jamk.fi

Tölkkikuljettimien ohjelmointi

Jukka-Pekka Muotio

Opinnäytetyö
Toukokuu 2015

Automaatiotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Muotio, Jukka-Pekka	Opinnäytetyö	24.5.2015
	Sivumäärä 47	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: (x)

Työn nimi
Tölkkikuljettimien ohjelmointi

Koulutusohjelma
Automaatiotekniikan koulutusohjelma

Työn ohjaaja(t)
Häkkinen, Veli-Matti

Toimeksiantaja(t)
Valio Oy

Tiivistelmä

Opinnäytetyön tilaajana oli Valio Oy Jyväskylä. Valiolla on lukuisia toimipisteitä ympäri maata. Henkilökuntaa on noin 4600. Yhtiö harjoittaa maidonjalostus toimintaa. Opinnäytetyön tuloksena saatu sovellus tuli käyttöön Valio Oy:n Jyväskylän meijeriin.

Tavoitteena oli uudistaa ja yksinkertaistaa tölkkikuljettimien ohjauksia kuljettimien uusinnan yhteydessä. Kuljettimista oli tarkoitus vähentää käytettävien valokennojen määrää, poistaa tarve jononpaine jarruille sekä parantaa häiriötilanteissa toimimista. Kaikki anturit sekä moottorien ohjaukset tulivat ASi väylään. Moottoreina käytettiin energiatehokkaita SEW Eurodriven DRC moottoreita joissa on nopeuden säätö integroituna valmiiseen ratkaisuun sekä liityntä suoraan ASi väylään jolloin käytössä on seis, käy sekä kaksi eri käyntinopeutta. Tarvittava määrä valokennoja pystyttiin liittämään suoraan moottorien kautta ohjelmoitavaan logiikkaan.

Opinnäytetyössä selvitetään ohjelmointiin liittyviä standardeja. Mitä automaatio on ja millainen automaation elinkaari on. Näiden ymmärtäminen antaa pohjan ymmärtää laadun tärkeys loppukäyttäjälle.

Sovelluksen suunnittelua käydään läpi, sekä mitä ratkaisuita sovelluksessa on tehty ja miten ne on toteutettu.

Tuloksena opinnäytetyöstä saatiin valmis sovellus, joka on käytössä tölkkikuljettimien ohjauksessa sekä tulevaisuudessa perusta ohjauksille kun tölkkikuljettimien uusinta jatkuu.

Avainsanat (asiasanat)
ISA-88, PackML, Modulaarinen, Ohjelmointi, Ohjelmoitava logiikka, Automaatio

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Muotio, Jukka-Pekka	Bachelor's thesis	24.5.2015
	Number of pages 47	Language of publication Finnish
		Permission for web publication: (x)

Title of publication
Programming carton conveyors

Degree programme
Automation Engineering

Tutor(s)
Häkkinen. Veli-Matti

Assigned by
Valio Oy

Abstract

The bachelor's thesis was assigned by Valio Oy Jyväskylä. Valio Oy has multiple sites all over Finland. Valio Oy has average of 4600 employees. Valio Oy's prime function is milk processing. The outcome of this thesis is a software used in controlling carton conveyors of Valio Oy Jyväskylä.

The goal of this thesis was renew and simplify the controls of carton conveyor during the renewing of carton conveyors as well as aim at reducing the amount of used photocells, removing the need for line pressure brakes and improving the functioning during jams and equipment failures. All sensors and motors are connected via ASi bus. The motors used are Sew Eurodrive energy efficient DRC motors with the speed control integrated into the ready solution and off the shelf connection to ASi bus with stop, run and two running speeds. All needed photocells could be connected through motors to the controlling PLC.

The thesis describes standards applicable to programming of control systems, what automation is and what the life cycle of automation looks like. Understanding these questions is a prerequisite for understanding the importance of quality of automation.

