OULUN AMMATTIKORKEAKOULU

Leevi Leinonen

AUTOMAATIOJÄRJESTELMÄN MODERNISOINTI

AUTOMAATIOJÄRJESTELMÄN MODERNISOINTI

Leevi Leinonen
Opinnäytetyö
Syksy 2020
Automaatiotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikka, automaatiotekniikka

Tekijä: Leevi Leinonen
Opinnäytetyön nimi: Automaatiojärjestelmän modernisointi
Työn ohjaaja: Tero Hietanen
Työn valmistumislukukausi ja -vuosi: Syksy 2020
Sivumäärä: 69

Opinnäytetyössä modernisoidaan vanha Siemensin STEP 5 -automaatiojärjestelmä metallivalimolla. Työssä vertaillaan erilaisia mahdollisia korvaavia järjestelmiä Siemensin valikoimista ja valitaan niistä kohteeseen soveltuvin.

Ohjelmaan ei tule toiminnallisia muutoksia, joten nykyistä ohjelmaa voidaan käyttää myös uudessa järjestelmässä. Ohjelmaan kuitenkin joudutaan tekemään muutoksia, yhteensopivuusongelmien välttämiseksi nykyaikaisessa järjestelmässä. Uusi ohjelmointiympäristö tulee olemaan Siemens TIA Portal.

Samalla tehdään muutoksia järjestelmän I/O-määrään poistamalla käytöstä vanha LED-osoitintaulu ja korvaamalla se nykyaikaisella HMI-paneelilla. HMI-paneelille tehdään uusi grafiikka ja toiminnallisuus, sekä yhdistetään se päivitettyyn ohjelmaan ja ohjelmoitavaan logiikkaan.

Sähköpiirustuksiin tulee myös suuria muutoksia modernisoinnin vuoksi. Opinnäytetyössä tehdään päivitykset ja tarvittavat muutokset myös piirustuksiin. Työhön ei kuulu asennustyöt.

Asiasanat: automaatio, ohjelmoitava logiikka, modernisointi, Siemens TIA Portal, Siemens STEP 5, Siemens HMI

ABSTRACT

Oulu University of Applied Sciences
Electrical and Automation Engineering, Automation

Author: Leevi Leinonen
Title of thesis: Modernization of Siemens automation control system
Supervisor(s): Tero Hietanen
Term and year when the thesis was submitted: Autumn 2020
Number of pages: 69

This thesis modernizes an older version of Siemens automation system. The system consists of Siemens STEP 5 programmable logic and various I/O devices. Client is mid-sized foundry specializing in aluminum sand casting. The thesis compares various possible replacement systems from Siemens' selections and selects the most suitable one for the client.

There will be no functional changes to the PLC program, so the current program can also be used in the new system. However, changes need to be made to the program to avoid compatibility issues with the modern system. The new programming environment will be Siemens TIA Portal.

Changes are also made to the systems I/O count by disabling an older LED indicator panel and replacing it with a modern HMI panel. New graphics and functionality will be made for the HMI panel, and it will be combined with the updated program and programmable logic.

There will also be major changes to the electrical wiring due to modernization, which leads to necessary changes being made to the electrical drawings. The thesis does not include installation work.

Keywords: automation, PLC, modernization, Siemens TIA Portal, Siemens STEP 5, Siemens HMI

SISÄLLYS

1 JOHDANTO

Opinnäytetyössä modernisoidaan automaatiojärjestelmä metallivalimolla. Asiakkaalla on käytössä jo vanhentunut Siemensin STEP 5 -järjestelmä. Ohjelmoitavan logiikan valmistaja ei enää valmista varaosia tai tarjoa tukea kyseiseen automaatiojärjestelmään. Asiakas kuitenkin haluaa varmistaa luotettavan tuotannon kulun tulevaisuudessa, jonka vuoksi opinnäytetyössä tehtävä modernisointi tapahtuu.

Uusi ohjelmoitava logiikkaa tulee myös olemaan Siemensin valmistama, sillä asiakkaalla on kunnossapito-osaamista juuri kyseisen valmistajan tuotteisiin. Työssä vertaillaan ja valitaan paras ratkaisu kohteeseen Siemensin valikoimista. Vanhan logiikan ohjelma siirretään uudelle alustalle ilman toiminnallisia muutoksia. Myös vanhoihin sähköpiirustuksiin tehdään päivitykset.

Opinnäytetyössä ei tehdä järjestelmään toiminnallisia muutoksia, mutta tuleva järjestelmä tulee sallimaan niiden lisäämisen myöhemmin. Asiakas voi siis halutessaan helposti laajentaa tai parantaa esimerkiksi diagnostiikkaa, tiedonkeruuta ja ohjausta myöhemmin.

Opinnäytetyössä käytetyt esimerkit ovat sovellettuja Siemensin järjestelmistä, osa asioista toimii tai näyttää erilaiselta eri valmistajan ohjelmoitavissa logiikoissa. Siemens on yksi suurimpia ohjelmoitavien logiikoiden valmistajia ja toimijoita Euroopassa.

2 AUTOMAATIOJÄRJESTELMÄ

Tässä työssä käsiteltävät järjestelmät ovat PLC (Programmable Logic Controller) eli ohjelmoitavia logiikoita. Ohjelmoitavat logiikat ovat pieniä tietokoneita. Ohjelmoitavat logiikat ovat yleisimpiä teollisuudessa, mutta niillä voi toteuttaa käytännössä minkä tahansa kokonaisuuden ohjauksen.

Yleensä ohjelmoitavat logiikat ovat suunniteltuja vaativiin olosuhteisiin, jolloin ne saattavat kestää suuria lämpötiloja, elektronisia häiriöitä ja piikkejä. Tavalliset mikrokontrollerit eivät välttämättä olisi optimaalisia teolliseen ympäristöön, joten yleensä niihin valitaan ohjelmoitava logiikka ohjaamaan automaatiojärjestelmää.

Tyypillisiä kohteita ovat kaikki tehtaat mutta myös esimerkiksi laivat, liikennevalot ja hissit. Ohjelmoitavien logiikoiden etu on niiden modulaarinen rakenne. Tämä mahdollistaa niiden helpon ja halvan huollon, sillä yleensä siitä vikaantuu vain yksi moduuli eikä koko järjestelmä.

2.1 Rakenne

Ohjelmoitava logiikka sisältää virtalähteen, prosessorin ja valinnaisen määrän I/O-moduuleita. Ohjelmoitavissa logiikoissa voi myös olla kehikoita, johon se asennetaan ja erilaisia liitäntämoduuleita. Ohjelmoitaviin logiikoihin voidaan lisäksi liittää erilaisia ohjauspaneeleita ja valvomopäätteitä, joilla prosessia voidaan ohjata. Ohjelmoitavia logiikoita on olemassa pieniä, jotka koostuvat vain kymmenistä tuloista ja lähdöistä. Järeämmät logiikat mahdollistavat tuhansien tulojen ja lähtöjen liitännän, lisäksi tuen erilaisille väylille, jolloin ohjelmoitavia logiikoita voidaan kytkeä toisiinsa tai niihin voidaan kytkeä älykkäitä laitteita. (1.)

Tulot toimivat ohjelmoitavissa logiikoissa tietona, joka tuodaan nimensä mukaisesti logiikalle. Lähdöt ovat tietoa, joka lähetetään logiikalta. Tulojen ja lähtöjen tilaa kuvataan yleisesti yhdellä bitillä. Bitti esittää tietoa, joka voi olla vain kahta erilaista arvoa (1 tai 0, tosi tai epätosi). Tulo voi esimerkiksi kuvata painikkeita

ohjaustaululta tai rajakytkimiä laitteilta. Lähdöillä voidaan ohjata esimerkiksi valoja ja moottoreita.

Tuloille ja lähdöille on myös mahdollista saada tarkempaa tietoa, tätä kutsutaan analogiseksi signaaliksi. Analoginen signaali on tyypillisesti 0 – 10 V:n tai 4 – 20 mA:n alueella oleva viesti, joka muutetaan I/O-kortin AD-muuntimella tietokoneelle luettavaan muotoon. Analogisen signaalin tarkkuus riippuu I/O-kortin AD-muuntimen resoluutiosta. (1.)

2.1.1 Analogiset tulot ja lähdöt

Analogisilla tuloilla voidaan lukea tietoa, joka täytyy esittää tarkemmassa kuin yhden bitin muodossa. Tieto kuitenkin muutetaan binääriseksi AD-muuntimen jälkeen. AD-muuntimen resoluutio vaikuttaa huomattavasti mitattavan suureen tarkkuuteen. Mitä suurempi mitta-alue on, sitä enemmän resoluutio vaikuttaa sen tarkkuuteen. Esimerkiksi pinnan tasoa mittaava pinnankorkeuden anturi voisi mitata aluetta 50 – 200 metriä, jonka se muuttaa virtaviestiksi välille 4 – 20 mA.

Taulukon 1 mukaan esimerkiksi binäärinen lukema 0110_0110_0101 vastaisi tässä tilanteessa 10.5 mA, jos alue olisi 4 – 20 mA. 10.5 mA saadaan laskemalla jokaista bittiä vastaavat arvot yhteen (4.00 + (4.00 + 2.00 + 0.25 + 0.13 + 0.016 + 0.0039) = 10.5 mA). Tämä samalla tarkoittaa, että 12-bittisen AD-muuntimen resoluutio on 0.0039 mA alueella 4 – 20 mA (16 mA / 2^{12}). Tätä tarkempaa arvoa ei voida esittää, sillä se on vähiten merkitsevän bitin arvo. Kaikki muut arvot voidaan esittää aina 20 mA:iin asti 0.0039 mA:n askelissa.

TAULUKKO 1. 12-bittisen AD-muuntimen arvot skaalalla 4 – 20 mA

Bit	11	10	9	8	7	6	5	4	3	2	1	0
mA	8.00	4.00	2.00	1.00	0.50	0.25	0.13	0.06	0.03	0.016	0.008	0.0039
Bin	0	1	1	0	0	1	1	0	0	1	0	1

Tässä esimerkissä mitattaisiin 50 – 200 metrin väliä, jolloin sen mittatarkkuus olisi 3,7 senttimetriä (150 m / 2^{12}). Anturi saattaisi kyetä paljon parempaankin tarkkuu-

teen, mutta AD-muuntimen resoluutio rajoittaa sitä. Riippuu paljon ohjelmoitavista logiikoista, minkä resoluution I/O-moduuleita siihen on saatavilla. Monesti myös joitain bittejä on varattu esim. ilmoittamaan virheistä, kuten johdon katkeamisesta. Bittikuviossa voi olla myös tilaa ylivuodon varalta, mikä laskee entisestään resoluutiota.

Analogisen virtaviestin mukana voidaan myös siirtää dataa HARTin (Highway Addressable Remote Transducer) avulla. Tämä mahdollistaa esimerkiksi anturien asetuksien muuttamisen samoja kaapeleita pitkin, joissa virtaviestikin kulkee. Olisi myös mahdollista esimerkiksi siirtää virtaviestillä painetta ja HARTin avulla lämpötilaa. HART siirtää digitaalista dataa siniaallon avulla samassa piirissä. Aaltojen taajuudella on voitu osoittaa binääristä tietoa. Ensimmäisiä käyttökohteita HARTin kaltaiselle modulaatiolle ovat olleet vanhat analogiset lankapuhelimet, joissa sen avulla on siirretty soittajan puhelinnumero vastaanottajalle.

2.1.2 Profinet

Profinet (Process Field Net) on väylä, jolla voidaan siirtää suuria määriä dataa Ethernetin yli. Uudemmissa ohjelmoitavissa logiikoissa tuki Profinetille löytyy vakiona, vanhempiin sitä varten on voinut joutua hankkimaan erillisen kommunikointimoduulin. Profinet on tullut markkinoille 2000-luvun alussa, joten sitä ennen on käytetty laajasti erilaisia muita väyliä. Profinet on nykyään kuitenkin lähes vaatimus etenkin suuremmissa Siemensillä toteutetuissa järjestelmissä.

Profinetin avulla ohjelmoitava logiikka voidaan liittää esimerkiksi käytettävään HMI-paneeliin tai operointipäätteeseen. Tekniikan ollessa tavallista Ethernettiä, voidaan väliin asentaa kytkimiä, jolloin järjestelmää saadaan helposti laajennettua. Profinetin avulla voidaan myös tehdä erilaisia hajautettuja järjestelmiä, joissa ohjelmoitavat logiikat ovat kytkettynä toisiinsa. Toimilaitteitakin voidaan kytkeä järjestelmään Profinetin avulla, mutta yleensä silloin on kyse älykkäistä toimilaitteista kuten taajuusmuuttajista.

2.2 Ohjelmointi

Ohjelmoitavia logiikoita voidaan ohjelmoida useilla eri menetelmillä ja kielillä. Yleisimpiä käytössä olevia ohjelmoitavien logiikoiden ohjelmointikieliä ovat standardin IEC-61131-3 sisältämät kielet. Siihen kuuluu kolme graafisesti luettavaa kieltä ja kaksi tekstimuotoista kieltä. (2.)

Standardiin kuuluvat tekstimuotoiset kielet Instruction List (IL) ja Structured Text (ST). Standardin graafiset kielet ovat Ladder Diagram (LD), Function Block Diagram (FBD) ja Sequential Function Chart (SFC). Standardi määrittelee samalla käytettävissä olevat datatyypit. Datatyyppejä ovat esimerkiksi kokonaisluvut (Integer), etumerkittömät kokonaisluvut (Unsigned Integer) ja liukuluvut (Real).

Käydään seuraavaksi esimerkkien kanssa läpi näitä kieliä, kuitenkin Siemensin järjestelmiin painotettuina. Kielien ominaisuudet ja funktiot voivat hieman vaihdella eri valmistajan mukaan. Esimerkiksi Siemensin vanha STEP 5 -järjestelmä voi osittain ominaisuuksiensa mukaan erota jo tuoreemmista STEP 7 -järjestelmistä, vaikka käytettävät kielet ovatkin samoja. Yleensä erot painottuvat valmistajien tarjoamiin valmiisiin funktioihin ja prosessorin käskykantaan.

2.2.1 Ladder Diagram

Tikapuulogiikka eli Ladder Diagram (LD, myös LAD) on edelleen yksi käytetyimmistä kielistä ohjelmoitavissa logiikoissa. Suosio johtuu pääsääntöisesti siitä, että kieli on alkuperäinen kieli, jolla ohjelmoitavia logiikoita on lähdetty ohjelmoimaan. Se on myös todella yksinkertainen, jolloin oppiminen on helppoa. Tikapuulogiikka pohjautuukin vanhoihin relekaavioihin, jolloin sähköasentajien oli helppo siirtyä niistä suoraan ohjelmoitaviin logiikoihin. (3.)

Tikapuulogiikat rakennetaan neljästä perusosasta. Koskettimet vastaavat ohjelmoitavalle logiikalle tulevia tuloja. Käämit puolestaan vastaavat lähtöjä.

- `–[ ]–` Normaalisti auki oleva kosketin. Kosketin sulkeutuu, jos sitä vastaava tulo tai ohjaus aktivoituu.

- `–[\]–` Normaalisti kiinni oleva kosketin. Kosketin aukeaa, jos sitä vastaava tulo tai ohjaus aktivoituu.
- `–( )–` Normaalisti epäaktiivinen ohjaus. Tämä aktivoituu, jos sitä edeltävä piiri on sulkeutunut, esim. edellä oleva kosketin aktivoituessaan sulkeutuu.
- `–(\)–` Normaalisti aktiivinen ohjaus. Tämä epäaktivoituu, jos sitä edeltävä piiri on sulkeutunut, esim. edellä oleva kosketin aktivoituessaan sulkeutuu.

Graafiset ydinpiirteet tikapuulogiikassa pohjautuvat siis vanhoihin relekaavioihin, joilla nyt toteutetaan Boolen algebran mukaisia lausekkeita. Tikapuulogiikkaa luetaan vasemmalta oikealle ja ylhäältä alas.

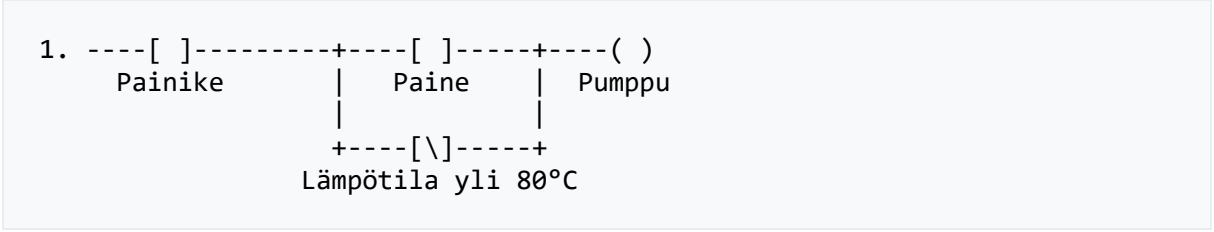
```
------[ ]--------------[ ]----------------( )
   Painike 1        Painike 2           Valaisin
```

Yllä on kuvattu Boolen lauseke AND. Valo aktivoituisi, eli menisi tilaan tosi / 1, kun painike 1 ja painike 2 ovat tilassa tosi / 1. Tilanteessa voisi olla kaksi kytkintä, jotka molemmat aktivoituessaan aktivoisivat katossa roikkuvan valaisimen. Logiikkaa voi ajatella eräänlaisena piirinä, kun molemmat painikkeet ovat pohjassa, sähkö pääsee vapaasti kulkemaan valaisimelle.

```
---+-------[ ]-------+-----------------( )
   |     Painike     |               Valaisin
   |                 |
   +-------[ ]-------+
      Liiketunnistin
```

Tämä puolestaan vastaa Boolen lauseketta OR. Valo aktivoitusi, eli menisi tilaan tosi / 1, kun painike tai liiketunnistin olisivat tilassa tosi / 1. Tilanteessa voisi olla oma painike lampun aktivointia varten. Esimerkin kuvatessa huoneiston käytävää voisi siellä olla myös erillinen liiketunnistin, joka aktivoituessaan aktivoisi tämän saman valaisimen. Piirissä painike ja liiketunnistin ovat siis rinnan, jolloin sähkö pääsee kulkemaan piirin lävitse, kun vain toinen tai molemmat ovat aktiivisena.

Yllä olevat esimerkit AND- ja OR-piireistä ovat yksinkertaisia segmenttejä. Todellisuudessa esimerkiksi teollisuuslaitosta ohjaava ohjelmoitava logiikka saattaisi koostua tuhansista segmenteistä. Tarkastellaan esimerkkiä, jossa käytetään hyväksi tätä toiminnallisuutta.



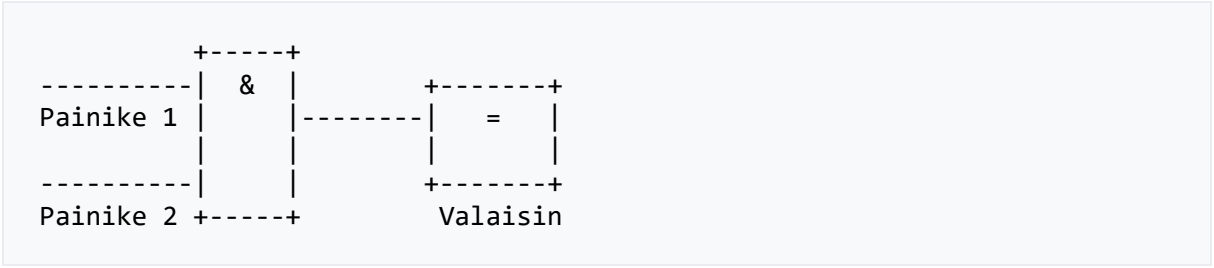
1. Pumppu = Painike AND (Paine OR (NOT Lämpötila))
2. Venttiili = Pumppu AND (NOT Hälytys)

Tämä esimerkki on jo hieman monimutkaisempi, mutta tuo esille hyvin, kuinka tikapuulogiikallakin voidaan tehdä monimutkaisia toiminnallisuuksia. Toinen segmentti ei voi aktivoitua, ellei ensimmäinen aktivoidu. Pumppu esiintyy molemmissa segmenteissä. Tämä tuo ilmi sitä, että samoja tuloja ja lähtöjä voidaan tarkastella useasta paikasta ohjelmaa.

2.2.2 Function Block Diagram

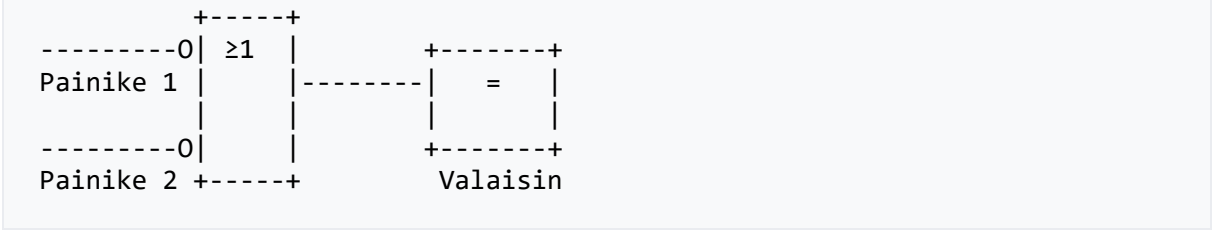
Toimilohkokaavio eli Function Block Diagram (myös CSF) ovat toisia yleisesti käytössä olevia standardin mukaisia graafisia ohjelmointikieliä. Toimilohkoilla voidaan kuvata suurempia kokonaisuuksia ja monimutkaisempia funktioita kuin tikapuulogiikalla. Tikapuulogiikalla voidaan kuitenkin tehdä samat asiat, mutta silloin tikapuulogiikkaan tuodaan toimilohkoja mukaan, jolloin kielet toimivat tavallaan sekaisin.

Toimilohkokaavio rakentuu lohkoista, joissa on lohkon omaava toiminnallisuus ja lohkon rajapinnat kuten tulo- ja lähtöportit.



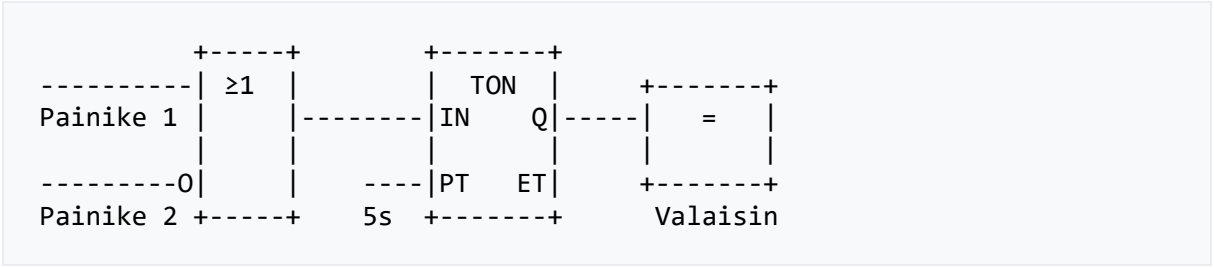
Yllä on kuvattu Boolen lauseke AND, joka aktivoi valaisimen. Vastaava esimerkki tehtiin myös tikapuulogiikalla. Toimilohkokaaviolla toiminteet ovat aina lohkoja, jolloin toiminnallisuutta kuvaa lohkon sisällä oleva merkintä tai nimi. AND merkitään &, OR merkittäisiin ≥1.

