikoinnin tyyppi)
        ContinousMode:=TRUE  , (jatkuva luku, ei taukoja viestien välissä)
        pComIn:=ADR(MB.InData) , (Luettavan datan rajapintamuuttuja)
        pComOut:=ADR(MB.OutData) , (kirjoitettavan datan rajapintamuuttuja)
        SizeComIn:=SIZEOF(MB) , (Selitys Beckhoffin sivuilla)
        Done=>ConfigDone , (Alustus suoritettu signaali)
        Busy=> ConfigBusy, (Alustusta suoritetaan)
        Error=> , (Virhe)
        ErrorId=> ); (Virheen koodi)


IF ConfigDone=TRUE THEN  (jos alustus suoritettu)
        ConfigFinished:=ConfigDone;
        bConfig:= FALSE;
END_IF
*) Tämä tekee lauseista kommentteja kaikki näiden (*   *) välillä muutetaan kommenteiksi
ConfigFinished:=TRUE;

IF ConfigDone=TRUE THEN
        ConfigFinished:=ConfigDone;
        bConfig:= FALSE;
END_IF
IF NOT bWrite THEN (tarkistetaan että Datan kirjoitusta ei olla suorittamassa)
  iAddress:=stModbusComm[iCounter].Address; (huomaa että laskurin arvo määrää luettavan osoitteen)
  iModbusFunction:=stModbusComm[iCounter].FunctionType;
        MB.ReadRegs(
```

```
                UnitID:= 1,                 (* Tamun slave osoite *)
                Quantity:= 1,               (* Luettavien wordien määrä *)
                MBAddr:=iAddress,           (* Osoite tulee globaali taulukosta *)
                cbLength:= 2,               (* Tavujen määrä *)
                pMemoryAddr:=ADR(iData1) , (* muuttaja mihin arvo luetaan *)
                Execute:=bRead1 ,                   (* Lohkon suoritus komento *)
                Timeout:= timeoutvalue,             (* Kommunikointikatkoksen viive *)
                Busy => bReadBusy           ,       (* Väylä varattu tieto *)
                cbRead=> bRead);

 IF NOT bReadBusy AND NOT ReadBusyTmp AND ConfigFinished THEN (Jos ehdot tosi asetetaan bRead1
TOSI)
   bRead1 := TRUE;
 END_IF

 IF NOT bReadBusy AND ReadBusyTmp THEN (Virhekäsittely jos mb.Error on tosi luetaan error tiedot)
   IF  mb.Error       THEN
            stModbusComm[iCounter].Error:=mb.Error;
            stModbusComm[iCounter].ErrorID:=mb.ErrorId;
   ELSE
            stModbusComm[iCounter].value:=iData1; (laskurin arvon mukaiseen muuttujapaikkaan saa iDa-
            ta1 mukaisen arvon
            stModbusComm[iCounter].Error:=FALSE; Virhe saa epätosi ja id arvon 0
            stModbusComm[iCounter].ErrorID:=0;
   END_IF
   bRead1 := FALSE;
   iCounter := iCounter + 1;   Laskurin askellus
   IF  iCounter > iArrayLenght THEN   Kun laskurin arvo on suurempi kuin iArrayLenght luku loppuu ja
                                      kirjoitus alkaa samalla laskurin arvo asetetaan arvoon 1
            bRead1:=FALSE;
            bWrite:= TRUE;
    iCounter:=1;
   END_IF

 END_IF
 ReadBusyTmp := bReadBusy; Sisäinen ohjelmamuuttuja, jolla saadaan nouseva signaali koska ensimäisel-
lä kerralla bReadBusy on epätosi. signaali kirjoitetaan MB.readRegs toimilohkon lähdöstä. Signaalin vuo-
rottelulla saadaan nouseva reuna bRead1 signaalille, ettei olisi koko ajan päällä.
END_IF