The design of software is discussed in further detail: what kind of solutions were decided for control problems and how they were implemented.

The result of this thesis is a software used in controlling carton conveyors at Valio Oy Jyväskylä and a basis on future renewal of carton conveyor systems.

Keywords/tags (subjects)
ISA88, PackML, Modular, Programming, Programmable logic controller, Automation

Miscellaneous

SISÄLTÖ

KUVIOT

1 Johdanto

Opinnäytetyöni sai alkunsa työnantajani Valio Oy:n Jyväskylän meijerin tarpeesta uusia harjapakkausten pakkauslaitteiston yhteydessä myös pakkauslinjan täyttökoneen ja jälkipakkauskoneiden väliset tölkkikuljettimet. Tavoitteena oli tehdä mahdollisimman helposti jatkossa monistettava sovellus ja samalla lisätä kuljettimien toiminnan järkevyyttä ja ylläpidon helppoutta. Aikaisemmin kuljettimien ohjaukset on toteutettu hyvin perinteisellä ajattelutavalla jossa kuljetinkokonaisuuden asetusaikoja kuten ruuhkapysäytys, tyhjäkäyntipysäytys ja ruuhkan jälkeisen käynnistyksen aikoja on voitu asettaa vain linja tasolla. Myöskään sovelluksen osien uudelleenkäytettävyyteen eikä ylläpidon helppouteen ole aikaisemmin kiinnitetty huomiota.

Jotta pakkauslaitteiston uusimisen jatkuessa eteenpäin saataisiin mahdollisimman suuri hyöty ohjelmiston uudelleenkäytettävyyden suhteen, on pyritty luomaa ohjelmapohja jossa perustoiminnot on eriytetty ja kapseloitu muuttuvasta osasta sovellusta. Samalla on minimoitu käytettävien signaalien kuten valokennojen käyttö, vähentäen myös laitteisto kustannuksia ja vikaantumispaikkoja.

Ensin käydään läpi mitä standardeja automaatiosovelluksen kehityksen tueksi on käytettävissä ja mitä ne tuovat sovelluksen kehitykseen mukaan. Lisäksi käydään läpi automaatiojärjestelmän elinkaari. Kaikkein pitkin vaihe automaatiojärjestelmän elinkaaressa on käyttövaihe, eli vaihe jossa järjestelmään kohdistuu ylläpito ja muutospaineita. Mahdollisimman helposti luettava ja modulaarinen

sovellus laskee suoraan ylläpitokuluja. Osiossa tutustutaan myös laatuun sekä sovelluskehityksen prosessiin ja sovelluksien perus arkkitehtuureihin.

Loppuosassa esitetään hyvin modularisoituvan ja siten uudelleenkäytettävän ohjelmiston perusrakenteita ja miten niitä on käytetty tölkkikuljettimien ohjauksen toteutuksessa. Lisäksi pohditaan miten standardien soveltamisesta saadaan maksimaalinen hyöty ja miten niitä työssä onnistuttiin käyttämään hyödyn tavoittamiseksi sekä millaisia edelleen kehityskohteita sovelluksessa on.

2 Valio Oy

Valio Oy on suomen suurin maidontuottaja, joka on perustettu vuonna 1905. Valion omistaa 18 eri osuuskuntaa, joissa on yhteensä noin 9500 maidontuottajaa. Valio konsernin liikevaihto oli vuonna 2013 yli 2 miljardia euroa josta investointeihin 118 miljoonaa euroa. Liikevaihdosta suurin osa, 46 %, tuli tuoretuotteista ja toiseksi suurin osa, 28 %, tuli juustoista. Henkilökuntaa on keskimäärin noin 4600. (Valio 2015).