Tulojen tai lähtöjen invertointi toimilohkoilla tehdään asettamalla tulon tai lähdön päätyyn rengas, joka kuvaa kyseisen signaalin arvon invertointia. Alla on esimerkki, jossa on toteutettu sama esimerkki kuin aiemmin tikapuulogiikoilla, mutta nyt OR-lohkolle tulevat signaalit on lisäksi invertoitu.



Riippuen paljon ohjelmoitavan logiikan valmistajasta, ominaisuuksista sekä käytettävästä ohjelmointiympäristöstä voi valmiita toimilohkoja olla satoja. Yleensä käytetyimmät ja eniten tarvittavat ominaisuudet ovatkin valmiiksi ohjelmoituina toimilohkoiksi, jolloin logiikan ohjelmoija voi keskittyä niiden avulla ohjelman toteuttamiseen.

Esimerkiksi ajastin on hyvin yleisesti käytettävä toimilohko. Tämä löytyy poikkeuksetta valmiina, jolloin ohjelmoijan ei itse tarvitse rakentaa ajastinta vaan hän voi käyttää tätä valmista toimilohkoa. Tämä nopeuttaa ohjelmointia. Useimmat perustoiminnot ovatkin valmiina toimilohkoina ohjelmoijan käytettäväksi.



Yllä on toteutettu 5 sekunnin viive valaisimen käynnistykselle, kun painiketta 1 tai 2 painaa. Esimerkissä käytettävä TON-ajastin on Siemensin TIA-portaalista löytyvä toimilohko. Toimilohkossa on 4 porttia, joista 2 on tulosignaaleille ja 2 lähtösignaaleille. Toimilohkoilla tulot ovat aina lohkojen vasemmalla puolella, kun lähdöt ovat oikealla puolella. Ohjelmaa tuleekin lukea vasemmalta oikealle ja ylhäältä alas.

Kun IN-porttiin tulee signaali, ajastin aloittaa PT-portissa määritellyn ajan laskemisen. Esimerkissä PT-porttiin tuodaan arvona 5 sekuntia, jonka kuluttua Q-portti aktivoituisi. ET-portista saisi tarvittaessa sen ajan, jossa ajastin on juuri sillä hetkellä menossa. Jos IN-porttiin tuleva signaali ehtii epäaktivoitua ajastimen vielä odotellessa määriteltyä aikaa, loppuu myös ajan laskeminen ja kulunut aika palautuu alkutilaan.

2.2.3 Instruction List

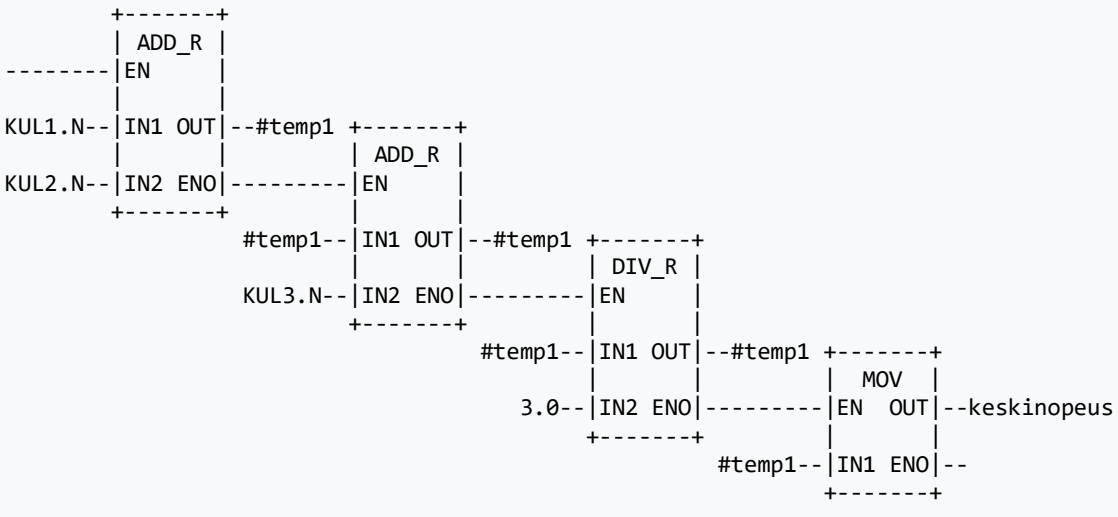
Käskylista eli IL tai Siemensin järjestelmissä Statement List eli ST on toinen standardin mukaisista tekstipohjaisista ohjelmointikielistä. Käskylista on matalan tason ohjelmointikieli. Kieli muistuttaa Assemblyä. Kielellä toteutettavat toiminnot muodostuvat käytettävän prosessorin käskykannasta. Käskyt ovat kaksiosaisia, osat ovat operaatiokoodi ja operandi.

Operaatiokoodi on vasemmalla puolella oleva osuus käskystä, joka voisi esimerkiksi olla ADD tai AND. Se kertoo, mitä täytyy tehdä. Operandi on oikealla puolella sijaitseva osa käskystä, se osoittaa, mihin vasemmalla puolella oleva operaatiokoodin mukainen funktio kohdennetaan, ohjelmoitavissa logiikoissa tämä yleensä kohdennettaisiin tuloihin ja lähtöihin, mutta myös rekistereihin.

```
A       I 0.0   // Painike 1
A       I 0.1   // Painike 2
=       Q 1.0   // Valaisin
```

Yllä on jo aikaisemmin käytetty esimerkki, jossa kaksi painiketta aktivoisi valaisimen. AND-toimintoa merkitään A ja esimerkiksi OR-toiminto olisi O. Lyhyet toiminallisuudet saadaan pysymään selkeinä ja helppolukuisina. Jos kuitenkin tarvitaan vähän laajempaa tai monimutkaisempaa toiminnallisuutta, tulee yleensä Instruction Lististä pitkä, jolloin sen lukeminen on hitaampaa kuin esimerkiksi graafisen FBD:n. Instruction Listissä ei ole abstraktiota ihmistä varten, vaan se on todella lähellä konekieltä.

Otetaan vertailuun kaksi esimerkkiä. Toinen toteutetaan FBD:llä ja toinen IL:llä. Toiminnallisuus voidaan kuitenkin koittaa pitää yksinkertaisena. Esimerkki voisi kuvata 3 eri taajuusmuuttajaa, joista saadaan nopeustieto REAL-datatyyppinä eli liukulukuna. Toiminnallisuuden tavoite olisi laskea näiden 3 taajuusmuuttajan nopeustiedon keskiarvo. Tähän voidaan käyttää hyödyksi valmiita funktioita: ADD_R ja DIV_R. Lohkojen välillä siirreltävää tietoa merkitään tässä esimerkissä paikallisella muuttujalla #temp1. Tämä olisi siis tietoa, joka ei ole saatavilla toimilohkon ulkopuolella. Kyseistä muuttujaa käytettäisiin vain tämän toimilohkon sisällä apuna yhteenlaskun laskemisessa. Jos muuttujan nimen edessä on ristikkomerkki, se tarkoittaa muuttujan olevan toimilohkon sisäinen muuttuja.



Edellä toiminnallisuus on toteutettu FBD-kieltä käyttäen. Toiminto mahtuu hyvin yhdelle ruudulle ja on helposti luettavissa. Tämä voisi olla Siemens S7-300-ohjelmointiympäristöstä.

Tarkastellaan seuraavaksi, miltä täysin samalla toiminnallisuudella toteutettu funktio näyttäisi Statement List -kielellä, Siemens S7-300-ohjelmointiympäristöstä.

```
      A(                      // And with nesting open 1
      A(                      // And with nesting open 2
      A(                      // And with nesting open 3
      L     ”KUL1”.N          // Load KUL1 value -> ACCU 1
      L     ”KUL2”.N          // ACCU 1 -> ACCU 2,KUL2 value -> ACCU 1
      +R                      // ACCU 2 + ACCU 1 = ACCU 1
      T     #temp1            // temp1 = ACCU 1
      AN    OV                // NOT STATUS bit 5, overflow
      SAVE                    // Save, STATUS bit 8 BR == 1
      CLR                     // Clear, STATUS bit 1 RLO == 0
      A     BR                // Binary Result == ENO = 1
      )                       // End nesting stack 3
      JNB   _001              // If RLO = 0, Jump to _001:
      L     #temp1            // Load temp1 value -> ACCU 1
      L     “KUL3”.N          // ACCU 1 -> ACCU 2,KUL3 value -> ACCU 1
      +R                      // ACCU 2 + ACCU 1 = ACCU 1
      T     #temp1            // temp1 = ACCU 1
      AN    OV                // NOT STATUS bit 5, overflow
      SAVE                    // Save, STATUS bit 8 BR == 1
      CLR                     // Clear, STATUS bit 1 RLO == 0
      A     BR                // Binary Result == ENO = 1
      )                       // End nesting stack 2
      JNB   _001              // If RLO = 0, Jump to _001:
      L     #temp1            // Load temp1 value -> ACCU 1
      L     3.000000e+000     // ACCU 1 -> ACCU 2, 3.0 -> ACCU 1
      /R                      // ACCU 2 / ACCU 1 = ACCU 1
      T     #temp1            // temp1 = ACCU 1
      AN    OV                // NOT STATUS bit 5, overflow
      SAVE                    // Save, STATUS bit 8 BR == 1
      CLR                     // Clear, STATUS bit 1 RLO == 0
      A     BR                // Binary Result == ENO = 1
      )                       // End nesting stack 1
      JNB   _001              // If RLO = 0, Jump to _001:
      L     #temp1            // Load temp1 value -> ACCU 1
      T     keskinopeus       // REAL keskinopeus = ACCU 1
_001: NOP   0                 // no operation / jump tag
```

STATUS-rekisteristä ei tarvitse välittää paljoa, jos ohjelmoidaan FBD- tai LAD-kielillä. Sen huomioiminen on kuitenkin tärkeää SL-kielellä. STATUS-rekisteri on 2 tavun pituinen CPU:n sisäinen rekisteri (Taulukko 2). CPU voi käydä tämän rekisterin lävitse yhden kellojakson aikana.

TAULUKKO 2. Rekisterin sisältämät bitit

8	7	6	5	4	3	2	1	0
BR	CC0	CC1	OV	OS	OR	STA	RLO	/FC

Rekisterissä käytetään vain ensimmäiset 9 bittiä, jolloin rekisterin tilaa voidaan kuvata binäärisesti: 0_0000_0000. Rekisterin muut bitit ovat aina 0.

Tärkeät bitit rekisteristä ovat bitti 0 eli /FC (First Check), bitti 1 eli RLO (Result of Logic Operation) sekä bitti 2 eli STA (Status).

```
      ↓STA=1   ↓RLO=1    ↓STA=1  ↓RLO=1    ↓STA=0   ↓RLO=0  ↓STA=0
------[-]----------------[-]---------------[ ]-------------( )
   Painike 1          Painike 2         Painike 3         Valaisin
```

LAD-kielellä bittien tarkoituksen huomaa hyvin. Vihreä kuvaisi käskyjen etenemistä ja piirin tilaa. Painike 1 ja 2 ovat pohjassa, mutta 3 ei ole. STA kuvaa aina jokaisen osoitteellisen kytkimen tilaa, eli jos painike (1) = 1, niin painike 1 STA = 1. RLO kuvaa käsiteltävän käskyn tulosta. Jos RLO = 1, niin silloin voidaan siirtyä seuraavaan käskyyn (4).

First Check osoittaa onko kyseinen kohta piirin ensimmäinen osa. Jos /FC on tilassa 0, sitä käsitellään ensimmäisenä käskynä. Jos /FC on tilassa 1, silloin käsittelyssä käytetään edellisen käskyn logiikkaa. Tietyt funktiot kuten SR-kiikku voi asettaa /FC bitin tilaan 0, joka tarkoittaisi uuden logiikan aloittamista sen jälkeen.

Viides bitti eli OV tulee käyttöön, jos tehdään laskemista liukuluvuilla. OV bitti asetetaan arvoon 1, jos aritmeettisen käskyn jälkeen on tapahtunut virhe. Virhe voisi kuvata esimerkiksi ylivuotoa, esiintyvää epälukua tai nollalla jakamista.

2.2.4 Structured Text

Rakenteinen teksti eli ST tai Siemensin järjestelmissä Structured Control Language eli SCL eroaa jo muista standardin asettamista ohjelmointikielistä melko selvästi. Se on korkean tason ohjelmointikieli, joka muistuttaa lähinnä Pascalia. Tällä kielellä voidaan toteuttaa vaativin toiminnallisuus ohjelmoitavissa logiikoissa nopeiten ja helpoiten. Tämä on uusimpia tuettuja ohjelmointikieliä ohjelmoitavissa logiikoissa. Tämän takia edelleen useissa vanhemmissa kohteissa törmätään aiemmin esitettyihin kieliin. ST on ohjelmointikieli, jota ei välttämättä ensisijaisesti opeteta esim. sähköasentajille, jotka ovat ohjelmoitavien logiikoiden kanssa tekemisissä kunnossapidon kautta. Voi olla siis edullista käyttää vanhoja ja perinteisiä kieliä, joihin on osaamista jo jollain tasolla.

Tarkastellaan aiempaa esimerkkiä kolmen taajuusmuuttajan keskinopeuden toteuttamisesta ST:llä.

```
keskinopeus := KUL1.N + KUL2.N + KUL3.N / 3.0;
```

Tästä voikin jo päätellä, miksi tämä on tehokkaimpia käytettäviä kieliä ohjelmoitavissa logiikoissa.

Jos ohjelmoitava toiminnallisuus koostuu vain Boolen lausekkeista, voivat LAD- ja FBD-ohjelmointikielet olla parempia ratkaisuja, juuri siitä syystä, että mahdollinen kunnossapito- tai huoltohenkilöstö kykenee diagnosoimaan sitä helpommin ja näkemään yleiskuvan.

Jos käsitellään muita kuin Boolen lausekkeita, on ST ylivoimainen luettavuuden kannalta, eikä se eroa suoritustehokkuudeltakaan muista saatavilla olevista ohjelmointikielistä (5, s. 14).

2.3 Ohjelman rakenne Siemens-ympäristössä

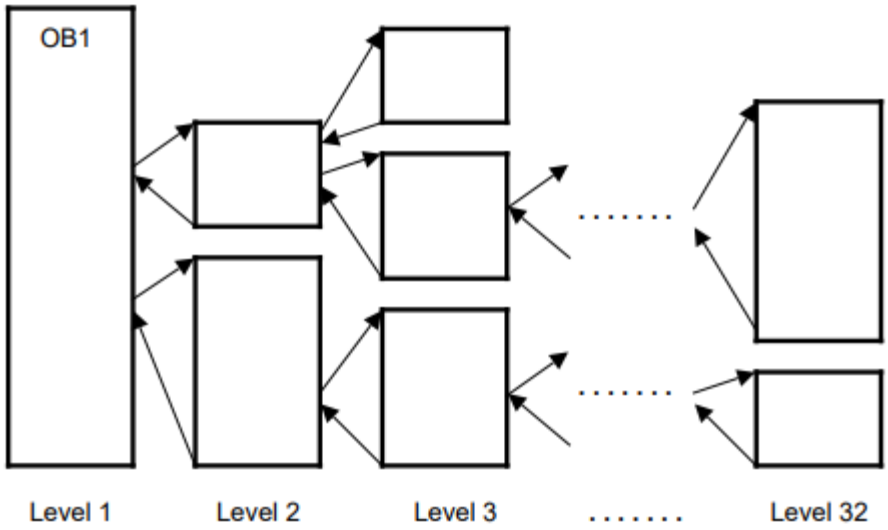
Ohjelmat rakennetaan Siemensin tarjoamissa ympäristöissä aina lohkoihin. Lohkoilla on oma hierarkiansa, joka määrää, mitä lohkoa pitää seurata aina milloinkin. OB1-lohko on kaikista tärkein, tämä lohko määrää, miten ohjelmaa suoritetaan normaalin käynnin aikana. OB-lohkot ovatkin ylimmällä tasolla. OB1-lohkon lisäksi on muitakin lohkoja, joita käytetään hieman erilaisissa tilanteissa kuten äkillisessä jännitteen häviämisessä tai tunnistetussa laitteiston viassa, kuten rikkoutuneesta I/O-moduulista. On myös olemassa OB-lohkoja, joiden suoritusväli voidaan itse määrittää. Esimerkiksi sekvensseissä tämä on hyödyllinen, sillä tällaista toiminnallisuutta ei tarvitse käydä ohjelmallisesti lävitse kuin esim. 10 kertaa sekunnissa.

Todella yksinkertaiset ja suoraviivaiset ohjelmat voidaan kirjoittaa suoraan OB1-lohkoon, eikä ohjelmassa tällöin ole tarvetta muille lohkoille. Kuitenkin jos tahdotaan selvittää monimutkaisempia toiminnallisuuksia, on järkevämpää jakaa koko ohjelma yksilöllisiin lohkoihin. Lohkojen etu onkin todella selkeä ohjelmallinen rakenne. Lohkoja voidaan myös käyttää tarvittaessa uudelleen samassa projektissa, mutta myös täysin eri projekteissa. Virheiden ja ongelmien löytäminen helpottuu, sekä virhe on mahdollista eristää helpommin selkeän rakenteen ansiosta.

Lohkot ja niiden tehtävät saattavat erota hieman eri STEP-versioiden välillä, mutta pääsääntöisesti ne ovat pysyneet tehtäviltään samoina. Kuitenkin ohjelma täytyy aina rakentaa näiden lohkojen ympärille. Lohkon tyyppi määrittelee pitkälti, millaista toiminnallista rakennetta lohkolta voidaan olettaa.

2.3.1 Organization Block

Organisointilohko eli OB on käytännössä lohkoista määräävin. Prosessori normaalin käynnin aikana kutsuu tätä lohkoa ensimmäisenä, jolloin OB-lohkon mukaisesti ohjelman suoritus alkaa. Kun viimeinen käsky on suoritettu, alkaa suoritus alusta taas OB-lohkon ensimmäisestä käskystä. Esimerkiksi STEP 5 -ympäristössä OB1 voi kutsua muita lohkoja tasolle 32 asti (Kuva 1).



KUVA 1. Nesting in OB (6, s. 180)

OB1 onkin rajapinta käyttöjärjestelmän ja ohjelmoijan ohjelman välillä ohjelmoitavissa logiikoissa. Tyypillisiä muita OB-lohkoja voisivat olla esim. OB21 ja OB22, jotka määräävät toimenpiteistä, jos ohjelmoitava logiikka asetetaan tilaan STOP tai jos se käynnistetään uudelleen. OB1-lohkoa voisikin kutsua Main-lohkoksi.

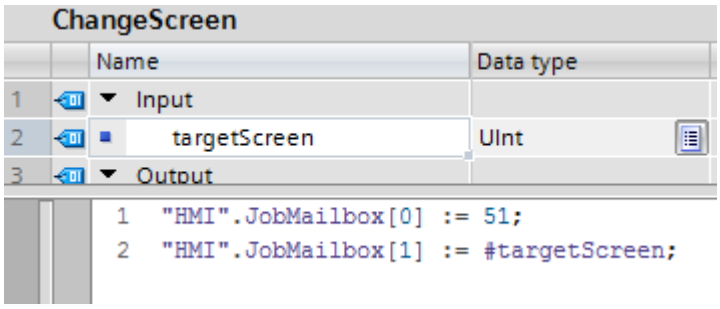
STEP 7 (TIA Portal) ympäristössä voi olla useampia MainOB-lohkoja, mikä mahdollistaa ohjelman entistä paremman kapseloinnin. Jos ohjelmassa on rakenteita, jotka voisi hoitaa täysin toinen ohjelmoitava logiikka, voidaan tämä kapseloida oman MainOB-lohkon alle. Jos on useampi MainOB-lohko käytössä, kannattaisi välttää luomasta kommunikaatiota niiden välille, sillä silloin ne eivät ole täysin erotettavissa toisistaan.

2.3.2 Program Block

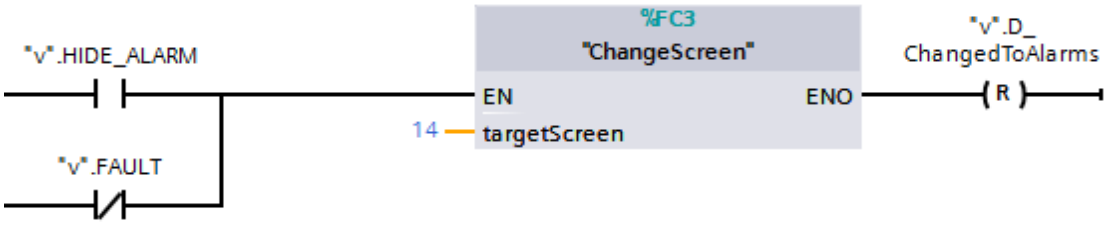
Ohjelmalohkot eli PB tai FC eli Function STEP 7 -ympäristössä ovat yksinkertaisia lohkoja. Funktiolohkot eivät varaa muistia, mikä hieman rajoittaa niiden käyttötarkoitusta. Yleensä funktiolohkot ovat tarkoitettu toimintoihin, joita voidaan kutsua mistä ohjelman kohdasta tahansa ja useita kertoja yhden suoritussyklin aikana.

Aikaisempi esimerkki kolmesta taajuusmuuttajasta ja niiden keskinopeuden laskennasta voisi olla yksi tällainen funktiolohko. Funktiolohkolla olisi silloin 3 tuloa ja 1 lähtö. Sen tuloihin olisi liitetty taajuusmuuttajien nopeudet. Lohko tekisi laskutoimituksen ja palauttaisi lähdön kautta vastauksen. Kun tämä lohko on suoritettu, ohjelma voi jatkua sieltä, mistä tätä funktiota kutsuttiin (Kuva 3). Tämä on esimerkki todella yksinkertaisesta funktiosta.

Monesti kuitenkin funktioissa on paljon muutakin toiminnallisuutta, sillä niillä voidaan tehdä esimerkiksi fyysisten I/O-moduuleiden lähtöjen ohjauksia tiettyjen ehtojen täyttyessä. Funktioilla ei ole myöskään pakko olla tuloja tai lähtöjä, sillä ne pääsevät käsiksi datalohkoihin ja bittimuistiin, joita ohjelmoija voi halutessaan asettaa toimenpiteiden ehdoiksi funktiossa (Kuva 2). Yksi tapa on myös tehdä OB1-lohko FC-lohkoksi, jolloin todellisessa OB1-lohkossa olisi vain 1 kutsu MainFC-lohkoon.



KUVA 2. Yksinkertainen ”ChangeScreen” FC, jolla voidaan pakottaa HMI vaihtamaan aktiivista sivua PLC:n ohjelmasta



KUVA 3. Kuvan 2 funktiota voitaisiin kutsua ohjelmasta esim. tähän tapaan, tuloon targetScreen voisi myös liittää suoraan UInt-tagin

2.3.3 Data Block

Datalohkoissa säilytetään käytettävät muuttujat. Datalohkoihin saadaan samat asiat kuin M (tai STEP5 F, flags) muistialueelle. Näiden lisäksi datalohkoille yksinomaisia ovat esim. taulukot ja structit. Datalohko säilyttää sinne tallennetun tiedon. FC ei säilytä sinne säilytettyä tietoa, mutta FC pääsee käsiksi datalohkoihin, jolloin datalohkoa käytetäänkin hyvin usein juuri esim. FC-lohkon yhteydessä.