StatusWord:=stModbusComm[1].value; Tilatieto joka saa ensimmäisen taulukkopaikan arvon osoite 2100,
taajuusmuuttajassa 2101. Huomaa että osoitteet ovat Vacon taajuumuuttajalle
AuxStatusWord:=stModbusComm[2].value; Ylimääräinen tilasana
ActSpeed:=stModbusComm[3].value; Nopeusolo saa taulukkomuuttaja 3 arvon
```

// Tästä alkaa osoitteiden kirjoitus osuus. Luku ja Kirjoitus eivät voi olla samaan aikaan päällä.

Start:= Run; Start **signaalin kirjoitus**
Reset:= Run1; **Vikakuittaus signaali**

//Tässä kirjoitetaan Nopeusohje
stModbusWrite[1].value:= SpeedReference;

IF Start THEN
stModbusWrite[0].value:=1; **Jos start on Tosi kirjoitetaan ensimmäiseen kirjoitustaulukkopaikkaan 1**
ELSE
stModbusWrite[0].value:=0; **Muulloin muuttuja saa arvon 0**

END_IF

IF DriveFaulted AND Reset THEN **Jos käyttössä vika ja Reset signaali päällä kirjoitetaan control wordiin 4 eli Fault Reset**
stModbusWrite[0].value:=4;

END_IF

IF DriveFaulted AND NOT **Reset THEN Jos käyttö on viassa ja ei Reset signaalia, kirjoitetaan Control wordiin 0**