Valion tehtävänä on Valiolaisten maidontuottajien elinkeinon edistäminen. Valion arvo on tehdä parasta. Toimintaperiaatteina edellisten toteutumiseksi Valio pyrkii tekemään todellisia tuoteuutuuksia, ohjaamaan toimintaansa asiakas- ja tuotekannattavuuden mukaan ja ottaa vastuun sekä oppii osaajilta. Valio pyrkii tekemään kestävää tulosta ja takaamaan tulevaisuuden maidon tuottajille. Henkilöstöperiaatteina Valio korostaa Työntekijöiden henkilökohtaista vastuuta, avoimuutta, oikeudenmukaisuutta sekä tasa-arvoa. (Valio 2015).

2.1 Jyväskylän meijeri

Kuvio 1. Valio Oy, Jyväskylä. Etualalla juuston kypsyttämö, keskellä meijeri sekä jakeluvarasto ja taaimpana autokorjaamo.

Valion Jyväskylän meijeri valmistui vuonna 1980 korvaamaan kaupungin keskustassa sijainnutta meijeriä. Samalla tontilla on kuvion 1 esittämällä tavalla juuston kypsyttämö, meijeri sekä jakeluvarasto. Laitosta on modernisoitu kovalla tahdilla läpi 2000 - luvun. Vuonna 2013 Jyväskylän meijeri otti vastaan 200 milj. litraa maitoa eli 18 täysperävaunullista rekkaa päivässä. Meijerin tärkeimmät tuotteet ovat maidot, kermat, piimät, sekä erikoismaidot.

2.2 Automaatio Jyväskylän meijerissä

Jyväskylän meijerin automaatioaste on hyvin suuri. Tuotanto on automatisoitu hyvin pitkälle alkaen maidon vastaanottamisesta loppuen

tuotteiden tilausten keräilyyn kulkien valmistuksen, pakkauksen sekä varastoinnin kautta.

Prosessiautomaatio on uudistettu 2004 lähtien täysin väyläpohjaiseksi, viimeiset osat uudistuivat 2009. Prosessia ohjaa Metso DNA DCS, jonka kaikki IO on hajautettu kentälle käyttäen Profibus DP, Profibus PA ja ASi väyliä.

Pakkausautomaatio on käytännössä kappaletavara-automaatiota, poikkeus tähän on nesteprosessin ja kappaletavara-automaation rajapintana toimivat täyttökoneet joilla tuote pakataan erimuotoisiin pakkauksiin kuten kaikkien tuntemaan Tetra Rex® harjapakkaukseen. Pakkausautomaatiolaitteistojen sovellusten ja käytettyjen ohjelmoitavien logiikoiden kirjo on suuri. Logiikoita on yli viideltä valmistajalta, kuten Rockwell automationin ControlLogix, Siemensin S7 ja Mitsubishin Q sarjat. Myös pakkausautomaation toteutuksessa on käytetty kattavasti erilaisia väyliä. DeviceNet, ASi , Profibus DP sekä Profinet ovat käytössä, esimerkiksi suuri osa tölkkiratojen ohjauksista on toteutettu ASi väylän avulla.

Varastointijärjestelmänä perinteisen massankäsittelyn lisäksi on pitkälle automatisoitu Cimcorpin robottivarasto järjestelmä. Varasto koostuu siirtovaunuista, kuljettimista, massaradoista sekä Cimcorpin MultiPick roboteista joilla voidaan keräillä asiakaskohtaisia tilauksia. Ohjaukset on toteutettu Siemens S7 sarjan logiikoilla joita ohjaa ylemmän tason pc pohjainen järjestelmä. Myös varastoautomaatiossa on käytetty erilaisia väyliä kuten Profibus DP ja ASi.