Datalohkoissa muuttujille ei tarvitse asettaa osoitteita. Jos luodaan M-muistialueelle bitti, sitä kutsuttaisiin sen osoitteesta, joka voisi olla esim. %M60.0. Vastaavaa bittiä voitaisiin kutsua datalohkosta datalohkon nimellä ja bitin nimellä esim. ”Datalohko1”.bitti.

2.4 Siemens-ohjelmointiympäristöt

Siemensillä on useita versioita useiden vuosien ajalta ohjelmoitavista logiikoista, jonka vuoksi niitä on ohjelmoitavilla ohjelmointiympäristöilläkin jo useita toisistaan erilaisia versioita. Tässä opinnäytetyössä joudutaan käyttämään kolmea eri sukupolven ohjelmointiympäristöä. Nämä ovat käytännössä ohjelmia, jotka voidaan asentaa tietokoneelle. Ohjelmoitava logiikka voidaan siten yhdistää tietokoneeseen ja siihen voidaan tehdä ja ladata ohjelma tietokoneelta käyttäen ohjelmointiympäristöä.

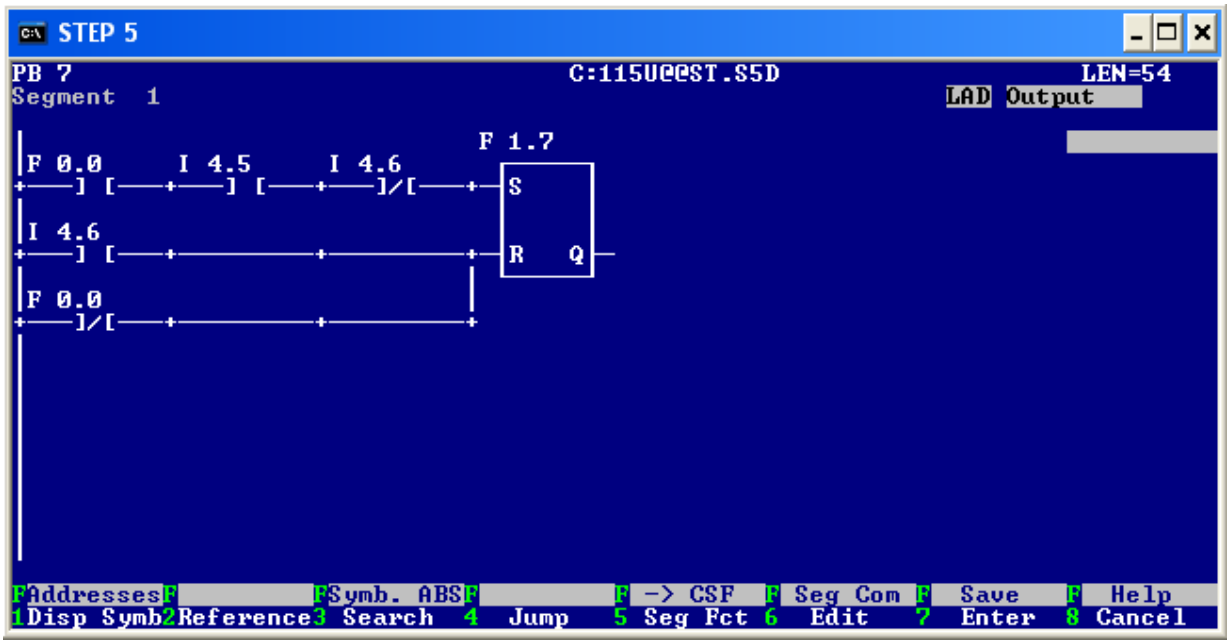
Ohjelmointiympäristöissä käytetään pääasiallisesti samoja kieliä, mutta myös niissä on eroja. Myöskin uudemmat kielet kuten Structured Text ei ole saatavilla vanhimmissa ohjelmointiympäristöissä. Siemens on myös erottanut tiettyjä ominaisuuksia sisällyttämällä ne vain osaan ohjelmointiversioistaan Basic- ja Pro-tasojen mukaisesti. Siemens myös päivittää ympäristöjä kohtuullisen usein sekä julkaisee välillä suuriakin päivityksiä niihin. Monesti näissä tapauksissa uusimmilla versioilla ei pysty ohjelmoimaan vanhempien versioiden ohjelmia, vaan van-

hat ohjelmat pitää ensin päivittää uudelle ohjelmointiympäristön versiolle. Ohjelmointiympäristöjen vanhoilla versioilla ei pysty silloin muokkaamaan uusia ohjelmia lainkaan.

2.4.1 STEP 5

STEP 5 on vanhin tässä työssä käytettävä ohjelmointiympäristö. Se on julkaistu vuonna 1979. Sillä ohjelmoidaan Siemensin STEP 5 -sarjan ohjelmoitavia logiikoita. STEP 5 -ympäristö toimi CP/M-käyttöjärjestelmässä, mutta myöhemmin myös MS-DOS:ssa. Tästä johtuukin ohjelmiston retrohenkisyys, sillä siinä ei ole tukea edes hiirelle. Ohjelmointi ja navigointi tehdäänkin pelkästään näppäimistöllä. Viimeisin versio on kuitenkin julkaistu jopa Windows XP -käyttöjärjestelmälle. Ohjelmaan viimeisimmät päivitykset ovat kuitenkin tulleet vuonna 2006.

Ohjelmointikielinä STEP 5 tukee tikapuulogiikkaa, toimilohkoja sekä käskylistoja (Kuva 4). Käskylistan ansiosta sillä on voitu toteuttaa käytännössä samat asiat kuin nykyisillä korkeantason kielilläkin. Toiminnallisuus on kuitenkin ollut tämän vuoksi hieman epäselvää, etenkin jos ajatellaan vaikka tehtaan kunnossapitohenkilöstöä.



KUVA 4. LAD-kielellä ohjelmointia STEP 5 -ohjelmointiympäristössä

Suuria eroavaisuuksia esiintyy uudempiin ohjelmointiympäristöihin verrattuna etenkin valmiiden kirjastojen ja toimilohkojen suhteen. Esimerkiksi hyvin usein käytettävä ajastin on todella erilainen STEP 5 -maailmassa. SD:tä vastaava ajastin nykyisistä järjestelmistä olisi TON. Ajan syöttäminen SD-ajastimeen on kuitenkin hieman erilaista. Ajan arvo ilmaistaan 16 bitillä, mutta kaikkia bittejä ei käytetä kokonaan ajan ilmaisuun.

```
A       I       20.5
L       KT      360.3
SD      T1
A       T1
=       Q       10.0
```

Yllä olevasta esimerkistä käy hyvin ilmi, että ajastimelle annetaan arvoksi 360.3. Uudemmissa STEP 7 -järjestelmissä tässä voisi olla esim. t#10s, joka vastaa 10 sekuntia. KT 360.3 vastaa kuitenkin yhtä tuntia.

KT:n formaatti on 3 desimaalia BCD:nä syötettynä. BCD on tapa esittää desimaalilukuja 4 bitillä. Heksadesimaalinen esitysmuoto veisi myös 4 bittiä, mutta BCD ei käytä hyväksi ylijääviä bittejä. Tämän takia suurin aika, jota voidaan käyttää, on 999.3 eli 2 tuntia, 46 minuuttia ja 30 sekuntia.

Viimeisimmät 3 puolitavua käytetään siis kolmen BCD-luvun ilmaisuun, ensimmäisellä puolitavulla osoitetaan ajan resoluutiota (0 – 3). Resoluutio osoitetaan myös BCD-luvulla. KT:n formaatti on siis: KT [BCD].[resolution]. Resoluutiossa 0 vastaa 0.01 s aikaa, 1 on 0.1 s, 2 on 1 s ja 3 on 10 s. Todellinen aika saadaan siis kertomalla BCD:n arvo resoluutiota vastaavalla ajalla. Eli 360.3 = 360 * 10 s = 3600 s.

2.4.2 STEP 7 Classic

STEP 7 on tarkoitettu S7-300 ja S7-400 ohjelmoitavien logiikoiden ohjelmointiin. STEP 7 Classic nimenä viittaa STEP 7 -versioihin, jotka ovat V5.6 tai vanhempia. Tämä ohjelmointiympäristö on suuri päivitys aikaisempaan STEP 5 -ympäristöön.

Se julkaistiin vuonna 1995. Ohjelmointikieliin tässä on lisätty Structured Text, joka helpottaa ja nopeuttaa etenkin monimutkaisen toiminnallisuuden ohjelmointia. Samalla ympäristössä on parannettu etenkin diagnostiikkaominaisuuksia. Ohjelmoitavan logiikan ohjelmaa voi seurata reaaliajassa ohjelmointiympäristöstä. Ohjelmoitavia logiikoita on myös mahdollista simuloida suoraan ohjelmointiympäristöstä, jolloin ohjelmoitavaa ohjelmaa voidaan testata jo ohjelmointivaiheessa. Ohjelmointiympäristöä voisi samalla pitää helppokäyttöisenä, etenkin verrattuna STEP 5 -ympäristöön.

2.4.3 TIA Portal

TIA Portal on kaikista uusin järjestelmä, jota Siemensillä on tarjota. Se pohjautuu samaan STEP 7 -järjestelmään kuin STEP 7 Classic. Sillä on kuitenkin täysin uusi kehikko, joka julkaistiinkin vuonna 2009. STEP 7 V10.5 tai TIA Portal (Totally Integrated Automation) V10.5. TIA Portalia on kehitetty aktiivisesti ja siitä onkin saatavilla jo useita versioita. Tuorein versio on TIA V16, joka julkaistiin 2019.

TIA Portalilla voidaan ohjelmoida myös vanhempia S7-300- ja S7-400-logiikoita, mutta ei täysin kaikkia vanhimpia malleja. TIA Portal onkin kehitetty ja tarkoitettu etenkin uusille S7-1200- ja S7-1500-sarjoille. Tässä opinnäytetyössä järjestelmä modernisoidaan TIA Portal -ohjelmointiympäristölle.

TIA Portalin suurin eroavaisuus vanhempiin STEP 7 -versioihin verrattuna on sen yhtenäisyys. Vanhemmissa versioissa useat eri osat olivat jaettuina eri ohjelmien alle, joiden välillä ei välttämättä voinut nopeasti ja helposti kopioida ja siirtää tietoa. TIA Portalissa kaikki on yhdistetty samaan pakettiin. Kaikki toiminnallisuus on yhden yhtenäisen editorin alla. Varjopuolena TIA Portal on raskaampi sovelluksena, jolloin se vaatii myös suorituskykyä enemmän.

3 MODERNISOITAVA KOHDE

Tässä opinnäytetyössä modernisoitava kohde on yksittäinen metallivalimon osa. Kohteen ohjauksessa käytetään tällä hetkellä Siemensin S5-sarjan ohjelmoitavaa logiikkaa. Opinnäytetyössä tehdään suunnittelu tämän vanhemman ohjelmoitavan logiikan modernisoinnista eli sen päivityksestä tuoreempaan Siemens S7-sarjan ohjelmoitavaan logiikkaan.

Opinnäytetyössä modernisoidaan Siemensin S5-sarjan automaatiojärjestelmä tuoreempaan S7-sarjan järjestelmään. SIMATIC S5-sarja tuli markkinoille vuonna 1979 (7).

S5-sarjan automaatiojärjestelmät olivat siihen aikaan edistyksellisiä ohjausjärjestelmiä. Nykyään laitteet ovat kuitenkin jo aikansa eläneitä, eikä niitä enää valmisteta. Tämä onkin yleisin syy modernisointiin, sillä nykyiseen järjestelmään ei valmisteta enää varaosia. S5-sarjan tuoteperheeseen kuuluivat alustat 90U, 95U, 101U, 100U, 105, 110, 115, 115U, 135U ja 155U (6). Mitä korkeampi numero on, sitä hienostuneempi ja kalliimpi järjestelmä oli. Tässä työssä modernisoitava alusta on 115U.

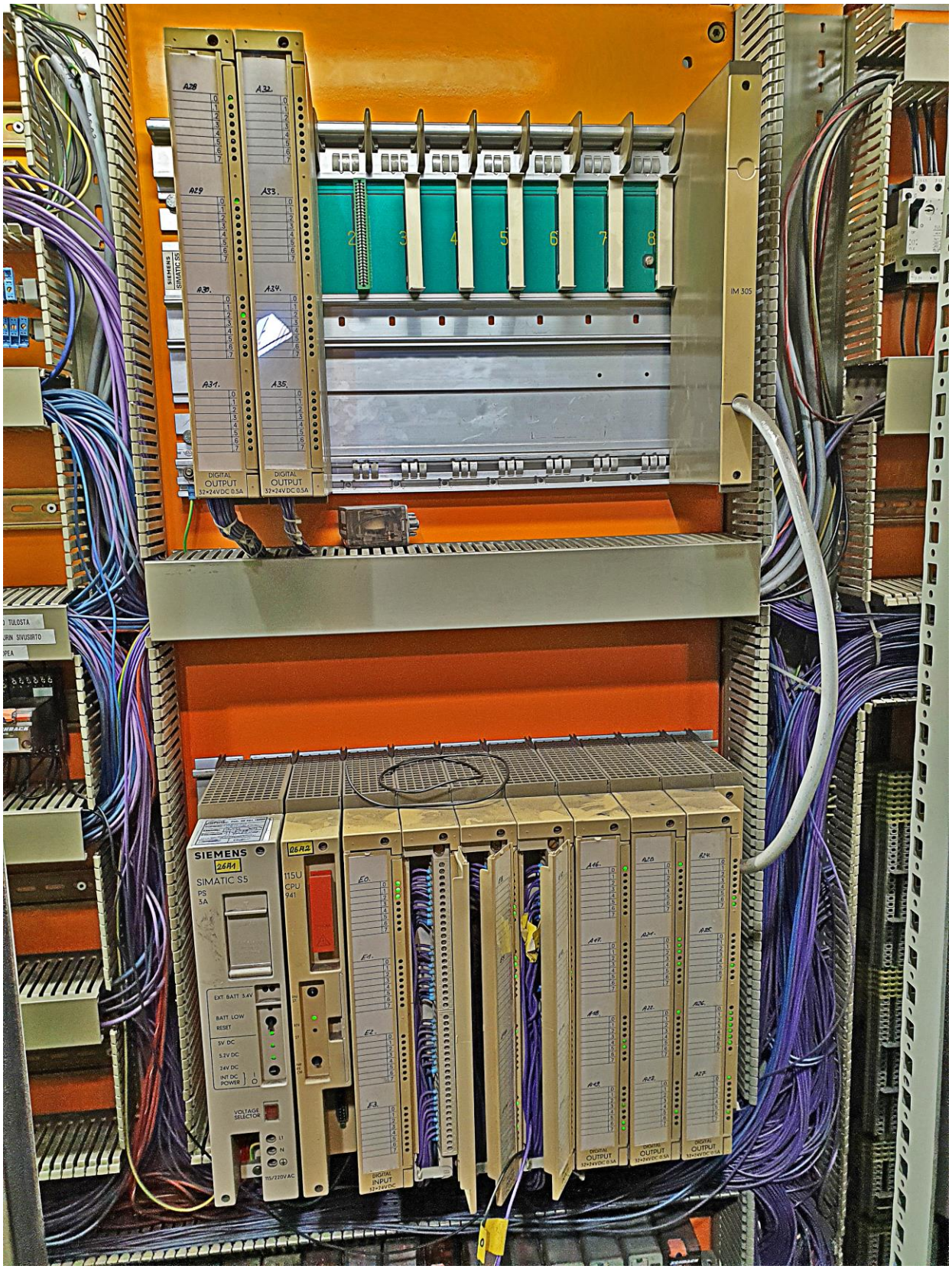
3.1 Nykyinen laitteisto

Modernisoitava ohjelmoitava logiikka tässä kohteessa on S5-115U (Kuva 5). Tätä ohjelmoitavaa logiikkaa on lähtökohtaisesti suunnattu keskitason kohteisiin. Tämä ohjelmoitava logiikka on modulaarinen. Modulaarisuus tarkoittaa sitä, että järjestelmään voidaan liittää ja siitä voidaan irrottaa juuri kohteeseen tarvittavia moduuleita. Moduulit asennetaan kehikkoon, joka pitää ohjelmoitavan logiikan kasassa.

Näihin moduuleihin kuuluvat esimerkiksi virtalähteet, prosessorit, tulo- ja lähtömoduulit, erilaisia kommunikointirajapintamoduuleita sekä laajennusmoduuleita järjestelmän laajentamista varten.

Yksinkertainen S5-sarjan ohjelmoitava logiikka koostuisi kehikosta, johon on asennettu virtalähde, CPU sekä I/O-moduuleita. Kehikossa on kuitenkin rajallinen määrä paikkoja moduuleille (slots). Tätä varten järjestelmää on mahdollista laajentaa laajennuskehikoilla.

Kun käytetään useampaa kehikkoa, voidaan puhua keskitetystä sekä hajautetusta järjestelmästä. Kehikkoa, jossa CPU sijaitsee, kutsutaan yleisesti keskikehikoksi (CC, Central Controllers) ja muita kehikoita laajennuskehikoiksi (EU, Expansion Units).



KUVA 5. Modernisoitava kohde

3.1.1 Virtalähde

Virtalähde on yksi moduuli, joka tulee kiinni CPU:n viereen varaten yhden slotin I/O-kehikolta. Kohteessa oleva virtalähde on 6ES5 951-7LB14, jonka syöttö on 3 ampeeria. Virtalähde antaa virran prosessorille sekä tarvittaessa myös moduuleille. Lähtökorteilla on yleisesti kuitenkin oma virtalähde. Virtalähteessä on myös varavirtalähde, jonka tehtävä on estää CPU:n RAM-muistin tyhjeneminen. Jos tarvitaan useita lisäkehikoita, tulee yleensä tarpeen hankkia lisäkehikoille omat virtalähteet, etenkin jos puhutaan hajautetusta konfiguraatiosta.

3.1.2 Prosessori

Prosessoria voitaisiin kutsua ohjelmoitavan logiikan ”aivoiksi”. Prosessorissa sijaitsee ohjelma sekä muisti. S5-115U sarjan ohjelmoitavaan logiikkaan on saatavilla viisi erilaista prosessoria. Prosessorit eroavat toisistaan lähinnä ominaisuuksien mukaan, kuten muistin ja suoritusnopeuden. Prosessori myös rajoittaa siihen kytkettävien I/O-moduulien määrää.

Tässä kohteessa on käytössä 941 CPU, joka on suoritusteholtaan ja muistiltaan huonoin S5-115U-sarjan prosessoreista. Siitä huolimatta se on todella riittävä tähän kohteeseen. CPU 941:ssä on yhteensä 18 kilotavua muistia ohjelmaa varten. Prosessorissa on sisäinen 2 kilotavun keskusmuisti (RAM), lisäksi siinä on 16-kilotavuinen EPROM-muisti. EPROM-muistin erikoisuus on sen tyhjentäminen. Se täytyy tyhjentää osoittamalla muistin kvartsilasiin UV-valoa, jolloin piiri tyhjenee. Yksi käsky vie muistia yleisesti kahden tavun verran. Muistia prosessorilla olisi siis hieman yli 9000 yksittäiselle käskylle. (6, s. 37.)

```
------[ ]--------------[ ]----------------( )
     I 0.0            I 0.1              Q 1.0
```

Esimerkiksi yllä oleva LAD-kielellä toteutettu ohjelma veisi muistia 6 tavun verran, sillä siinä on 3 käskyä. Muistin käyttöä voidaan ymmärtää paremmin, jos tarkastellaan, miltä yllä oleva esimerkki näyttäisi suoraan konekielellä. Käännetään se ensiksi käskylistaksi.

```
A       I       0.0
A       I       0.1
=       Q       1.0
```

Konekielellä käskyt käsitellään 16-bittisinä, eli 2-tavuisina. Näihin 2 tavuun saadaan mahtumaan pääsääntöisesti aina jokainen käsky. Alla on edellä oleva esimerkki puhtaasti konekielellä.

```
C000H           // 1100_0000_0000_0000
C100H           // 1100_0001_0000_0000
D881H           // 1101_1000_1000_0001
```

Konekielessä jokainen operaatiokoodi yhdistettynä operandiinsa edustavat yhtä madollista arvoa. A I 0.0, olisi $C000_H$, mikään muu operaatiokoodi ja operandi eivät voi olla tätä samaa. A I 0.5 olisi $C500_H$ ja A I 15.7 olisi $C70F_H$. Tästä myös johtuvat rajoitteet esimerkiksi tulojen määrässä. Tähän prosessoriin on mahdollista ohjelmallisesti lisätä 1024 tuloa, joiden väli olisi 0.0 – 127.7.

Konekielellä käskyt muodostuvat neljästä puolitavusta, jolloin sitä voidaan ilmaista heksadesimaalilla. Esimerkiksi $FFFF_H$ tarkoittaa käskylistalla käskyä NOP 1. Vastaavasti jo esimerkkinäkin käytetty A I olisi konekielellä $C0_b0_a0_a$, jossa *b* osoittaa tulon bittiosoitetta ja *a* osoittaa tavuosoitetta. $C7FF_H$ olisi A I 127.7, eli suurin osoitettava tulo. $CFFF_H$ olisi puolestaan O I 127.7, koska O I 0.0 olisi $C800_H$. (6, s. 619.)

Käsky voi myös olla 32-bittinen, se kuitenkin osoittaa, että käskyssä on mukana 16-bittinen arvo. Esimerkiksi ajastimissa käytettävä käsky, jolla määritellään odotettava aika, on L KT 000.0. Tämä kääntyisi konekielelle muotoon $3002_H0_e0_e0_e0_e$, jossa *e* tarkoittaa 16-bittistä arvoa. Jotkut 16-bittiset käskyt myös sisältävät 16-

bittisen arvon osoittamalla siihen, esimerkkinä L IW 10 ($520A_H$). 16-bittiset arvot voivat olla kokonaislukuja (-32768 – 32767), heksadesimaalilukuja (0000 – FFFF), BCD-lukuja (0000 – 9999) sekä erilaisia bittikuvioita.

32-bittisten käskyjen käsittely voi viedä jopa yli 50-kertaisen ajan verrattuna 16-bittiseen käskyyn, jossa on vain totuusarvomuuttujia. Tämä johtuu siitä, että nämä käskyt joudutaan käsittelemään prosessorin akkurekisterissä (Accumulator). Lähtökohtaisesti kaikki käskyt, jotka liittyvät muihin kuin totuusarvomuuttujiin, käsitellään akkurekisterissä.

Tässä prosessorissa akkurekisterejä on todellisuudessa kaksi kappaletta. Näitä kutsutaan ACCU 1 ja ACCU 2. ACCU 2 on rekisteri, johon ei voida syöttää suoraan tietoa. Sieltä ei voida myöskään lukea suoraan tietoa, mutta siinä olevaa tietoa voidaan käyttää apuna esimerkiksi aritmeettisissa käskyissä. Tämä tarkoittaa sitä, ettei käsky pysty siirtämään operandinsa osoittamaa arvoa ACCU 2 -rekisteriin, vaan se aina siirtyy ACCU 1 -rekisteriin. Tätä ennen kuitenkin tapahtuu automaattisesti siirto ACCU 1 -rekisteristä ACCU 2 -rekisteriin. Tällä tavoin prosessorilla on rekisterissä aina 2 viimeisintä käsiteltävää 16-bittistä arvoa.