stModbusWrite[0].value:=0;
END_IF

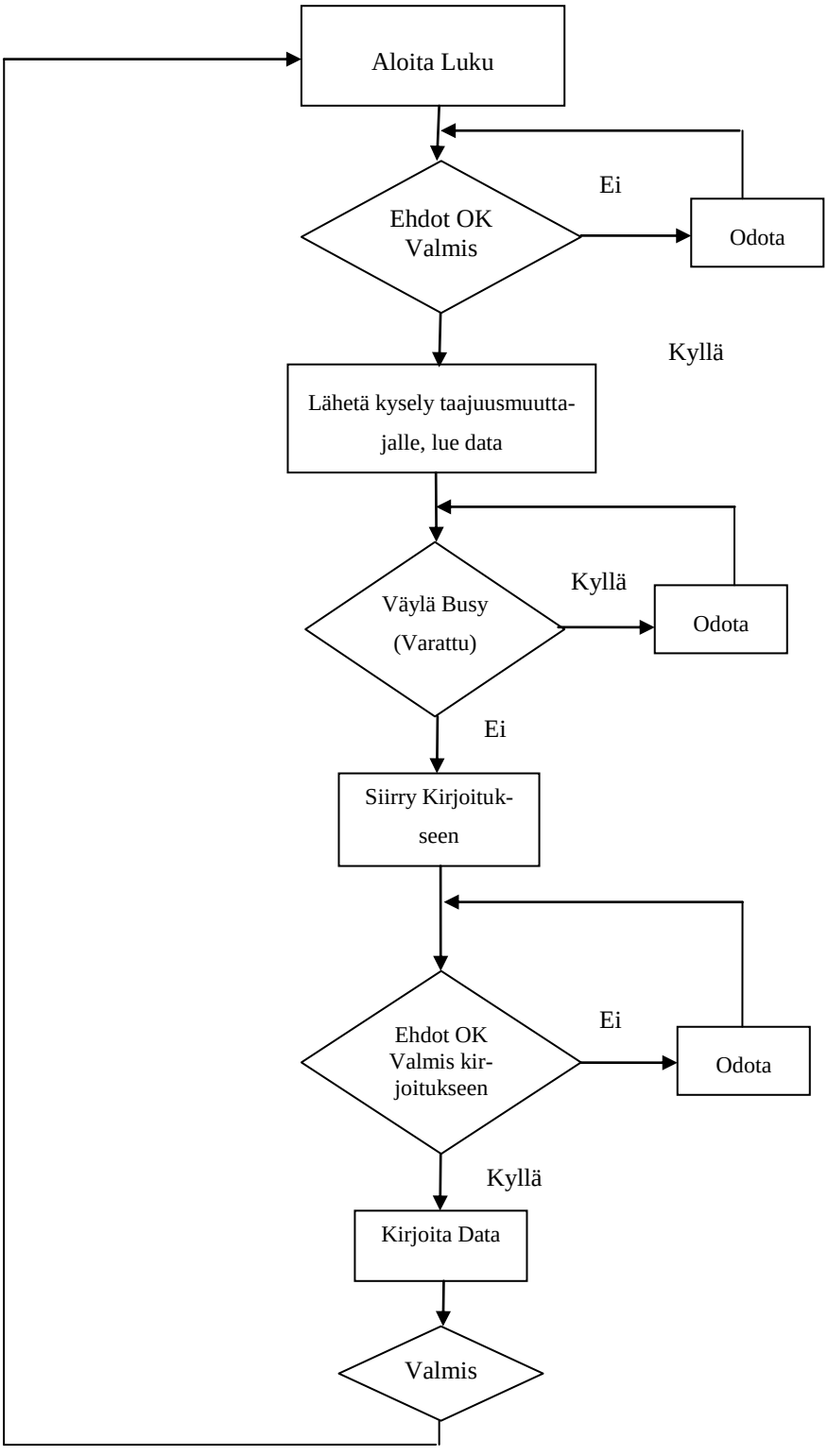
IF bWrite AND NOT bRead1 AND ConfigFinished THEN **Tässä ohjelmaosassa kirjoitetaan signaalien arvot laskurin osoituksen mukaan**
iDataOut2:=stModbusWrite[iWCounter].value;
iWriteAddress:=stModbusWrite[iWCounter].Address;
iModbusFunction:=stModbusWrite[iWCounter].FunctionType;
END_IF

IF bWrite THEN **Jos kirjoitus on päällä niin suoritetaan toimilohko**
MB.WriteSingleRegister(
UnitID:= 1, (* station address *) **Orjan osoite**
Quantity:= 1, (* WORDs *) **sanojen määrä tässä tapauksessa aina 1**
MBAddr:= iWriteAddress, **MB osoite laitteessa, tulee taulukosta**
cbLength:= 2, **Tavujen määrä**
pMemoryAddr:=ADR(iDataOut2), **Kirjotusmuuttuja**
Execute:=Write , **Toimilohkon suoritus signaali kun tosi**
Timeout:= timeoutvalue, **Aikaviive kommunikoinnille eli jos pitemmän ajan**
Busy => bWriteBusy , **Portti varattu eli kirjoitus meneillään**
Error=> , **Virhetila**
ErrorId=> , **Virhekoodi**