3 Automaatio

Automaatio on liiketoiminnan edistäjä. Tuottavuuden tehostaminen on ollut pitkään yksi keino säilyttää teollisuustuotannon edellytysten säilyttämiseksi. Automaatiolla on suuri kehityksen mahdollistajan rooli juuri tuotannon tehostamisessa. Lisäksi automaatio tuo huomattavaa lisäarvoa koneisiin, tuotannon järjestelmiin ja suunnittelupalveluihin. Globaalien markkinoiden, tekniikan kehityksen ja oman muuttuvan yhteiskuntamme vuorovaikutuksessa toimimiseksi yritykset tarvitsevat kykyä havaita muutoksia sekä taitoa ja ketteryyttä muuttua oikeaan suuntaan oikea-aikaisesti. Tämä on suuri haaste kaikille toimijoille, tarvitaan verkottumista haasteiden ymmärtämiseksi ja niihin tehokkaasti vastaamiseksi. Automaatiolla on merkittävä rooli lähes kaikessa teollisessa toiminnassa, sekä kasvavana osana myös ihmiselle suoremmin havaittavissa olevia asioita, kuten palveluita ja rakennuksia (Tekes 2010).

3.1 Automaatiojärjestelmä

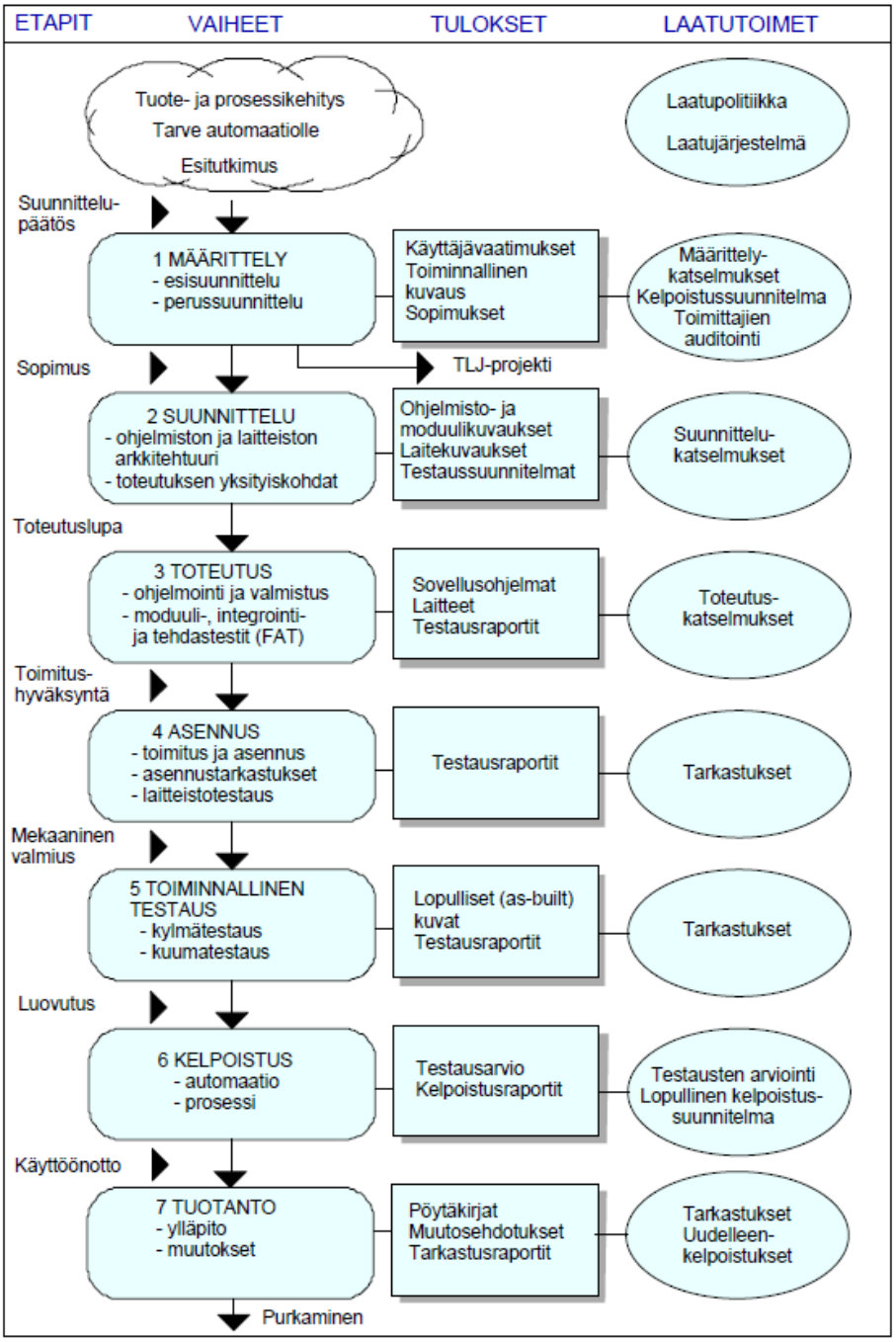
Automaatiotekniikalla tarkoitetaan toiminnan tai tehtävän suorittamista ilman jatkuvaa huomiota käyttäjältä. Automaatio tekniikkana käsittää mm. menetelmät, toteutustekniikan ja teorian (SAS 2001) Automaatiolaitteista eli ohjelmoitavasta ohjauslaitteesta (PLC, DCS) toimilaitteista, antureista ja ohjelmistosta koostuva järjestelmä on automaatiojärjestelmä. Automaatiojärjestelmä on jotain tiettyä tehtävää varten rakennettu järjestelmä, esimerkiksi tuotantolinjan tai tuotantoprosessin ohjausta varten (SAS 2005).