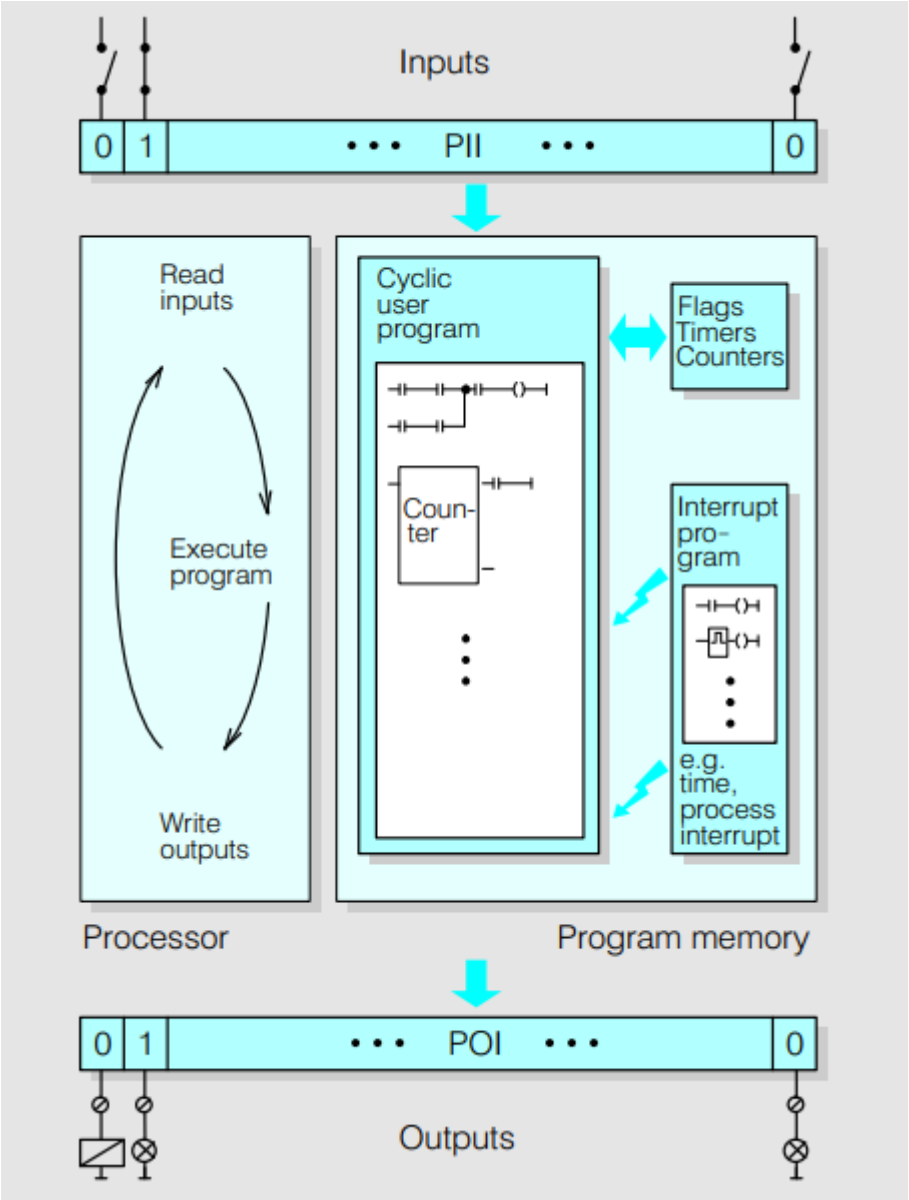
```
L       KF 81   // Siirretään ACCU 1 -> ACCU 2, Siirretään 81 -> ACCU 1
L       KF 339  // Siirretään ACCU 1 -> ACCU 2, Siirretään 339 -> ACCU 1
+F              // ACCU 2 (81) + ACCU 1 (339) = ACCU 1 (420)
T       FW 0    // Kopioidaan ACCU 1 arvo -> FW 0 (Flag word)
```

Yllä on esimerkki, jossa suoritetaan yhteenlasku akkurekistereillä. Suorittaessa esimerkiksi yhteen- ja vähennyslaskuja, talletetaan tulos myös aina ACCU 1 -rekisteriin.

Muistin suhteen on hyvä huomioida, ettei esimerkiksi ohjelmassa olevia kommentteja tai tulojen/lähtöjen nimiä säilytetä prosessorin muistissa. Näiden täytyy olla erillisellä tiedostolla ohjelmointiin käytettävällä alustalla.

S5-115U ohjelmoitava logiikka pohjautuu pitkälti prosessorin ja muistin toiminnallisiin funktioihin (Kuva 6). Prosessori toimii syklisesti eli se toimii kolmessa eri

askeleessa silmukassa. Jokaisen silmukan alussa prosessori lukee tulosignaaleiden tilat ja tallentaa ne prosessin tulojen muistikuvaan (PII, Process Input Image). Muistikuvassa olevat arvot eivät ole reaaliaikaisia, vaan niiden arvot ovat kopioita arvoista lukuhetkellä. Kun muistikuva on luotu, voidaan alkaa suorittamaan muistissa olevaa ohjelmaa. Tämä ohjelma pääsee käsiksi nyt PII-muistikuvassa oleviin tietoihin.



KUVA 6. S5-115U toimintaperiaate (8)

Ohjelma käydään lävitse askel askeleelta, jolloin siinä tehdään kaikki tarvittavat laskennat ja toimenpiteet. Silmukka voidaan pysäyttää tai keskeyttää esim. tiet-

tyjen ehtojen mukaisesti. Lopuksi ohjelmasta saadaan prosessin lähtöjen muistikuva (POI, Process Output Image). Prosessori pystyy nyt tämän mukaisesti siirtämään ohjelman toimenpiteiden mukaiset arvot lähdöille. Tämän jälkeen silmukka alkaa alusta.

3.1.3 I/O-moduulit

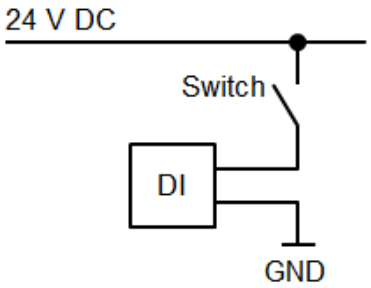
S5-115U ohjelmoitavaan logiikkaan oli saatavilla iso kirjasto erilaisia I/O-moduuleita. Näihin kuuluivat digitaaliset tulo- ja lähtömoduulit. Nämä koostuvat 8-, 16- ja 32-kanavaisista moduuleista. Analogisia tulo- ja lähtömoduuleita oli myös saatavilla. Analogisia moduuleita sai tulopuolella 8- sekä 16-kanavaisina, mutta lähtöpuolella vain 8-kanavaisina.

On myös olemassa hybridikortteja, jotka ovat 32-kanavaisia, mutta sisältävät 16 tuloa ja 16 lähtöä. Näissä erikoisuutena on se, että osoitteina on käytettävä samaa aluetta tulojen ja lähtöjen kanssa. Esimerkiksi tulot I 0.0 – I 1.7 ja lähdöt Q 0.0 – Q 1.7 olisivat saman kortin alla, eikä niitä voisi vaihtaa esim. tulot 0.0 – 1.7 ja lähdöt 5.0 – 6.7 alueille.

Lähtökortteja on saatavilla myös erilaisilla tekniikoilla, yleisimpiä ovatkin rele- ja transistorikortit. Relekorttien etuna on suuremman kuorman käyttäminen. Relekortteja voi tasajännitteen lisäksi käyttää vaihtojännitteellä, joka mahdollistaa hieman erilaisten kohteiden käyttämisen. Transistorikortit pääsääntöisesti toimivat 24 V:n tasajännitteellä.

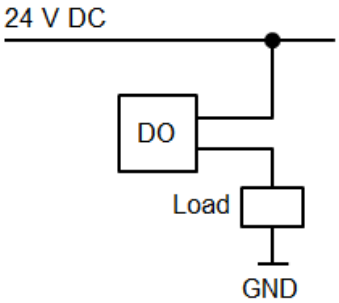
Opinnäytetyössä modernisoitava kohde on kuitenkin I/O-moduuleiden kannalta yksinkertainen. Käytössä on vain kahta eri moduulityyppiä. Tulokortteja on 4 kpl ja ne ovat 32-kanavaisia. Lähtökortteja on 5 kpl ja ne ovat myös 32-kanavaisia. Kaikki kortit ovat ns. digitaalisia eli tieto ilmaistaan yhden bitin tarkkuudella (1 / 0, tosi / epätosi). Kohteen I/O-määräksi tulee kuitenkin melko suuri: 128 tuloa ja 160 lähtöä.

Tulokorteille tuodaan miinus, jolloin kytketyt laitteet tuovat plussan kortille kentältä. Tulokortit eivät tarvitse omaa virtalähdettä, vaan toimivat prosessorin taustalevyn virralla. Kun tulokorttiin liitetty laite aktivoituu, päästää se kortille lävitse 24 volttia. Tätä voidaan kutsua sinking inputiksi, jossa kytkin sijaitsee kortin ja plussan välissä (Kuva 7).



KUVA 7. Sinking input, modernisoitavassa kohteessa tulokortit toimivat näin (9)

Lähtökortteihin pitää tuoda virta ulkoisesti. Näihin olisi mahdollista käyttää prosessorin virtalähdettä, mutta yleensä näille on vielä oma ulkoinen virtalähde, etenkin jos lähtökortteja on paljon. Lähtökortille tuodaan plussa suoraan kortille, jolloin kytketyt laitteet tuovat miinuksen kentältä. Laitteet sijaitsevat kortin ja miinuksen välillä. Kun ohjelmoitavassa logiikassa tahdotaan aktivoida lähtöön kytketty laite, päästää se laitteelle 24 volttia. Tätä voidaan kutsua sourcing outputiksi (Kuva 8).



KUVA 8. Sourcing output, modernisoitavassa kohteessa lähtökortit toimivat näin (9)

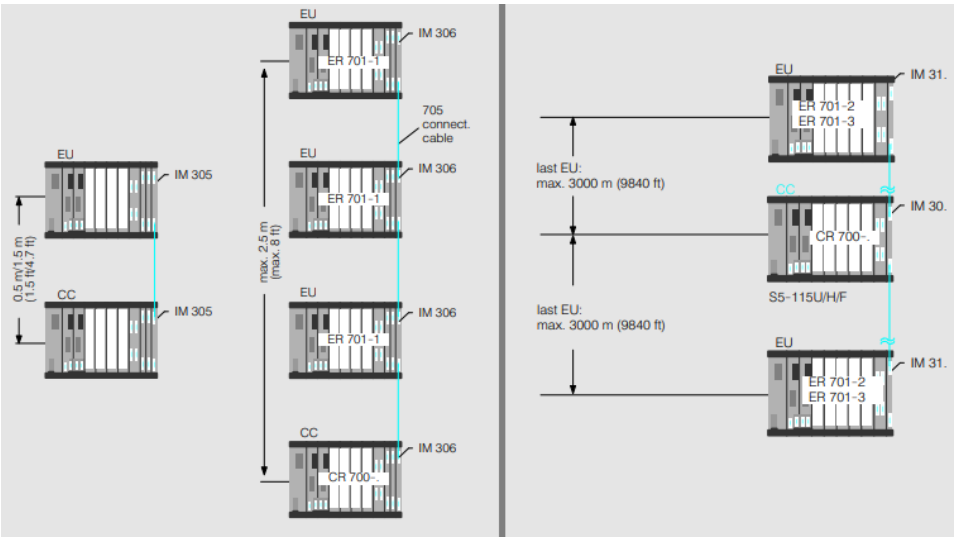
Sourcing input ja sinking output toimivat samalla tavalla, mutta laite sijaitsee juuri toisella puolella piiriä. Tällaiset kytkennät eivät ole niin yleisiä, mutta vastaavia saattaa tulla vastaan etenkin modernisoinneissa. Tämän vuoksi pitää osata valita

oikein toimivat kortit kohteeseen tai kytkentöjä voi joutua muuttamaan todella suuresti. Yleensä tulokortteihin voidaan tehdä kytkentä molemmilla tavoilla, vaihtamalla plus miinukseen ja miinus plussaan kortilla.

3.1.4 Kehikot ja laajennus

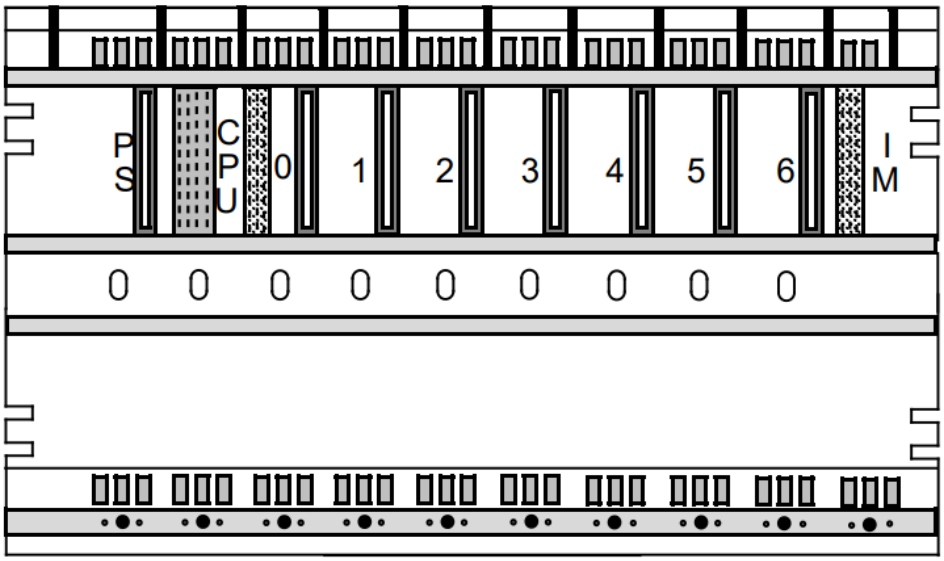
Kuvan 5 mukaisesti kohteessa on 2 eri I/O-kehikkoa. Alempi kehikko on ns. keskikehikko (CC, Central Controllers) ja ylempi laajennuskehikko (EU, Expansion Unit). Tätä konfiguraatiota kutsutaan keskitetyksi I/O:ksi. Periaatteessa se tarkoittaa sitä, että ohjelmoitava logiikka sijaitsee yhdessä paikassa. Olisi kuitenkin mahdollista, että yksi kehikko olisi keskikehikosta jopa kilometrien päässä. Tällaista konfiguraatiota kutsutaan hajautetuksi I/O:ksi.

Kehikot yhdistetään toisiinsa liitäntämoduuleita käyttäen (IM, Interface Module). Kohteen konfiguraatiossa on käytössä IM305 moduulit, jotka mahdollistavat yhden laajennuskehikon lisäämisen konfiguraatioon. IM306 on toinen moduuli, joka mahdollistaisi jopa 3 laajennuskehikon lisäämisen sarjassa keskitettyyn konfiguraatioon (Kuva 9).



KUVA 9. Vasemmalla keskitetty konfiguraatio, oikealla hajautettu (8)

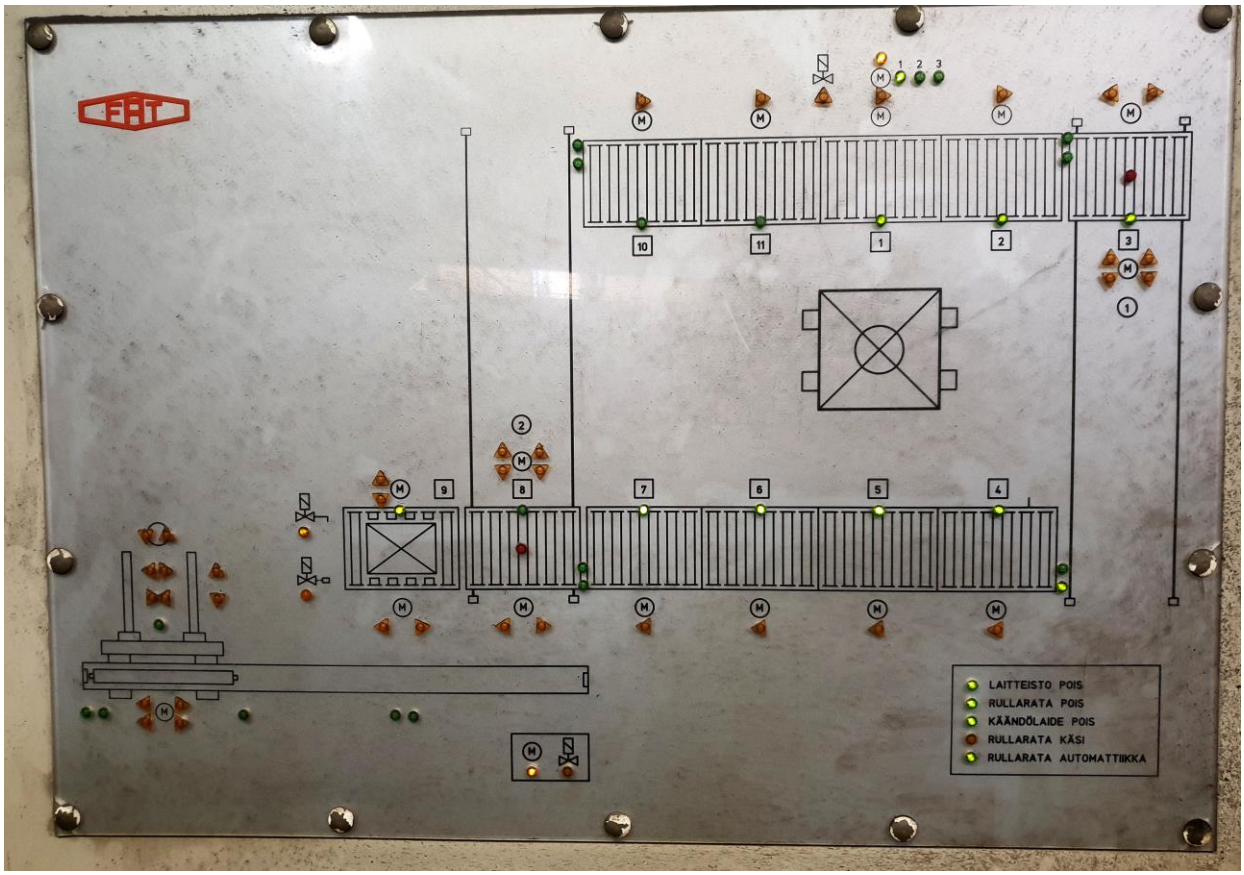
Käytettävä kehikko määrää lähtökohtaisesti, mitä moduuleita siihen voidaan liittää. Kehikot koostuvat alumiinisesta kiskosta, johon kaikki moduulit saadaan kiinnitettyä mekaanisesti. Kehikossa on myös yksi tai kaksi taustalevyä, jotka yhdistävät kehikkoon liitetyt moduulit sähköisesti (Kuva 10).



KUVA 10. Kohteen keskikehikko (6)

3.1.5 Muut laitteet

Laitteina on myös LED-valotaulu, jossa näkyy ohjattavan prosessin tilanne. Tämän osoitus on toteutettu yksittäisillä LED-valoilla, joista jokainen on kytketty aina yhteen lähtöön I/O-moduuleilla (Kuva 11).



KUVA 11. Käytössä oleva LED-taulu

Tässä LED-taulussa on käytössä yhteensä 77 lähtöä, joka on huomattavan suuri määrä. Tämä tarkoittaa siis ainakin kolmea 32-kanavaista lähtökorttia. Yksi mahdollisuuksista olisi korvata kyseinen laite nykyaikaisella HMI-paneelilla, jolloin 77 lähtöä saataisiin vaihdettua Profinet-väylään. Siemensin uudemmissa ohjelmoitavissa logiikoissa on Profinet-tuki vakiona, jolloin se voisi olla edullisempi vaihtoehto kuin korvata 3 I/O-moduulia.

3.2 Mahdolliset korvaajat

Asiakkaan kaikki tuotantolinjastot on toteutettu Siemensin järjestelmillä, joten muihin valmistajiin ei tässä tutustuta. Tämän vuoksi esimerkiksi kunnossapitohenkilöstön ei tarvitse opetella usean valmistajan järjestelmiä. Uusi korvaava laitteisto voi periaatteessa olla kolmea eri sarjaa Siemensiltä. Nämä sarjat olisivat S7-300, S7-1200 ja S7-1500. Sarjojen välillä on suuria eroja, vaikkakin kaikki kyllä soveltuisivat kohteeseen. Koska kohteessa ei ole kovin suurta I/O-määrää,

voi järkevin valinta olla edullisin vaihtoehto. Toisaalta korvaavalta laitteistolta voisi toivoa esim. Profinet-tukea, joka mahdollistaisi siihen liitettävän HMI-paneelin. Tällä tavalla I/O-määrää voitaisiin huomattavasti laskea, jos LED-valotaulu korvattaisiin tällä paneelilla. Profinet-väylä toisi muitakin etuja, kuten mahdollisen etäkäyttömahdollisuuden ja laajentamisen tulevaisuudessa esim. hajautettuun konfiguraatioon.

Kustannuksia tulee laitteiston lisäksi asennustöistä. Jokainen I/O täytyy yksitellen kytkeä uusille I/O-moduuleille. Jos I/O-järjestystä tulisi vaihtaa, tarkoittaisi tämä myös suuria muutoksia nykyisiin kytkentäpiirustuksiin, joihin modernisoinnissa on myös tulossa merkittävät muutokset. Tämän vuoksi on hyvä alkuun vertailla useaa mahdollista modernisoinnin konfiguraatiota, jonka jälkeen voi tehdä helposti päätöksen valittavasta laitteistosta esim. kustannuksien perusteella.

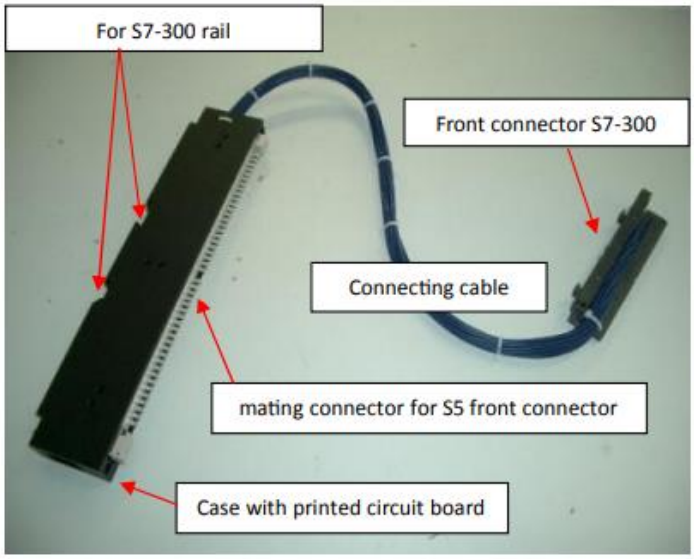
Modernisoinnin yhteydessä ei tule toiminnallisia muutoksia ohjelmaan, jonka vuoksi uuden laitteiston on järkevintä myös olla Siemensin toimittama. Tällä tavoin vanhaa ohjelmistoa voidaan käyttää uudessa järjestelmässä siltä osin kuin se on yhteensopiva. Tämä säästää puhtaasti suunnittelukuluissa ohjelmoinnin kannalta.