```
                    cbRead=>                    );

 IF NOT bWriteBusy AND NOT WriteBusyTmp THEN Jos nämä signaalit kaikki tosi asetetaan kirjoitus päälle
   Write:=TRUE;
 END_IF

 IF NOT bWriteBusy AND WriteBusyTmp THEN Jos nämä signaalit tosi on virhetila päällä
   IF mb.Error THEN
          stModbusWrite[iWCounter].Error:=mb.Error;
          stModbusWrite[iWCounter].ErrorID:=mb.ErrorId;
   ELSE                    Muulloin stModbusWrite vrhetilat asetetaan nollaksi
          stModbusWrite[iWCounter].Error:=FALSE;
          stModbusWrite[iWCounter].ErrorID:=0;
   END_IF
   Write:= FALSE;
   iWCounter := iWCounter + 1;        Laskurin askellus
 END_IF

 IF iWCounter > iWriteArrayLength THEN Kun laskurin arvo ylittää taulukon pituuden kirjoitus lopetetaan
   bWrite:=FALSE;
        //bRead1:= TRUE;
   iWCounter:=0;   Laskuri asetetaan nollaan
 END_IF

 WriteBusyTmp:=bWriteBusy; Sisäinen tilapäinen muuttuja jolla saadaan nouseva signaali
END_IF
```

LIITE 3

Sekvenssikaavio datan lukemiseen ja kirjoittamiseen Modbus-väylällä usean rekisterin toimilohkoja käyttäen



LIITE 4

Ohjelmalistaus Liiteen 3 mukaisen sekvessin toimintaan

//Tästä alkaa usean osoitteen luku ohjelma

(*//Datan luku Tamusta

```
MB.ReadRegs(
        UnitID:= 1,          ( Tamun slave osoite )
        Quantity:= 7,        ( Wordien lukumäärä käytettiin EL 6021 korttia )
        MBAddr:= 2100,   ( Ensimmäisen muuttujan osoite Tamussa -1, tamussa osoite on oikeasti 2101 )
        cbLength:= 14,       ( Tavujen määrä tähän täytyy laittaa word * 2 )
        pMemoryAddr:=ADR(iData2) ,          (Taulukko mihin luku suoritetaan )
        Execute:=bRead1,  (* Suorita lohko *)
        Timeout:= timeoutvalue, Viiveaika
        Busy => bReadBusy            , portti varattu
        cbRead=> bRead);
```

IF NOT bReadBusy AND NOT WriteBusy AND NOT Write AND ConfigFinished THEN **Ehdot lukemisen aloittamiseen**

```
    bRead1 := TRUE;
ELSE bRead1:= FALSE;
ReadBusyTmp := bReadBusy; Siäsinen muuttuja jolla saadaan aikaan nouseva signaali

END_IF;
```

//Luetaan sisään tilasanat ja nopeusolo ja muut signaalit datapaketissa ja sijoitetaan muuttujiin

```
StatusWord:=iData2[1]; Numero suluissa kertoo taulukon paikan
AuxStatusWord:=iData2[2];
ActSpeed:=iData2[3];
FreqOut:=iData2[4];
MotorSpeed:=iData2[5];
MotorCurrent:=iData2[6];
MotorTorque:=iData2[7];
```

// Tästä alkaa osoitteiden kirjoitus osuus. Luku ja Kirjoitus eivät voi olla samaan aikaan päällä.

//Run signaali on digitaalitulo suoraan kortille

```
Start1:=Run;

iDataOut1[2]:= SpeedReference; //Tässä kirjoitetaan Nopeusohje
```

```
//Start logiikka, ei käynnistystä jos käytössä vika päällä
IF Start1 AND NOT DriveFaulted THEN
iDataOut1[0]:=1;
ELSE
iDataOut1[0]:=0;
END_IF
//Reset signaalin kirjoitus jos käyttö on viassa
IF DriveFaulted AND Reset THEN
iDataOut1[0]:=4;
END_IF

//Datan lähetys
  IF  NOT bRead1 AND NOT bReadbusy THEN
      Write:=TRUE;
  ELSE Write:= FALSE;

  WriteBusyTmp:=bWriteBusy;
END_IF;


 MB.WriteRegs(
            UnitID:= 1, (* station address *)
            Quantity:= 3,          (* WORDs *)
            MBAddr:= 2000,  (* Ensimmäisen sanan osoite taajuusmuuttajassa *)
            cbLength:= 6,          (* Tavujen (bytejen) määrä *)
            pMemoryAddr:=ADR(iDataOut1), (* lähetystaulukko *)
            Execute:=Write ,     (*lohkon suoritus signaali*)
            Timeout:= timeoutvalue, (*kommunikoinnin aikaviive*)
            Busy => WriteBusy             , (*Väylä varattu kirjoitukselle*)
            Error=>                       , (*Virhetila*)
            ErrorId=>                     ,);*)  (*Virhekoodi*)
```