3.2.1 S7-300

S7-300 on Siemensin ohjelmoitavien logiikoiden vanhempi sarja. Se esiteltiin markkinoille jo vuonna 1994. Se on edelleen laajasti käytössä oleva. Osaa sarjan ohjelmoitavista logiikoista ei enää valmisteta. Tästä huolimatta sarjaa on vielä laajasti saatavilla päivitetyillä malleilla. Edullisimmista logiikoista ei kuitenkaan yleisesti löydä esim. Profinet-tukea. Prosessori myös määrää paljonko laitteeseen on mahdollista liittää I/O-moduuleita ja tuen mahdollisille I/O-kehikoille.

Ainoita etuja, miksi kohteeseen voitaisiin harkita S7-300-sarjaa, on se, että S5 on helppo päivittää S7-300-sarjaan. Sovelluspuolelle on valmiita ohjelmia, jotka melko hyvin onnistuvat kääntämään ohjelman STEP 5:lta STEP 7:lle. Samalla on saatavilla erilaisia adaptereita, jotka voidaan kiinnittää nykyisiin S5 I/O-kortteihin,

jolloin vanhaa järjestelmää ei tarvitsisi purkaa, eikä uudelle järjestelmälle tehdä johdotuksia (Kuva 12).



KUVA 12. S5-115U -> S7-300 adapteri (10)

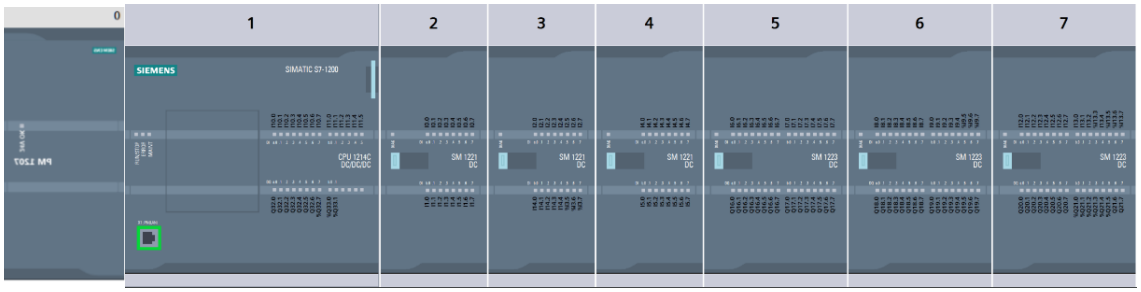
Adaptereja tarvitsisi jokaiselle kortille aina oman, mikä lisäisi itsessään kustannuksia, mutta säästäisi huomattavasti asennuskustannuksista. Toisaalta tällä konfiguraatiolla pitäisi jättää LED-valotaulu ennalleen, mikä tarkoittaisi, että suuremman I/O-määrän vuoksi tarvitaan toinen I/O-kehikko. Kuvassa 13 on esimerkkiehdotus mahdollisesta uudesta laitteistosta S7-300-sarjalla (Kuva 13).

KUVA 13. Modernisointi S7-300 hardwarella

Laitteistoon kuuluisi prosessori, jossa ei ole Profinet-tukea, mutta joka kuitenkin tukisi toista I/O-kehikkoa, jolloin kaikki nykyiset I/O:t saataisiin kytkettyä siihen. Uudessa konfiguraatiossa olisi 5 kpl 32-kanavaisia lähtökortteja ja 4 kpl 32-kanavaisia tulokortteja. Tämän lisäksi mukana olisi laajennusmoduuli IM 360 ohjauskehikossa ja IM 361 laajennuskehikossa. Tämän mukainen konfiguraatio ei ole kustannuksiltaan kovin kallis, sillä asennuksissa voitaisiin säästää ja mahdollisesti myös ohjelman kääntämisessä.

3.2.2 S7-1200

S7-1200 on Siemensin tarjoama uudempi sarja. Se on tullut markkinoille vuonna 2009. Tyypilliset kohteet tällä ohjelmoitavalla logiikalla ovat ehdottomasti pienemmät prosessit ja koneet. S7-1200 ei ole laajennettavissa samoin kuin muut tässä työssä verrattavat ohjelmoitavat logiikat. Tähän sarjaan on kuitenkin saatavilla Profinet-tuki vakiona, joten tässä konfiguraatiossa voidaan vaihtaa LED-taulu HMI-paneeliin, joka säästäisi 77 lähtöä. Tällä säästöllä voitaisiin välttää toinen I/O-kehikko kokonaisuudessaan, johon S7-1200-sarjalla ei olisi tukea muutenkaan. Samalla korttien määrää voitaisiin vähentää, mikä luo säästöä kustannuksiin (Kuva 14).



KUVA 14. Modernisointi S7-1200 hardwarella

Tällä konfiguraatiolla saataisiin siis tarvittavien komponenttien määrää selvästi alemmas. S7-1200-sarja on tällä hetkellä edullisin Siemensin tarjoama sarja, mutta johdotukset pitäisi uusia suurilta osin. Lisäksi HMI-paneeli lisää kustannuksia (Kuva 15).

KUVA 15. S7-1200-sarjaan lisättävä HMI-paneeli KTP900 Basic (11)

Tähän ei myöskään tarvitse erillisiä kehikoita, vaan laitteisto voidaan asentaa tavalliseen DIN-kiskoon. Konfiguraatioon tulisi mukaan ns. hybridkortteja, joissa on 16 tuloa ja 16 lähtöä (Taulukko 3). S7-1200-sarjaan ei ole saatavilla muita 32-kanavaisia kortteja. Samalla prosessori ei tue kuin korkeintaan 8 liitettävää I/O-moduulia. Tämän vuoksi tähän konfiguraatioon ei kannata edes miettiä tilannetta, jossa LED-taulu pidettäisiin ennallaan, koska I/O-määrä menisi prosessorin tukeman määrän rajoille. Prosessorissa on itsessään 14 tuloa ja 10 lähtöä, joita voidaan myös käyttää hyväksi.

TAULUKKO 3. PLC:n konfiguraatio S7-1200 hardwarella

Slot 0	Virtalähde PM1207
Slot 1	Prosessori 1214C
Slot 2	16-kanavainen tulokortti
Slot 3	16-kanavainen tulokortti
Slot 4	16-kanavainen tulokortti
Slot 5	32-kanavainen hybridkortti (DI/DO)
Slot 6	32-kanavainen hybridkortti (DI/DO)
Slot 7	32-kanavainen hybridkortti (DI/DO)

Ohjelmallisesti S7-1200:ssä on pieniä rajoitteita, kuten että se ei tue ollenkaan SL-ohjelmointikieltä, eikä myöskään vanhat S5-ajastimet ole tuettuina tässä sar-

jassa. Tämä voi aiheuttaa pieniä ongelmia ohjelman käännössä, sillä pääosa modernisoitavaa S5-ohjelmaa on tehty SL-kielellä. Tämä on kuitenkin huomattavasti halvempi vaihtoehto kuin ehdotettu konfiguraatio S7-300-sarjalla.

3.2.3 S7-1500

S7-1500 on uusin ja ominaisuuksiltaan parhain ohjelmoitava logiikka, mitä Siemensillä on tällä hetkellä tarjota. Se tuli markkinoille vuonna 2013. Kohteeseen se on tavallaan liian hyvä, eikä kaikille sen hienoille ominaisuuksille olisi käyttöä ollenkaan. Sarjassa on kuitenkin tuki S5- ja S7-sarjan vanhoille ominaisuuksille, lähinnä modernisointien ja yhteensopivuuksien kannalta. Tämän vuoksi ohjelma kääntyisi tälle sarjalle yhtä hyvin kuin S7-300-sarjallekin. Etuna tässä on Profinet-tuki vakiona, mikä uupui S7-300-sarjasta. Tämän vuoksi tässä voidaan harkita samaa konfiguraatiota kuin S7-1200-sarjassa. Toisaalta komponenttien hintataso on tässä samaa luokkaa kuin S7-300-sarjassa.

Näissä uudemmissa logiikoissa on myös erikoisuutena niihin integroitu näyttö ja näppäimet, joilla logiikkaa voidaan hallita käytön aikana. Tähän kohteeseen tämän sarjan logiikat ovat kuitenkin liian järeitä. Onkin selvää, ettei tätä ole edes tarkoitettu näin pieniin kohteisiin, vaikkei se olisi tietenkään mikään este. S7-1500-sarjalla toteutettu konfiguraatio tulisi laitteiston kannalta halvemmaksi kuin S7-300-sarjalla toteutettu, mutta se ei pärjää hinnassa S7-1200-sarjalle.

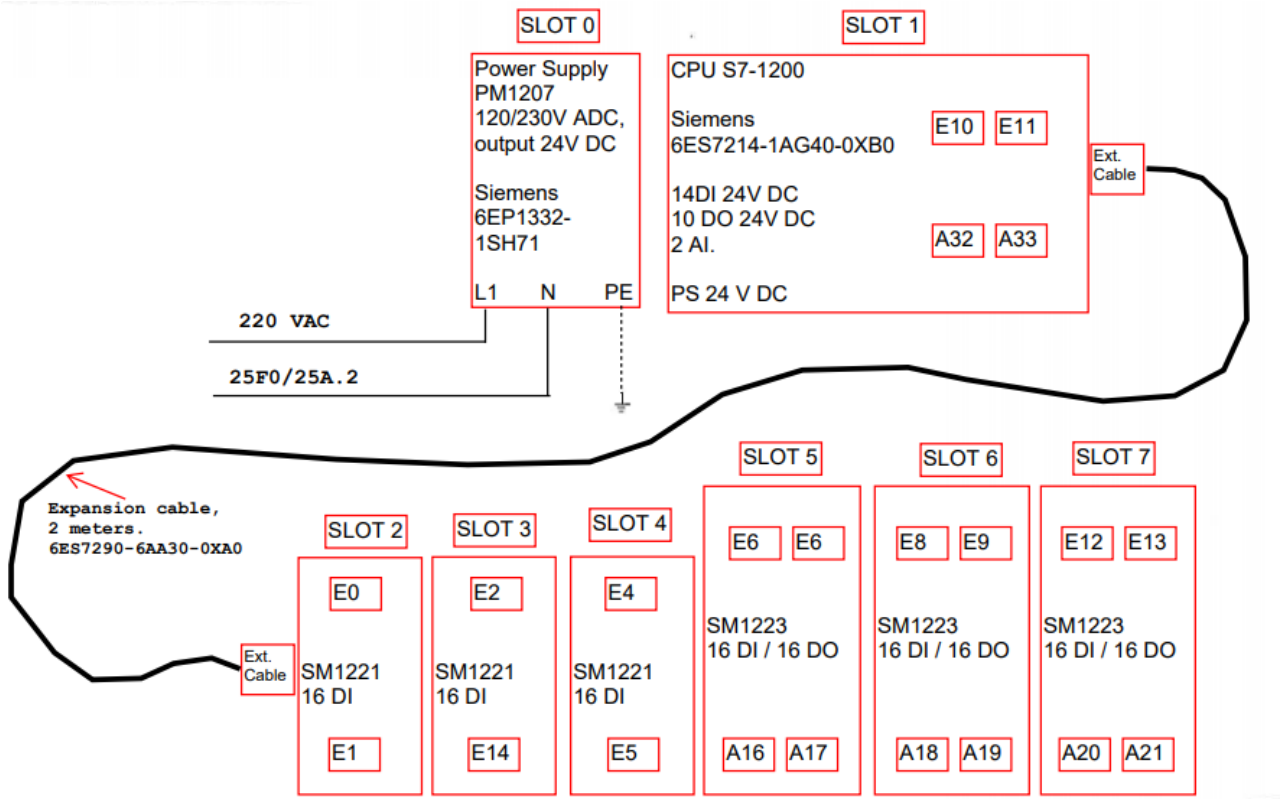
3.3 Korvaava hardware

On todella vaikea arvioida täysin tarkasti kustannuksia eri sarjojen välillä, mutta S7-1200-sarja vaikuttaa suoraan parhaimmalta valinnalta. Koska kohde on todella vanha, eivät sen tekniset vaatimuksetkaan nouse esteeksi. Nykyisessä prosessorissa on 18 kilotavua muistia ohjelmaa varten. Kaikissa tässä mietityissä vaihtoehdoissa ohjelmamuistia olisi vähintään puolet enemmän.

S7-1200 ohjelmoitava logiikka paneelin kanssa on yli puolet edullisempi ratkaisu kuin esimerkiksi S7-300-sarjalainen. S7-1500 ei ole myöskään paljoa edullisempi ratkaisu kuin S7-300.

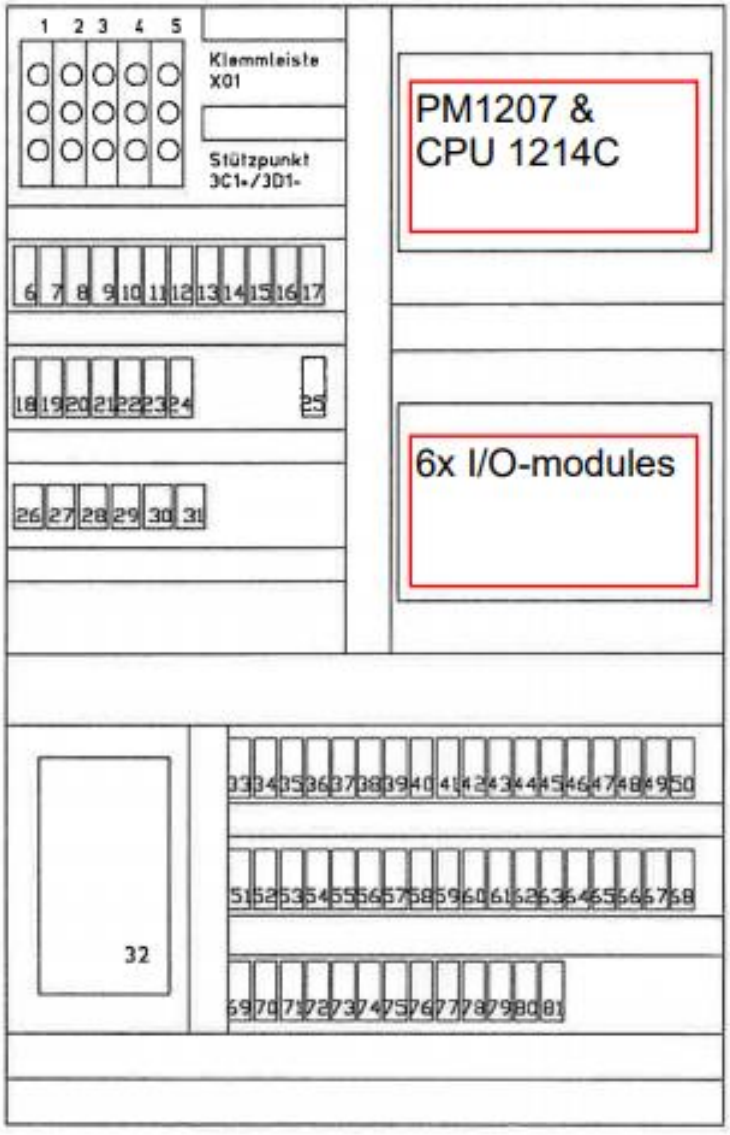
Jos kohde modernisoidaan S7-1200-järjestelmällä, tulee kustannuksia kuitenkin hieman lisää asennustöistä ja ohjelmistotöistä, joita HMI-paneeli vaatii. Nämä kustannukset eivät kuitenkaan ole niin suuria, että muut vaihtoehdot olisivat parempia. Vaikka laajennettavuutta ei jää suoraan tälle ohjelmoitavalle logiikalle I/O-moduulien mielessä paljoa jäljelle, voidaan järjestelmä aina kytkeä osaksi suurempaa kokonaisuutta Profinetin avulla.

Koska S7-1200:n komponentit ovat leveämpiä verrattuna nykyisiin, joudutaan tulevaa laitteistoa jakamaan nykyisten I/O-kehikoiden välille. Tämä voidaan ratkaista laajennuskaapelilla, joka voidaan asentaa mihin tahansa konfiguraation väliin (Kuva 16).



KUVA 16. Ohjelmoitavan logiikan layout

Tässä tilanteessa paras sijainti on kuitenkin CPU:n ja ensimmäisen I/O-kortin välissä. Tällä tavalla ylemmälle DIN-kiskolle asennetaan virtalähde ja CPU, kun alemmalle asennetaan kaikki 6 I/O-moduulia (Kuva 17). Tällä tavalla johdotuksetkaan eivät pääse muuttumaan kovin paljoa, mikä nopeuttaa ja helpottaa asennuksia.



KUVA 17. Asennuskaapin layout

4 OHJELMISTON MODERNISOINTI

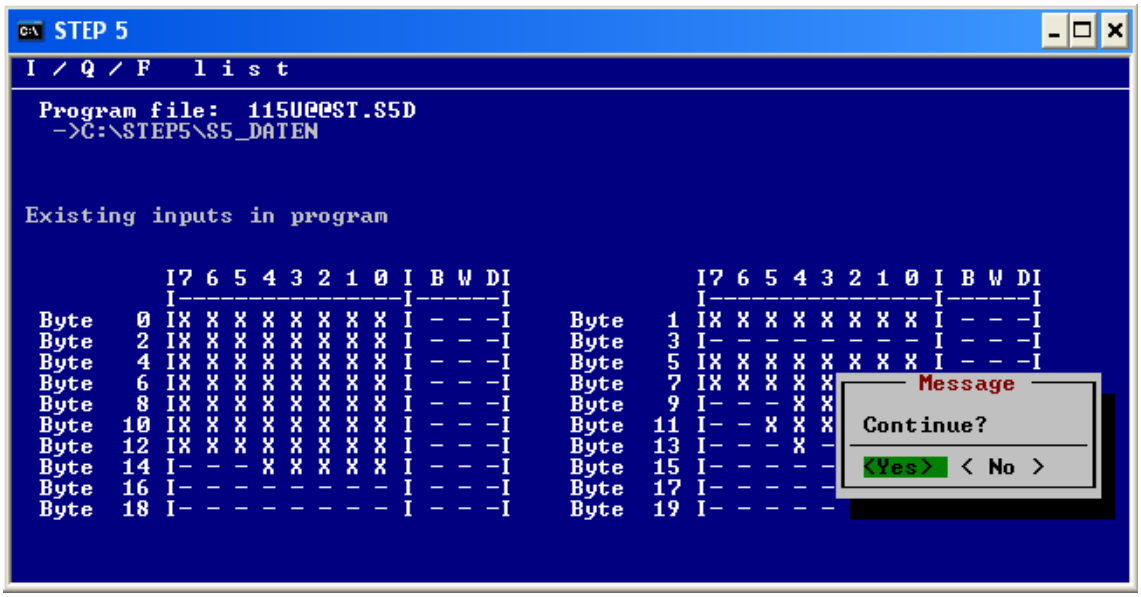
Modernisoitavan kohteen ohjelmisto on tehty vanhalla STEP 5 -ympäristöllä. Modernisoitavaan kohteeseen ei myöskään ole saatavilla alkuperäistä ohjelmointitiedostoa, jonka vuoksi on saatavilla vain se versio ohjelmasta, joka on ladattavissa prosessorilta. Tässä versiossa ohjelmasta ei ole siis mukana kommentteja tai minkään muuttujan nimeä. Se luo ehdottomasti pieniä haasteita modernisoinnille, koska on hyvin hankala tietää, mitä mikäkin osa ohjelmasta tekee ilman kommentointia. Toisaalta työssä ei ole tilattu toiminnallisia muutoksia, joten suurilta osin ohjelmaan ei ole syytä edes koskea.

4.1 I/O-määrä

STEP 5 pystyy generoimaan helposti I/O-luettelon, jossa nähdään kaikki käytössä olevat I/O:t. On tärkeää tietää, kuinka monta I/O:ta on todellisuudessa käytössä ohjelmassa. Modernisoinnissa joudutaan vaihtamaan laitteisto täysin, minkä vuoksi pitää olla ymmärrys tarvittavista I/O-moduuleista.

Olisi myös hyvä tarkastella valmiiksi mahdollista I/O-jaottelua. Yleensä modulaariset kortit ovat joko 8, 16 tai 32 I/O:ta sisältäviä. Yhteen korttiin pystyisi liittämään tuon verran tuloja tai lähtöjä.

Siemensin STEP 5 kykenee itse luomaan listan käytössä olevista I/O:ista (Kuva 18). Tulojen ja lähtöjen osoite muodostuu tavujen biteistä. Osoitteet muodostuvat tavujen biteistä, esimerkiksi tavu 1, jonka bittiä 0 tarkasteltaisiin, merkittäisiin muodossa I / Q 1.0. Vastaavalla tavalla voidaan muodostaa viidennen bitin osoite tavussa 3, joka merkittäisiin muotoon I / Q 3.5.



KUVA 18. STEP 5 I / Q / F listaus, jossa on näkyvillä sovelluksesta löytyvät tulot

```
Existing outputs in program

          I7 6 5 4 3 2 1 0 I B W DI            I7 6 5 4 3 2 1 0 I B W DI
          I----------------I------I            I----------------I------I
  Byte   0 I- - - - - - - - I - - -I    Byte   1 I- - - - - - - - I - - -I
  Byte   2 I- - - - - - - - I - - -I    Byte   3 I- - - - - - - - I - - -I
  Byte   4 I- - - - - - - - I - - -I    Byte   5 I- - - - - - - - I - - -I
  Byte   6 I- - - - - - - - I - - -I    Byte   7 I- - - - - - - - I - - -I
  Byte   8 I- - - - - - - - I - - -I    Byte   9 I- - - - - - - - I - - -I
  Byte  10 I- - - - - - - - I - - -I    Byte  11 I- - - - - - - - I - - -I
  Byte  12 I- - - - - - - - I - - -I    Byte  13 I- - - - - - - - I - - -I
  Byte  14 I- - - - - - - - I - - -I    Byte  15 I- - - - - - - - I - - -I
  Byte  16 IX X X X X X X X I - - -I    Byte  17 IX X X X X X X X I - - -I
  Byte  18 IX X X X X X X X I - - -I    Byte  19 IX X X X X X X X I - - -I
  Byte  20 IX X X X X X X X I - - -I    Byte  21 IX X - X X X X X I - - -I
  Byte  22 IX X X X X X X X I - - -I    Byte  23 IX X X X X X X X I - - -I
  Byte  24 IX X X X X X X X I - - -I    Byte  25 I- X X X X X X X I - - -I
  Byte  26 IX X X X X X X X I - - -I    Byte  27 IX X X X X X X X I - - -I
  Byte  28 I- - - X X X X X I - - -I    Byte  29 IX X X X X X X X I - - -I
  Byte  30 IX X X X X X X X I - - -I    Byte  31 I- - - - - X X X I - - -I
  Byte  32 I- X X X X X X X I - - -I    Byte  33 I- - - - - - - - I - - -I
  Byte  34 I- - - - - - - - I - - -I    Byte  35 I- - - - - - - - I - - -I
  Byte  36 I- - - - - - - - I - - -I    Byte  37 I- - - - - - - - I - - -I
          I----------------I------I            I----------------I------I
```

Oikealla olevat B – W – D osoittavat, onko sitä vastaava rivi tavu (B) osoitteellinen, sana (W) osoitteellinen tai kaksoissana (D) osoitteellinen. B varaa 8 bittiä, W 16 bittiä ja D 32 bittiä. Tässä ohjelmassa kaikki ovat kuitenkin pelkkiä bittiosoitteellisia. Kohteessa ei ole yhtään analogista tulo- tai lähtökorttia, joihin näitä datatyyppejä saatettaisiin käyttää.

Kaikki digitaaliset tulot ja lähdöt edustavat binäärimuuttujia. Niiden arvo voi olla vain 1 tai 0. Jos tahdotaan käsitellä tietoa, joka ei ole vain binääristä, voidaan käyttää hyödyksi Int (Integer) datatyyppiä eli kokonaislukua. Integer koostuu 2 tavusta, jolloin se voi ilmaista tasalukua välillä -32668 – 32767. Jos käyttöön tarvittaisiin vieläkin suurempia kokonaislukuja, voitaisiin käyttää DInt (Double Int), jonka muistinvaraus on jo 4 tavua. Silloin DInt voi ilmaista tasalukua välillä -2,147,483,648 − 2,147,483,647. Arvoalue on negatiivisella puolella yhden suurempi kahden komplementin avulla, joka on binääriluvuille erilainen esitystapa. Kokonaislukujen arvoaluetta voidaan kasvattaa poistamalla siitä etumerkki. UInt (Unsigned Integer) voisi ilmaista lukemaa välillä 0 – 65535.

Jos näitä datatyyppejä halutaan käyttää, niille täytyy varata samalla tavalla muistiavaruudesta paikka kuin biteillekin. Int voisi olla esim. IW 64, joka voisi olla oikea kentältä tuleva toimilaite, kuten lämpötila-anturi. Koska Integerin muistinvaraus on 2 tavua, silloin ei voida laittaa toista datatyyppiä välille 64.0 – 65.7. On mahdollista kyllä laittaa esimerkin IW 64 rinnalle digitaalinen tulo I 65.2, mutta tämän 65.2 tulon arvo sekoittuisi Integerin sekaan, jolloin IW 64 lukema olisi väärä, koska yksi bitti ei olisi välttämättä oikein. Periaatteessa vähiten merkitseviä bittejä voitaisiin käyttää tarvittaessa, mutta tälle ei ole nykyaikana todellista syytä, sillä muistia on riittävästi suuriinkin kohteisiin.

4.1.1 I/O-luettelo

Ohjelmasta saadaan kuitenkin tieto siitä, mitkä osoitteet ovat ohjelmallisesti käytössä. Tällä voidaan määrittää ainakin tarkka määrä, kuinka monta I/O:ta tarvitaan uudelta laitteistolta. Tällä hetkellä kohteessa onkin todella paljon ylimääräisiä I/O:ita. Korteilla olikin tilaa 128 tulolle ja 160 lähdölle. Ohjelmallisesti käytössä on kuitenkin vain 100 tuloa ja 125 lähtöä. Lähtöjen määrä pienenee 48 kappaleeseen, kun LED-valotaulu korvataan HMI-paneelilla.

I/O-luettelon avulla voidaan tarkastella tulevaa jaottelua, eli kuinka nykyisten 32-kanavaisten korttien I/O:t saadaan jaoteltua 16-kanavaisiin kortteihin. 32-kanavaiset kortit on kuitenkin jaettu tavallaan 4 eri osaan, sillä yhteen tavuun mahtuu

vain 8 osoitetta. Modernisoitavassa kohteessa nämä osat on nimetty tulojen puolelta E0 – E15 ja lähtöjen puolelta A16 – A33 (Kuva 19). Asennustöitä helpottaisi, jos aina yksi tällainen osa pidettäisiin järjestyksessä, eikä siihen puututtaisi. Tämä voi kuitenkin hankaloitua, jos useita varalähtöjä tai -tuloja on jätetty myös näiden osien sisään.

KUVA 19. Jokainen kortti on jaettu tai nimetty 4 osaan

Kun I/O-luettelo on saatu rakennettua, siihen on voitu lisätä kommentoinnit aina tulon tai lähdön osalta kytkentäpiirustuksista. Tämä auttaa hieman selventämään sitä, että missä on mitäkin laitteita ja missä järjestyksessä (Kuva 20).

Inputs				
	Kortti	Address	Ei käytössä	Kommentti
E0		I 0.0		Anlage Ein/Aus
		I 0.1		Rollenbahn Ein/Aus
		I 0.2		formkasten transportgerät Ein/Aus
		I 0.3		Störung Rollenbahn Antrieb Pos.1
		I 0.4		Störung Rollenbahn Antrieb Pos.2
		I 0.5		Störung Rollenbahn Antrieb Pos.3
		I 0.6		Störung Rollenbahn Antrieb Pos.4
		I 0.7		Störung Rollenbahn Antrieb Pos.5
E1		I 1.0		Störung Rollenbahn Antrieb Pos.6
		I 1.1		Störung Rollenbahn Antrieb Pos.7
		I 1.2		Störung Rollenbahn Antrieb Pos.8
		I 1.3		Störung Rollenbahn Antrieb Pos.9
		I 1.4		Störung Rollenbahn Antrieb Pos.10
	Siemens	I 1.5		Störung Rollenbahn Antrieb Pos.11
	32 CH	I 1.6		Störung Verschiebewagen 1 Kleine Drehzahl
		I 1.7		Störung Verschiebewagen 1 Große Drehzahl
E2	Input	I 2.0		Störung Verschiebewagen 2 Kleine Drehzahl
		I 2.1		Störung Verschiebewagen 2 Große Drehzahl
		I 2.2		Störung Frequenzumformer
		I 2.3		Störung Vibramotore Pos.1
		I 2.4		Störung Hubtisch Pos.9
		I 2.5		Störung Hydraulikagg formkasten transportgerät
		I 2.6		Störung Fahrmotor formkasten transportgerät Kleine Drehzahl
		I 2.7		Störung Fahrmotor formkasten transportgerät Große Drehzahl
E3		I 3.0	x	
		I 3.1	x	
		I 3.2	x	
		I 3.3	x	
		I 3.4	x	
		I 3.5	x	
		I 3.6	x	
		I 3.7	x	
E4		I 4.0		Endschalter Rollenbahn Antrieb Pos.1
		I 4.1		Endschalter Rollenbahn Antrieb Pos.2

KUVA 20. I/O-luetteloa

Kytkentäpiirustuksista pystyy tarkastamaan, mitkä I/O:t ovat käytössä LED-valotaululla. Nämä voidaan poistaa I/O-luettelosta, koska niitä ei kytketä enää uudelle konfiguraatiolle. Jaottelussa on myös hyvä ottaa huomioon, ettei uusien kytkentöjen sijainti vaihtuisi paljoa. Tämä mahdollistaisi sen, ettei tarvitsisi vetää uusia kaapeleita tai jatkaa vanhoja.

4.1.2 I/O-jaottelu

Haasteita saattaa kuitenkin syntyä uusien I/O-moduuleiden muodosta. Uusissa korteissa liitännät ovat kortin ala- ja yläpuolella. Hybridkorteissa tulot ovat aina yläpuolella ja lähdöt alapuolella. Nykyisessä konfiguraatiossa tulot ovat ensimmäiset 4 korttia ja loput 5 ovat lähtöjä. CPU:lle tulee myös hieman kytkentöjä, mikä tulee sijaitsemaan ylemmällä DIN-kiskolla, jossa tällä hetkellä on toinen I/O-kehikko. Tämä mahdollistaisi sen, ettei ainakaan ylhäälle meneviä kytkentöjä tarvitsisi tuoda alas. Tähän ei kuitenkaan täysin päästä, vaan alhaalta täytyy tuoda 14 tuloa CPU:lle. Uusia kytkentöjä on kuitenkin tulossa yli 150, joten jos vain 14 niistä joudutaan vetämään uudella kaapelilla, se ei ole vielä kovin huono tilanne (Kuva 21).

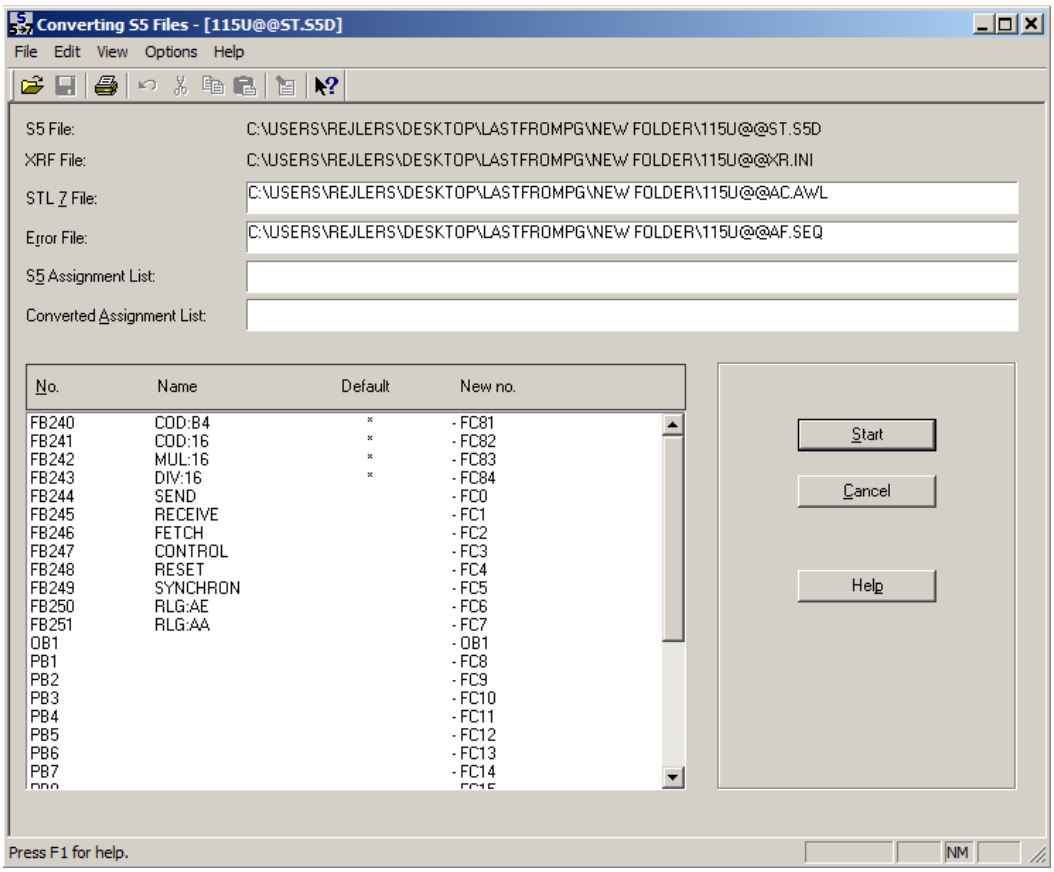
KUVA 21. Tuleva I/O-jaottelu uudelle konfiguraatiolle

4.2 Ohjelman konvertointi

Ohjelman konvertointia varten on useita työkaluja. Tällä hetkellä olemassa oleva ohjelma on STEP 5 -ohjelmointiympäristöön sopiva. Tavoitteena olisi saada tämä ohjelma toiminnallisuudeltaan samanlaisena konvertoitua TIA Portal V15.1 -alustalle. Tätä konvertointia ei pysty tekemään suoraan, vaan ensiksi ohjelma täytyy konvertoida STEP 7 Classic -ympäristölle. Tätä varten Siemensillä on tarjota sovellus ”Converting S5 Files”, jossa tämä konvertointi onnistuu hyvin pitkälti automatisoituna.

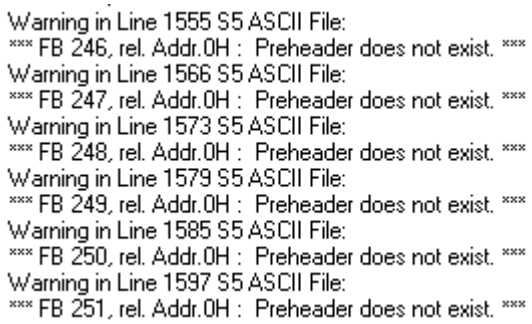
4.2.1 Ensimmäinen vaihe

Sovellukseen on syötettävä polut tiedostoista, joista halutaan luoda STEP 7 -lähdetiedosto (Kuva 22). Tällä saadaan S5-sovellus siirrettyä S7 Classicille ymmärrettävään muotoon, jossa se voidaan suoraan kääntää. Kääntäminen luo lähdetiedoston pohjalta kaikki STEP 5 -projektissa esiintyvät lohkot.



KUVA 22. Valmis työkalu projektin konvertointia varten

Konvertoinnissa on monta eri askelta, joissa voisi tulla ongelmia. Ohjelma kuitenkin osaa konvertoinnin päätteeksi ilmoittaa, mitä se ei osannut kääntää (Kuva 23).



KUVA 23. Konvertoinnin virheet ja varoitukset

Konverointityökalu varoittaa, että ohjelmassa olevista toimilohkoista puuttuu preheader, jossa säilytetään toimilohkon kommentointi ja hyppykomennoille tärkeät hyppykohdat. Ilmoitus johtuu siitä, että preheaderiä ei säilytetä CPU:n muistissa. Ohjelma jouduttiin alun perin lataamaan suoraan CPU:lta, joten ei näitä tietysti ole olemassa. Toimilohkot, joista varoitus on annettu, eivät myöskään ole käytössä nykyisessä ohjelmassa. Nämä ovatkin STEP 5:en valmiita toimilohkoja erilaisista kirjastoista, jotka jostain syystä ovat mukana ohjelmassa, vaikkei niitä ole käytetty mihinkään.

Konvertoinnin tuloksena saadaan yksi lähdetiedosto, joka sisältää koko ohjelmiston SCL-kielellä. Tämä tiedosto voidaan nyt tuoda uuteen projektiin STEP 7 Classicin puolella. Tätä ennen projektille on luotava hardware, jolle on valittava S7-300, sillä S7 Classic -ohjelmointiympäristö ei tue S7-1200-alustaa.

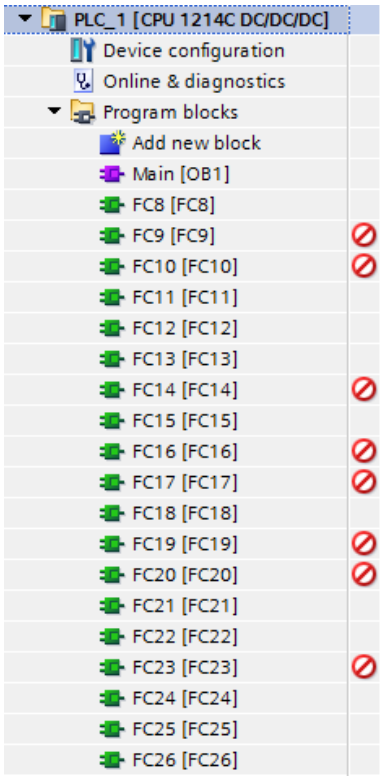
Kun lähdetiedosto tuodaan projektille, jossa on hardware, voidaan se suoraan kääntää. Tämä luo automaattisesti lähdetiedoston mukaiset lohkot, jotka ovat olleet käytössä ohjelmassa jo aiemmin. Koska kaikki lohkot ovat tässä vaiheessa SL-kielellä, ne voi valmiiksi kääntää FBD-kielelle, sillä S7-1200-alusta ei tue SL-kieltä. Ohjelmointiympäristö osaa kääntää LAD – FBD – STL kielten välillä lohkot automaattisesti, mutta ei kuitenkaan täydellisesti. Tämä projekti voidaan seuraavaksi siirtää TIA Portalin puolelle, jolloin samalla nähdään, kääntyivätkö kaikki

lohkot automaattisesti, vai tarvitseeko niitä itse kääntää käsin S7-1200 yhteensopiviksi.

STEP 7-projekteja voi siirtää TIA Portalille todella yksinkertaisesti, mutta näissä siirtyy projektissa käytettävä laitteisto mukana. Tämän vuoksi TIA Portalin puolelle joudutaan tekemään kaksi projektia: yksi projekti, joka voidaan suoraan siirtää tästä aiemmasta STEP 7-projektista, ja toinen täysin uusi projekti, johon täytyy määritellä projektin asetukset itse. Tähän määritellään samalla hardwareksi S7-1200, jolloin tähän projektiin voidaan rakentaa loppusovellus.

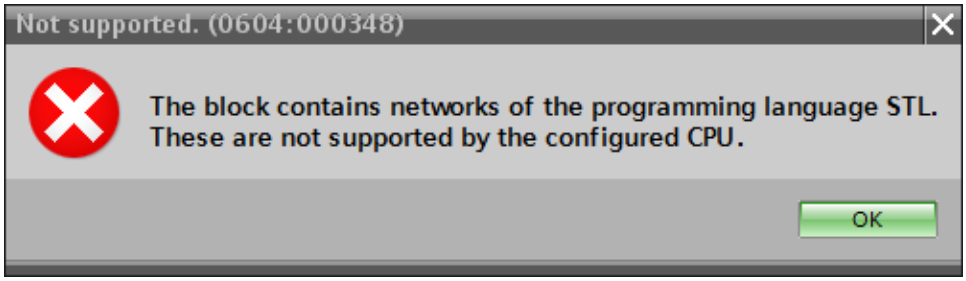
4.2.2 Toinen vaihe

Kun ohjelma on saatu siirrettyä TIA Portalin puolelle, voidaan se siirtää TIA Portalin projektiin, jossa on hardwarena S7-1200. Lohkoja voidaan siirtää helposti TIA Portalin sisäisesti eri projektien välillä. Tämän jälkeen lohkoista näkee suoraan mitkä toimivat ja mitkä eivät (Kuva 24).



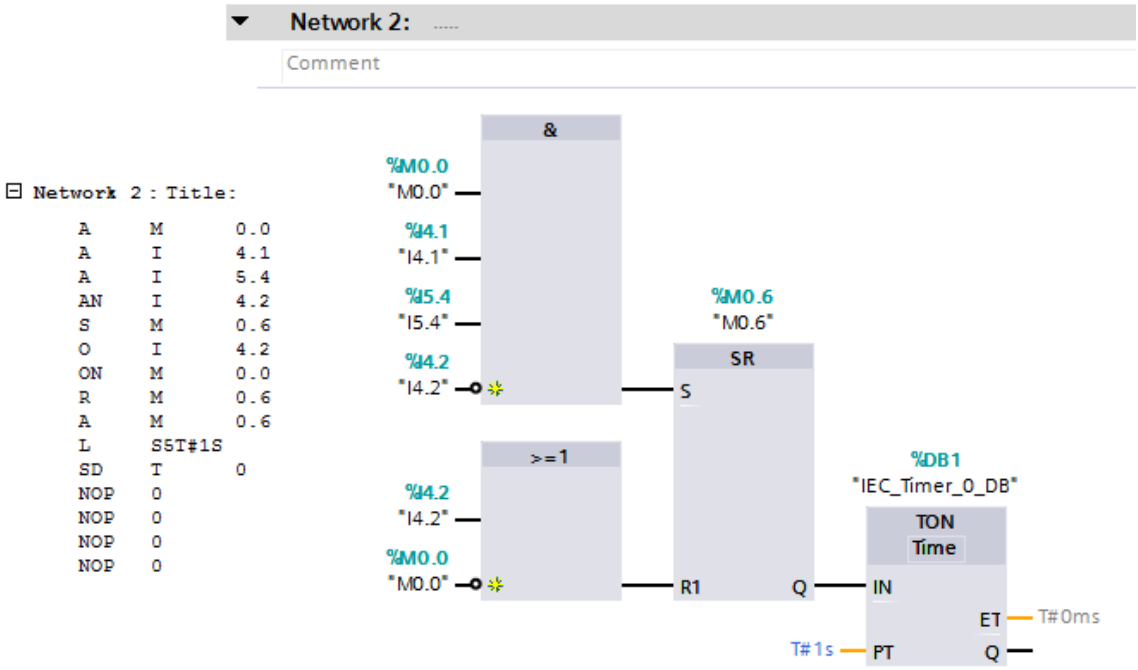
KUVA 24. Lohkot, joita ei voi voi tuoda S7-1200-alustalle

Nämä lohkot ovat lohkoja, joissa jokin osio (Network / Segment) ei kääntynyt S7 Classicin puolella suoraan FBD-kielelle (Kuva 25).



KUVA 25. Osa lohkoista ei ole yhteensopivia S7-1200:sen kanssa

Tähän voi olla useita syitä, eikä se ole mitenkään erikoista. Tässä vaiheessa TIA Portal osaa myös luoda automaattisesti Tag-listauksen käytössä olevista I/O:sta. Tag-listalta huomaa, että käytössä on 12 muuttujaa, joiden datatyyppi on Timer. Timer on datatyyppi, jota käytetään esimerkiksi ajastimien yhteydessä. S7-300 tukee tätä datatyyppiä, mutta S7-1200 ei. Tämä tarkoittaa sitä, että vanhat käytössä olevat s5time-ajastimet joudutaan myös käsin vaihtamaan IEC-ajastimiksi, jotka ovat tuettuina S7-1200-alustalla (Kuva 26).



KUVA 26. FC-lohko 9 ennen muutoksia ja muutoksien jälkeen, missä Network 2 käännettynä FBD-kielelle ja ajastin korjattuna

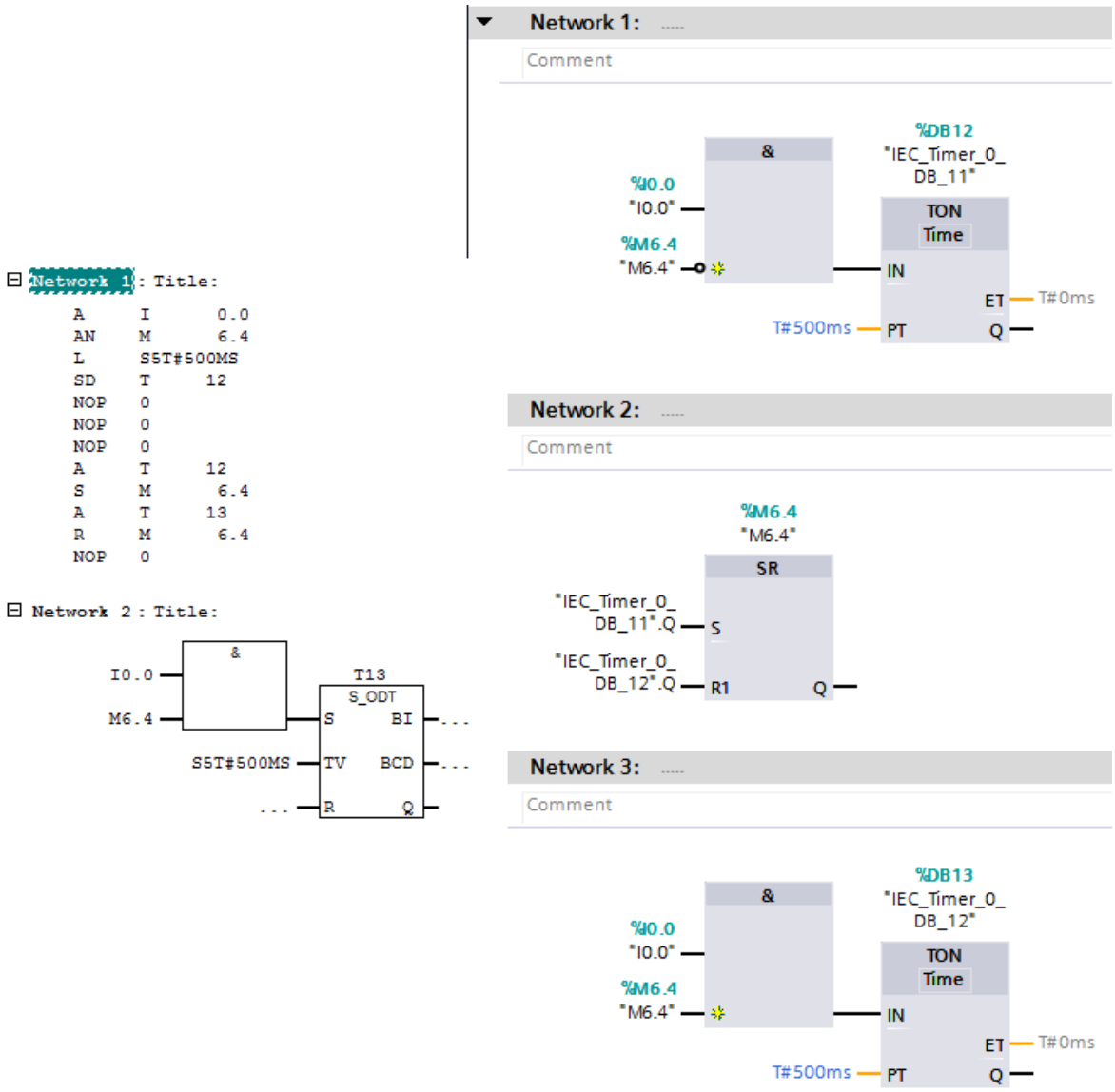
Kuvan 26 mukaisesti ohjelma saadaan käännettyä TIA Portaalille FBD-kielelle. Tämä network ei kääntynyt automaattisesti S7 Classicin puolella käytössä olleiden NOP 0 -käskyjen takia. Nämä ovat ”No Operation” käskyjä, jotka eivät tee mitään. Prosessori käsittelee NOP 0 -käskyn normaalina käskynä, jolloin siihen kuitenkin kuluu yksi kellojakso ja 2 tavua muistia. NOP 0 -käskyillä osoitetaankin toiminnallisuuden loppua tai esim. odottelua. Tätä vastaavaa lohkoa ei ole FBD-kielellä, jonka vuoksi sen kääntäminen automaattisesti ei onnistunut.

Samalla vanha s5time-ajastin (SD) on vaihdettu IEC-ajastimeksi TON. Ajastimien toiminta on käytännössä sama, mutta uudemmassa IEC-ajastimessa sen muuttujat ovat sijoitettuna datalohkossa. Esimerkiksi jos tästä ajastimesta tahdottaisiin tietää kulunut aika, sitä voitaisiin kutsua nimellä ”IEC_Timer_0_DB”.ET.

IEC-ajastimissa on myös toinen eroavaisuus, joka vaikuttaa modernisointiin. Jokaisen networkin pitää aina päättyä IEC-ajastimeen, eli ajastin ei voi olla esimerkiksi keskellä toimintoja. Tämä on myös syy, miksi networkit eivät kääntyneet automaattisesti. Näissä tilanteissa vanha kohta ohjelmasta voidaan kuitenkin yksinkertaisesti vain halkaista kahtia ajastimen kohdalta, jolloin saadaan 2 uutta networkia, joista toinen päättyy ajastimeen. Tätä samaa rajoitetta ei esiinny S7-1200- ja S7-1500-malleilla, mutta ohjelmaa muokataan vielä S7-300-hardwarella, jonka vuoksi ne täytyy ottaa huomioon.

Kuvan 27 tilanne oli juuri edellä kuvattu: SD-ajastinta oli käytetty ohjelman ”keskellä”, joten se on kuvassa oikealla puolella korjattu jakamalla se kahteen osaan. Alempi network 2 näyttää, kuinka pelkkä ajastin on vaihdettu, jotta se on saatu yhteensopivaksi uuteen järjestelmään.

Kun yhteensopimattomat lohkot on korjattu, ohjelma on konvertoitu onnistuneesti. Konvertointi onkin todella nopea prosessi, etenkin jos automaattiset työkalut toimivat suurilta osin, eikä vanhaa ohjelmaa tarvitse käsin muuttaa uudelle alustalle yhteensopivaksi.



KUVA 27. FC-lohko 16 ennen muutoksia ja muutoksien jälkeen, missä Network 6 on käännetty SL-kieleltä FBD-kielelle

Nämä konvertoidut lohkot voidaan nyt tuoda TIA Portalista, jossa käytössä on S7-300-hardware suoraan TIA Portaliin, johon hardwareksi on määritelty S7-1200. Ohjelmiston osalta modernisointi on tässä vaiheessa valmis.

4.3 HMI

Vanha LED-valotaulu halutaan korvata uudella HMI-paneelilla, joka laskee I/O-määrää merkittävästi. Tällä hetkellä jokainen LED-valo on ohjattu aina lähtökorttien lähdöistä. Esimerkiksi rajatunnistimen tilaa kuvaavan valon lähdön osoite

voisi olla %Q23.4. Kortti päästää tiettyjen ehtojen mukaisesti jännitteen LED-valolle, jolloin LED-valo syttyy. LED-valoja on yhteensä 77. Valojen tilaa kuvataan bitillä eli arvoilla päällä tai pois.

Näiden siirtäminen täysin HMI-paneelille on ohjelmallisesti todella yksinkertaista, eikä se vaadi edes muutoksia ohjelmaan. Jokainen osoite voidaan siirtää fyysiseltä lähtökortilta bittimuistiin eli M-muistialueelle. Tämä tapahtuu osoitetta muuttamalla esimerkiksi %Q23.4 -> %M23.4 (Kuva 28). Tällöin valot ovat käytettävissä samalla tavalla, mutta niitä ei ole kytketty I/O-moduuleille, vaan ne ovatkin ainoastaan ohjelman sisäisiä muuttujia.

197		Q20.5	Bool	%Q20.5	☐	☑	☑	☑	Flashing light molding box transport device
198		Q20.6	Bool	%Q20.6	☐	☑	☑	☑	Direction indicator flashing light 1
199		Q20.7	Bool	%Q20.7	☐	☑	☑	☑	Direction indicator flashing light 2
200		Q21.0_HMI	Bool	%M21.0	☐	☑	☑	☑	Sreuer voltage system on / off
201		Q21.1_HMI	Bool	%M21.1	☐	☑	☑	☑	Roller conveyor on
202		Q21.2_HMI	Bool	%M21.2	☐	☑	☑	☑	molded box transport device A
203		Q21.3_HMI	Bool	%M21.3	☐	☑	☑	☑	Roller conveyor hand
204		Q21.4_HMI	Bool	%M21.4	☐	☑	☑	☑	Automatic roller conveyor
205		Q21.6	Bool	%Q21.6	☐	☑	☑	☑	Ei sähkökuvissa
206		Q21.7	Bool	%Q21.7	☐	☑	☑	☑	Control cabinet collective fault
207		Q22.0_HMI	Bool	%M22.0	☐	☑	☑	☑	Roller conveyor drive pos. 1
208		Q22.1_HMI	Bool	%M22.1	☐	☑	☑	☑	Roller conveyor drive pos. 2
209		Q22.2_HMI	Bool	%M22.2	☐	☑	☑	☑	Roller conveyor drive pos. 3 right (from p...
210		Q22.3_HMI	Bool	%M22.3	☐	☑	☑	☑	Roller conveyor drive pos. 3 left (after pos....
211		Q22.4_HMI	Bool	%M22.4	☐	☑	☑	☑	Roller conveyor drive pos. 4
212		Q22.5_HMI	Bool	%M22.5	☐	☑	☑	☑	Roller conveyor drive pos. 5
213		Q22.6_HMI	Bool	%M22.6	☐	☑	☑	☑	Roller conveyor drive pos. 6
214		Q22.7_HMI	Bool	%M22.7	☐	☑	☑	☑	Roller conveyor drive pos. 7
215		Q23.0_HMI	Bool	%M23.0	☐	☑	☑	☑	Roller conveyor drive pos. 8 right (after po...
216		Q23.1_HMI	Bool	%M23.1	☐	☑	☑	☑	Roller conveyor drive pos. 8 left (from pos....
217		Q23.2_HMI	Bool	%M23.2	☐	☑	☑	☑	Retract roller conveyor drive item 9 (from i...
218		Q23.3_HMI	Bool	%M23.3	☐	☑	☑	☑	Extend roller conveyor drive pos. 9 (accor...
219		Q23.4_HMI	Bool	%M23.4	☐	☑	☑	☑	Roller conveyor drive pos. 10
220		Q23.5_HMI	Bool	%M23.5	☐	☑	☑	☑	Roller conveyor drive pos. 11
221		Q23.6_HMI	Bool	%M23.6	☐	☑	☑	☑	Transfer carriage 1 Slowly forward (after it...
222		Q23.7_HMI	Bool	%M23.7	☐	☑	☑	☑	Transfer carriage 1 Slowly back (according...

KUVA 28. TIA Portal tag-listaus, jossa fyysisten LED-valojen osoitteet vaihdettu M-muistialueelle

TIA Portal jakaa hardwaren tavallaan omille alueilleen, jolloin PLC:llä ja HMI:llä on molemmilla esimerkiksi omat tag-listaukset. Tällä hetkellä kaikki on PLC:n omalla tag-listalla. HMI ei pysty suoraan viittaamaan PLC:llä oleviin tageihin tai paljon muuhunkaan. HMI:n omalla tag-listalla on kuitenkin mahdollisuus luoda viittaukset näihin muuttujiin PLC:n listalla, jolloin niitä voidaan myös käyttää HMI:n puolella (Kuva 29).

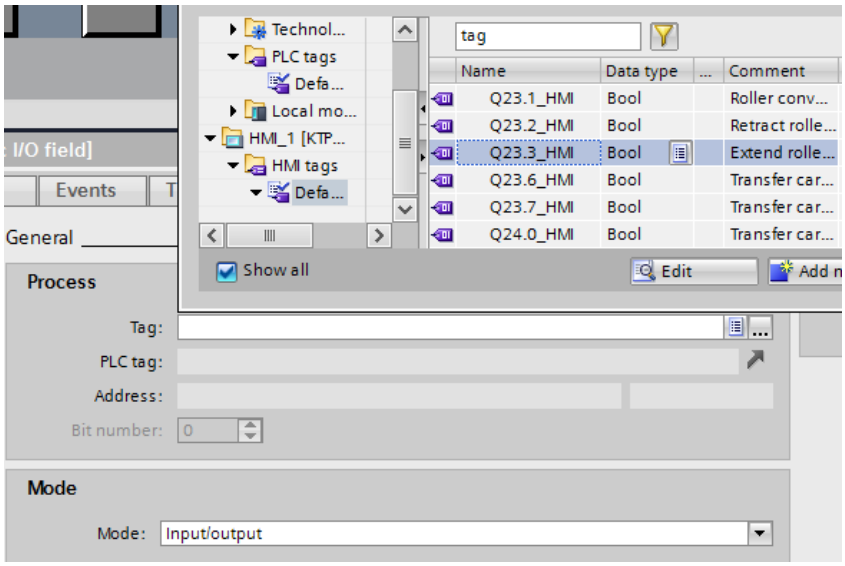
Q21.0_HMI	Bool	HMI_Connectio...	PLC_1	"Q21.0_HMI"	<symbolic access>	1 s	Sreuer voltage system on / off
Q21.1_HMI	Bool	HMI_Connectio...	PLC_1	"Q21.1_HMI"	<symbolic access>	1 s	Roller conveyor on
Q21.2_HMI	Bool	HMI_Connectio...	PLC_1	"Q21.2_HMI"	<symbolic access>	1 s	molded box transport device A
Q21.3_HMI	Bool	HMI_Connectio...	PLC_1	"Q21.3_HMI"	<symbolic access>	1 s	Roller conveyor hand
Q21.4_HMI	Bool	HMI_Connectio...	PLC_1	"Q21.4_HMI"	<symbolic access>	1 s	Automatic roller conveyor
Q22.0_HMI	Bool	HMI_Connectio...	PLC_1	"Q22.0_HMI"	<symbolic access>	1 s	Roller conveyor drive pos. 1
Q22.1_HMI	Bool	HMI_Connectio...	PLC_1	"Q22.1_HMI"	<symbolic access>	1 s	Roller conveyor drive pos. 2
Q22.2_HMI	Bool	HMI_Connectio...	PLC_1	"Q22.2_HMI"	<symbolic access>	1 s	Roller conveyor drive pos. 3 right (from pos. 2)
Q22.3_HMI	Bool	HMI_Connectio...	PLC_1	"Q22.3_HMI"	<symbolic access>	1 s	Roller conveyor drive pos. 3 left (after pos. 4)
Q22.4_HMI	Bool	HMI_Connectio...	PLC_1	"Q22.4_HMI"	<symbolic access>	1 s	Roller conveyor drive pos. 4

KUVA 29. HMI tag-listaus, joissa viittaukset PLC:n tag-listaukseen

4.3.1 Käyttöliittymä

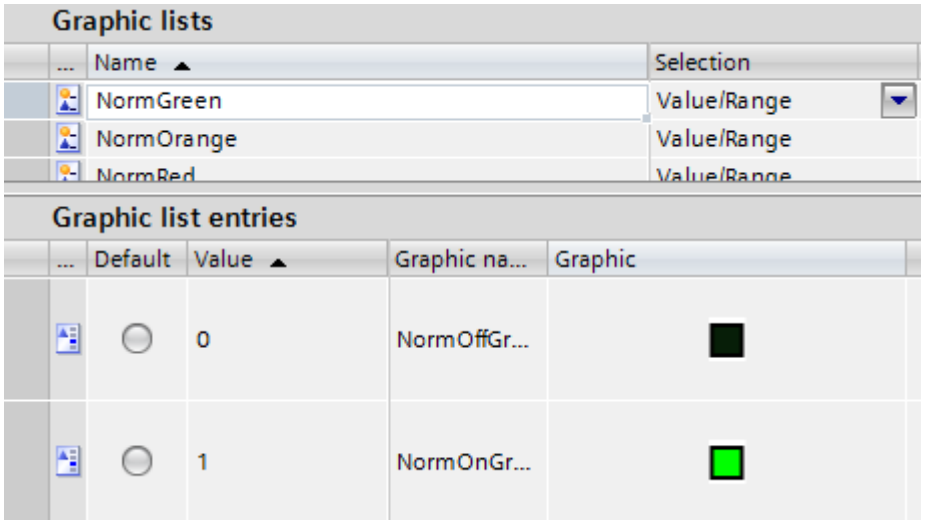
Projektissa ei ole tarkoitus lisätä ominaisuuksia tai tehdä olemassa olevaan järjestelmään toiminnallisia muutoksia. Tämän vuoksi ei ole tarvetta käyttää kovinkaan montaa ominaisuutta HMI:n rakentamiseen. Koko LED-valotaulu saadaan mahtumaan yhdelle HMI-sivulle, joka tarkoittaa sitä, ettei järjestelmään ole tarvetta rakentaa valikoita tai vastaavia järjestelmiä. Kun HMI käynnistyy, siinä näkyy valmiiksi yksi ja oikea sivu, eikä sitä voi vaihtaa.

Valojen tilaa voidaan kuvata Graphic I/O Field -elementeillä. Tämä on Siemensin valmiiksi tarjoama tapa toteuttaa esimerkiksi valon toiminta (Kuva 30). Näitä voidaan laittaa yhdelle sivulle useita, jolloin kaikki valot saadaan samalle sivulle.



KUVA 30. Graphic I/O Field, johon voidaan liittää HMI tag, PLC:llä olevan muuttujan viite

Kun Graphic I/O Fieldiin on liitetty muuttuja, se pystyy seuraamaan sen tilaa eli se tavallaan kopioi itseensä siihen linkitetyn muuttujan arvon. Graphic I/O Field voisi olla esimerkiksi neliö näytöllä, joka vaihtaa väriä aina liitetyn muuttujan tilan mukaan. Tähän elementtiin voidaan lisäksi liittää grafiikkalista, joka määrää mitä grafiikkaa sen kuuluisi näyttää (Kuva 31). Grafiikkalistalla voidaan siis määrätä mitä grafiikkaa Graphic I/O Field näyttää siihen liitetyn muuttujan arvon mukaisesti.



KUVA 31. Projektissa käytettävät grafiikkalistat

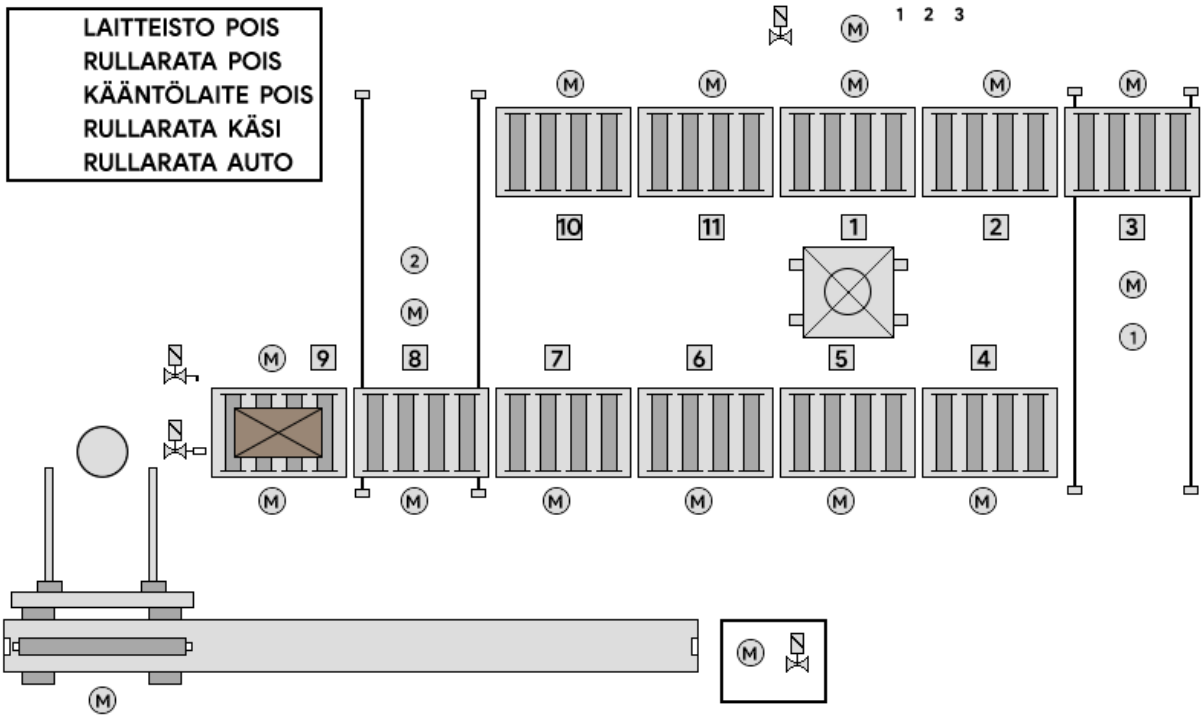
Kuvan 31 mukaisesti tumma neliö voi kuvata sammuksissa olevaa valoa, joka ei ole aktiivinen. Tälle asetetaan arvoksi 0, eli Graphic I/O Field näyttäisi tältä, jos siihen linkitetyn muuttujan arvo on 0. Vastaavasti vihreä neliö kuvaisi aktiivista valoa. Tälle arvoksi voidaan laittaa 1, eli se olisi aktiivinen, kun siihen linkitetyn muuttujan arvo olisi 1. Koska tässä käytetään bittejä, muita arvoja ei ole saatavilla.

Olisi myös mahdollista esim. Int (Integer) muuttujien avulla lisätä paljon enemmän grafiikkaa. Jokaisen grafiikan arvo voi olla myös väli, esimerkiksi 0 – 50 tai 50 – 100. Näin voitaisiin saada esim. lämpötilamittareille tehtyä kuvaavaa grafiikkaa.

Projektissa käytetään juuri näitä vihreitä valoja todella monta kertaa, joten grafiikkalistojen etu tulee helposti esille. Ne täytyy tehdä vain kerran, jolloin jatkossa

aina uusille Graphic I/O Field elementeille voidaan antaa parametriksi käytettävä grafiikkalista.

Projektille pitää myös rakentaa pohja, jonka päälle kaikki valot tulevat. Tämän voi suoraan kopioida kuitenkin vanhasta LED-valotaulusta, sillä grafiikka siinä on jo tarpeeksi havainnollistavaa. Kopiointi joudutaan kuitenkin tekemään käsin. Uusi kuva piirrettiin Adoben Photoshop-ohjelmistolla (Kuva 32). Kuvan voisi myös piirtää suoraan TIA Portalin omilla piirtotyökaluilla tai esimerkiksi Microsoft Windowsin mukana tulevalla Paintilla.



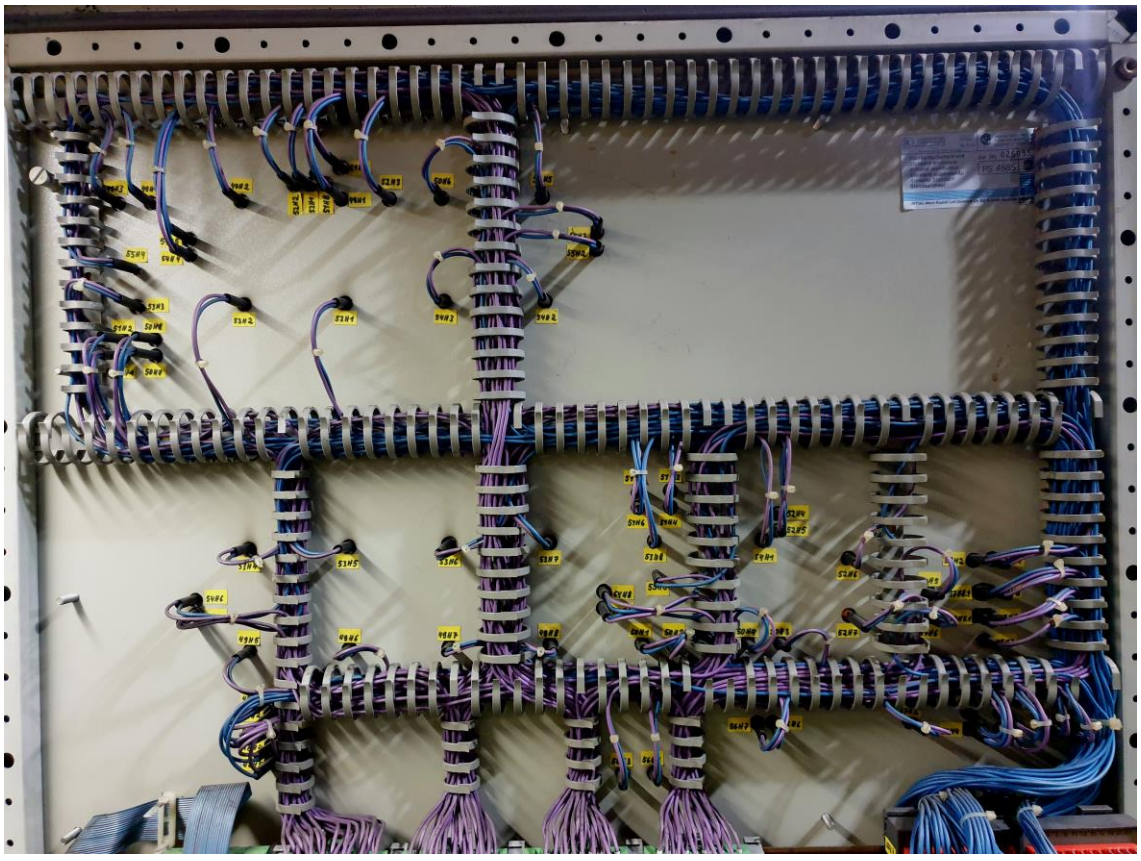
KUVA 32. LED-taulu digitaalisessa muodossa

Siemensin HMI-editori tukee myös layereitä eli kerroksia. Pohjakuva voidaan asettaa alimmalle kerrokselle, jolloin kaikki muut oliot kuvassa näkyvät sen päällä. Muun toiminnallisuuden halutaan olevan aina tämän päällä, joten esimerkiksi kaikki Graphic I/O Field -valot asetetaan ylemmille kerroksille.

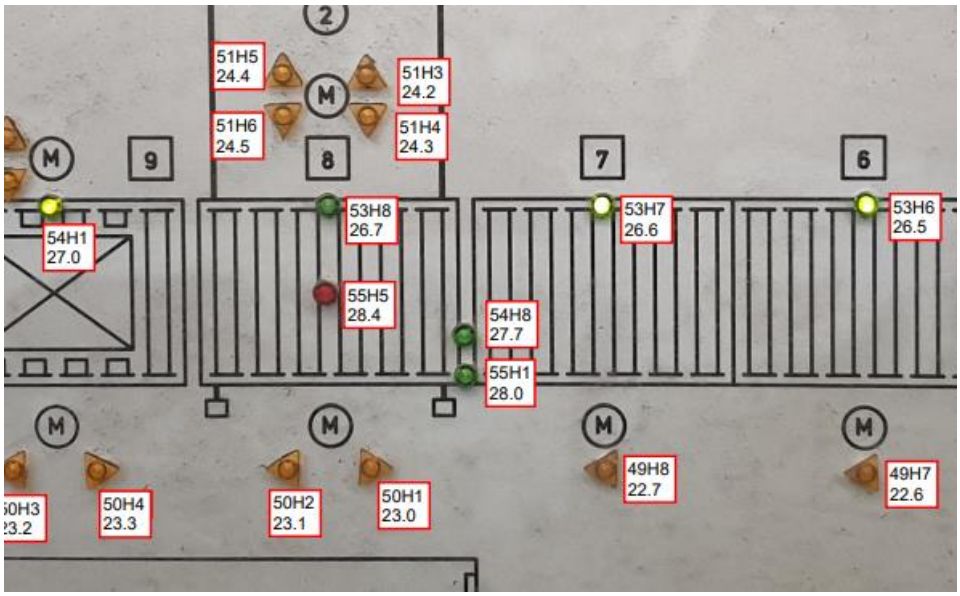
4.3.2 Osoitteisto

Graphic I/O Field -elementtejä tulee yhteensä 77 kappaletta, alkuperäisten LED-valojen tilalle. Haasteeksi kuitenkin tulee, kun nämä LED-valot pitää sijoittaa kuvaan alkuperäisiin kohtiin. Ei LED-valoista ole mitään apua, jos ne eivät ole oikeissa kohdissa kuvaa. Kytkentäpiirustuksia seuraamalla ja kommentteja lukemalla voi päätellä, mihin osoitteeseen liittyy mikäkin valo LED-taululta. Tällä ei kuitenkaan selviä kaikkien valojen osoitteet.

Nykyisen paneeliin taakse on kuitenkin merkitty jokaisen LED-valon vierelle merkintä (Kuva 33). Esimerkiksi merkintä 49H4 tarkoittaisi kytkentäpiirustuksista lehteä 49 ja valoa 4. Tämän avulla voidaan tarkistaa jokaisen valon todellinen osoite kytkentäpiirustuksia hyväksi käyttäen. Nämä voidaan sitten merkitä ylös, jolloin lopuksi HMI:n rakentaminen tulee olemaan nopeaa (Kuva 34).

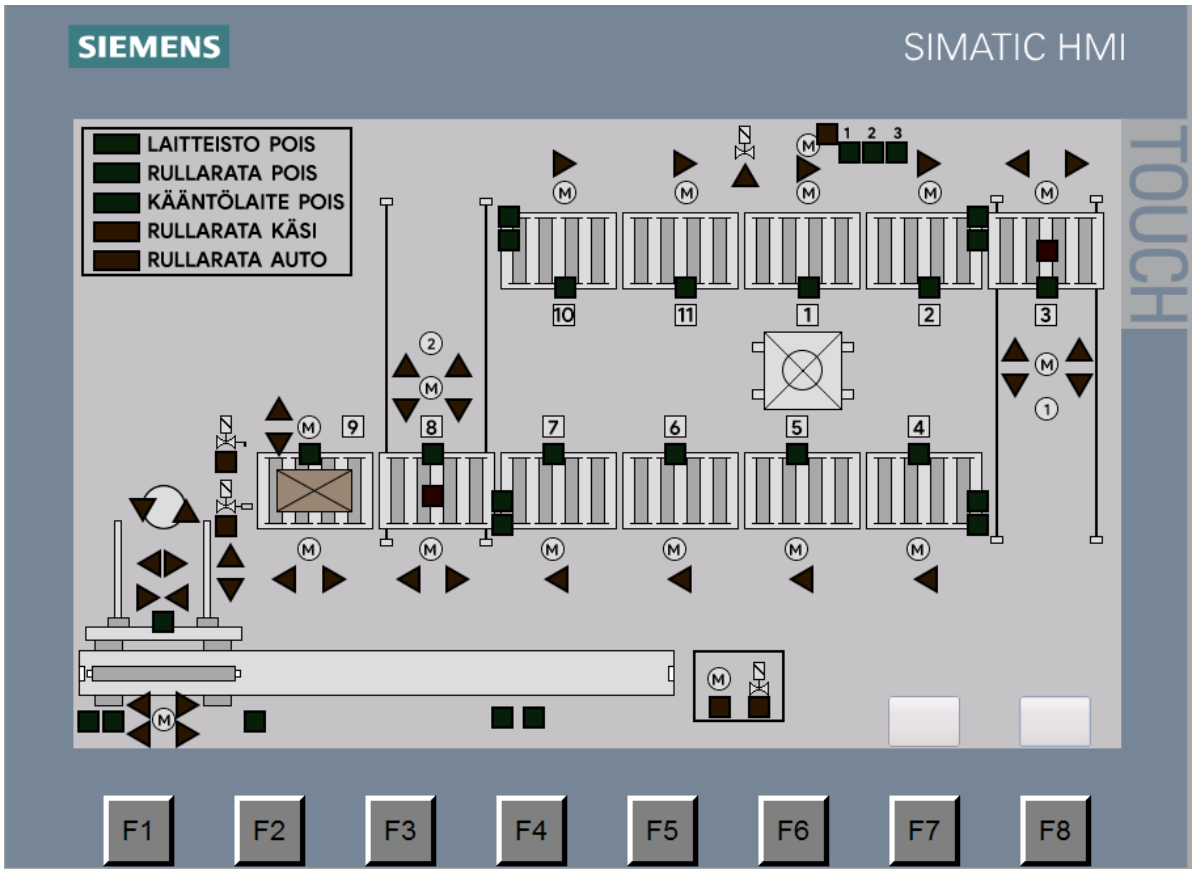


KUVA 33. Paneelin takana olevat merkinnät



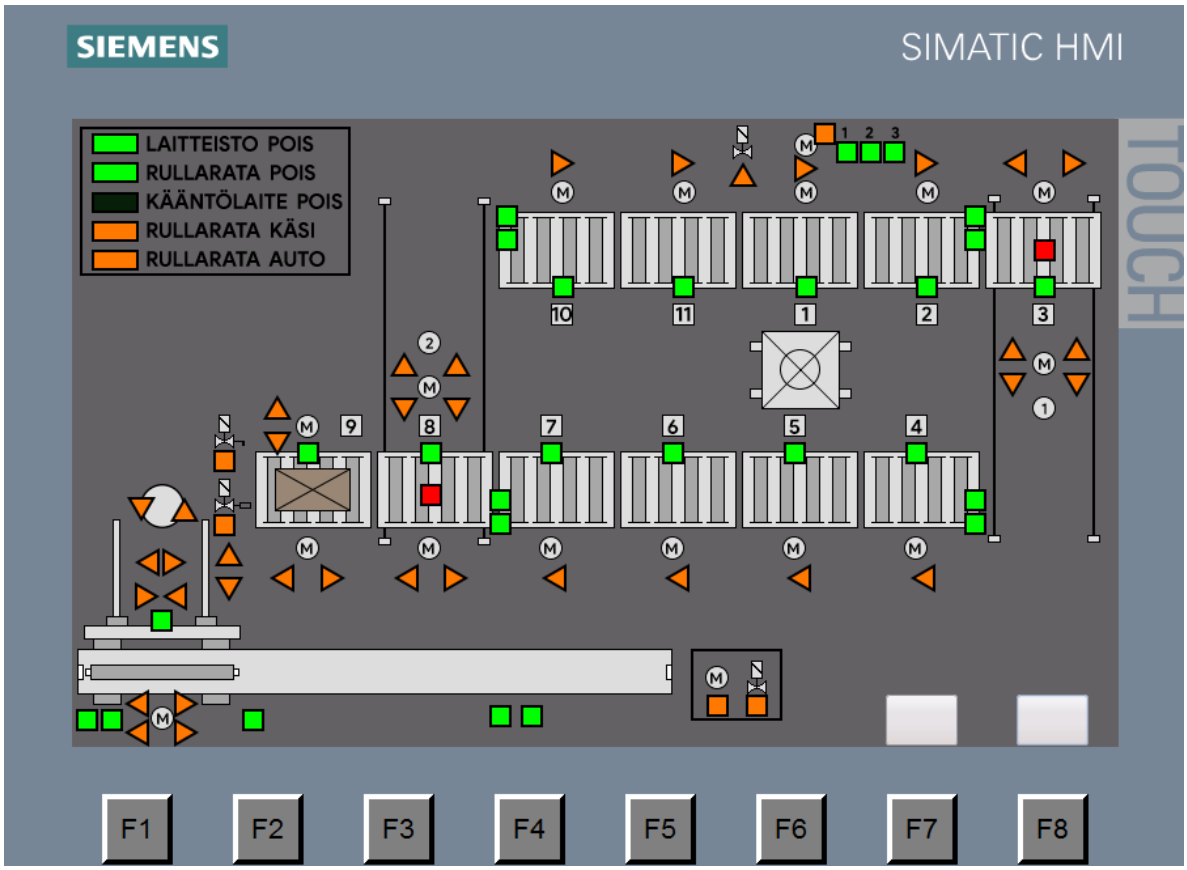
KUVA 34. Osoitteet merkittynä

Kun osoitteet on merkitty, voidaan nyt kaikki Graphic I/O Field -elementit linkittää oikeisiin osoitteisiin merkintöjä seuraten (Kuva 35).



KUVA 35. HMI valmiina simulaattorista katsottuna

Sivulle voidaan lisätä myös näppäimet, joilla taustaväriä voidaan vaihtaa tumman ja vaalean välillä (Kuva 36). HMI on valmis tältä osin. Järjestelmään jää valtavasti tilaa erilaisille ominaisuuksille. Laajennuksia voi tehdä tulevaisuudessa helposti.

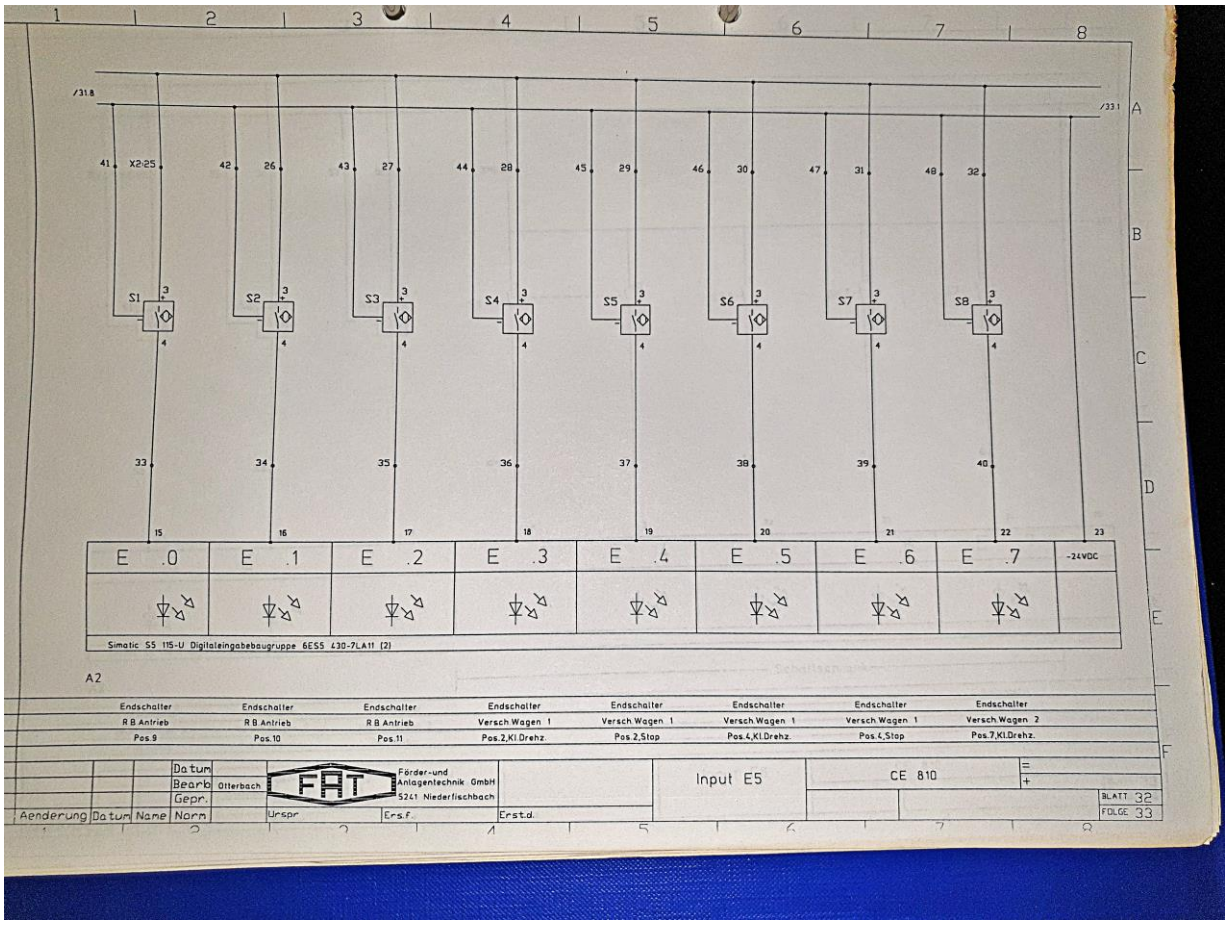


KUVA 36. Valojen tiloja simuloituna aktiiviseksi

5 PIIRUSTUSTEN PÄIVITTÄMINEN

Automaatiojärjestelmän modernisointi muuttaa kytkentöjä huomattavasti. Tämä johtaa siihen, että nykyisistä piirustuksista tulee vanhentuneita. Kohteen piirustukset ovat saatavilla vain paperisina versioina, eikä niitä ole saatavilla digitaalisina vektorikuvina. Piirustuksien siirtäminen digitaalisiksi olisi liian suuri kustannus, mutta sitä voi miettiä tarkemmin esim. käyttöönoton jälkeen. Käyttöönottoon riittää punakynät muutoksista, jolloin asennustyöt voidaan suorittaa käyttöönotossa niitä seuraten.

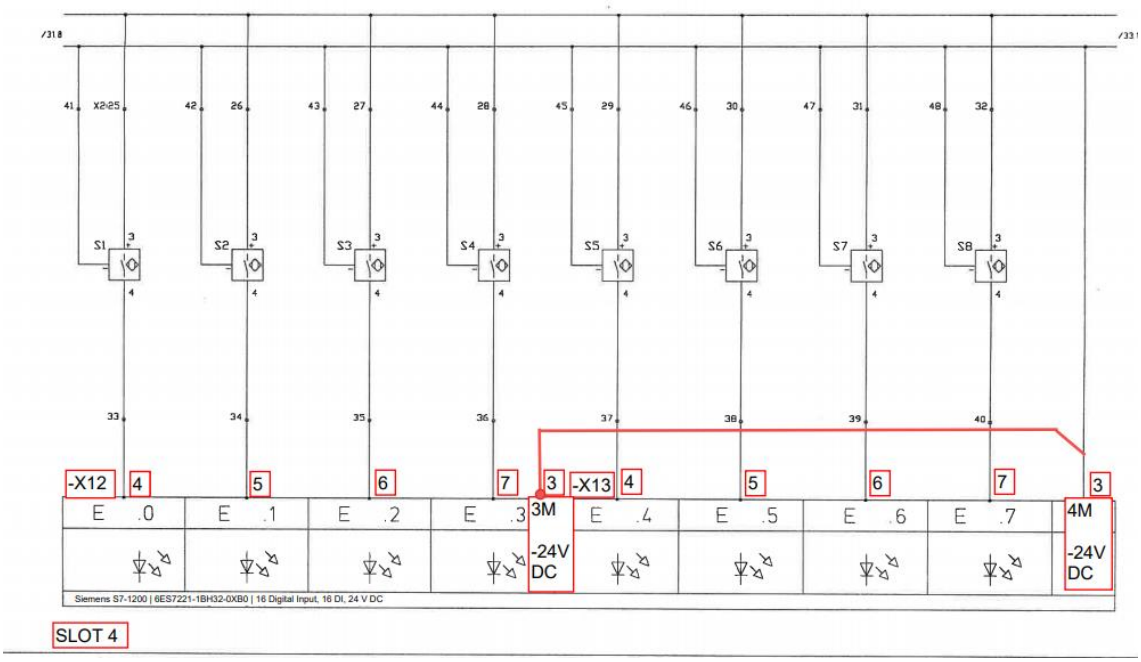
Kytkentäpiirustuksia modernisoitavasta kohteesta on noin 115 lehteä, joista arviolta kolmasosaan tulee muutoksia (Kuva 37). Alkuperäiset paperiset piirustukset voidaankin skannata digitaalisiksi rasterikuviksi, joihin muutokset voidaan merkitä niiden päälle.



KUVA 37. Alkuperäiset piirikaaviot

Alkuperäisissä piirustuksissa on käytetty tiettyjä tapoja toteuttaa ja hahmottaa automaatiojärjestelmän rakennetta. Tuleviin punakyniin tämä rakenne tulisi pitää ennallaan, jotta kuvia joutuisi muokkaamaan mahdollisimman vähän. Myös jos myöhemmin asiakas päättäisi haluta kuvista digitaaliset versiot, olisi piirtäminen suoraviivaisempaa.

Kytkennälliset muutokset onkin rajoitettu täysin automaatiojärjestelmän päähän, eikä muutoksia tarvitse tehdä esim. riviliittimillä tai kytkentäkoteloilla. Tämä helpottaa ja nopeuttaa asennustöitä. Punakynäversiotkin piirustuksista ovat riittäviä, sillä niissä näkyvät kytkennät pitävät paikkaansa (Kuva 38). Tämän vuoksi piirustuksia ei olisi välttämätöntä siirtää edes digitaalisiksi ja piirtää puhtaiksi.



KUVA 38. Punakynäykset kuvan 37 piirustuksen päällä

6 POHDINTA

Projekti itsessään onnistui hyvin ja pysyi aikataulussa. Asiakkaalle saatiin luotua paras mahdollinen kokonaisuus modernisoinnin suhteen. Valittu järjestelmä on edullisin kustannuksiltaan. Samalla uusi järjestelmä ottaa huomioon tulevaisuuden vaatimukset ja onkin näin päivitettävissä tarvittaessa. Uusi järjestelmä on vasta suunniteltu, eikä sitä ole vielä käyttöönotettu. Käyttöönotto ei pääse mukaan opinnäytetyöhön, koska sen ajankohta ei ole vielä varmistunut.

Ohjelmaan itsessään ei tullut yhtään toiminnallisia muutoksia, joten sen toimintakin pysyi täysin ennallaan. HMI-paneeli ja sen toiminnallisuus pysyi nyt todella yksinkertaisena, mutta siihen jää mahdollisuus päivittää sitä tulevaisuudessa. Järjestelmään voisi liittää tiedonkeruuta ja etäkäyttöönkin olisi mahdollisuus.

Projekti oli opinnäytetyöksi mielestäni todella hyvä, sillä se oli riittävän pieni, joten pystyin suunnittelemaan itse kaikki sen osa-alueet. Sisällöltään projekti ei myöskään ollut liian haastava. Samalla pääsin tutustumaan vanhempaan Siemensin STEP 5 -järjestelmään, josta ennestään minulla ei ollut kokemusta.

Haasteita syntyi siitä, kun alkuperäisestä ohjelmasta ei ollut tallella kommentteja tai muuttujien nimiä. Prosessista ei myöskään ollut selkeää toiminnallista kuvausta, eikä sen selvittämiseen varattu aikaa. Tämän vuoksi pitikin kiinnittää erityistä huomiota, ettei ohjelman toiminnallinen puoli muuttuisi modernisoinnin yhteydessä.

LÄHTEET

1. Electrical 4 U, Vidya Muthukrishnan. Programmable Logic Controllers (PLCs): Basics, Types & Applications. Saatavissa: https://www.electrical4u.com/programmable-logic-controllers/. Hakupäivä 07.07.2020.
2. International Electrotechnical Commission. IEC 61131-3:2013. Saatavissa: https://webstore.iec.ch/publication/4552. Hakupäivä 07.07.2020.
3. EC & M, R. Fehr. The Basics of Ladder Logic. Saatavissa: https://www.ecmweb.com/archive/article/20891380/the-basics-of-ladder-logic. Hakupäivä 09.07.2020.
4. PLC dev. Siemens S7 Status Word. Saatavissa http://www.plc-dev.com/siemens_s7_status_word. Hakupäivä 12.07.2020.
5. Siemens. Programming Style Guide for S7-1200/S7-1500. Saatavissa: https://cache.industry.siemens.com/dl/files/084/109478084/att_900353/v1/81318674_Programming_Style-guide_DOC_v12_en.pdf. Hakupäivä 14.07.2020.
6. Siemens. S5-115U Programmable Controller Manual. Saatavissa: https://cache.industry.siemens.com/dl/files/937/1085937/att_904/v1/6ES5_998-0UF23.pdf. Hakupäivä 17.07.2020.
7. Siemens. SIMATIC throughout history. Saatavissa: https://new.siemens.com/global/en/company/about/history/history-features/60-years-of-simatic.html. Hakupäivä 18.07.2020.
8. Siemens. Catalog ST 50 english, chpt. 1. Saatavissa: https://www.automation.siemens.com/salesmaterial-as/catalog/en/st5098_e.pdf%3FHTTPS%3DREDIR. Hakupäivä 20.07.2020.
9. Siemens. What is the meaning of the terms "sinking" (German: "P-lesend") and "sourcing" (German: "M-lesend") in digital modules of SIMATIC? Saatavissa:https://support.industry.siemens.com/cs/docment/23451499/what-is-the-meaning-of-the-terms-sinking-(german%3A-p-lesend-)-and-sourcing-(german%3A-m-lesend-)-in-digital-modules-of-simatic-?dti=0&lc=en-EC. Hakupäivä 22.07.2020.

10. Siemens. Interface Module Adapter. Saatavissa: https://assets.new.siemens.com/siemens/assets/api/uuid:f3b4c54a-8d0d-4ae2-b777-a97072847d78/version:1538367746/2-interface-module-adapter.pdf. Hakupäivä 23.07.2020.
11. Siemens. 6AV2123-2JB03-0AX0. Saatavissa: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6AV2123-2JB03-0AX0. Hakupäivä 23.07.2020.