3.2 Automaatiojärjestelmän elinkaari

Järjestelmien monimutkaistuessa suunnitteluprosessin hallittavuuden vaateet ovat kasvaneet testaamisen kustannuksella. Suunnittelun ja toteutuksen laatu

tulee systemaattisesta toimintatavasta, valvonnasta ja testauksesta. Tämän takia laatuun ja turvallisuuteen liittyvät standardit rakentuvat elinkaari mallien ympärille. Tämä esittää automaatioprojektille suositeltavan mallin. Malli on tarkoitettu ensisijaisesti vaativiin järjestelmiin, eli kohteisiin joissa laadun dokumentoitu osoitus on tarpeen. Menettelyä voidaan tarvittaessa keventää kriittisyyden ja sovelluksen koon mukaan.

Elinkaaren vaiheet on esitetty kuviossa 2 siten, että vasemmalla on elinkaarivaiheet, keskellä dokumentit ja oikealla laadunvarmistukselliset tehtävät (SAS. 2001).



Kuvio 2. Automaatiojärjestelmälle laadun kannalta suositeltava elinkaarimalli (SAS. 2001)

3.2.1 Määrittely

Märittelyvaiheen tarkoitus on kuvata järjestelmä käyttäjän kannalta ja toteutusriippumattomasti niin tarkasti, että teknisesti yksityiskohtainen suunnittelu voidaan aloittaa. Määritysvaihe voi jakaantua erityisesti suuremmissa projekteissa kahteen vaiheeseen, esisuunnitteluun ja

perussuunnitteluun. Joissain tapauksissa esisuunnittelua voi edeltää esitutkimusvaihe, jonka pohjalta voidaan tehdä investointipäätös. Eli päätös aloitetaanko varsinainen projekti.

Esisuunnittelun tavoite on selvittää tarpeet ja koota ne käyttäjävaatimuksiin. Asiakas tarvitsee lisäksi kelpoistussuunnitelman, jota käytetään laadun valvomiseen ja ohjaamiseen. Esisuunnitteluvaiheen vastuu on asiakkaalla johon tarvittaessa käytetään konsulttia.

Perussuunnittelun tarkoitus on kuvata automaation toiminnot ja toteutusperiaatteet. Tärkeimpiä asioita ovat tarjouspyyntö, tarjoukset sekä sopimusneuvottelut joihin tulee mukaan toimittajien auditoinnin tulos. Tarjouspyyntöön liitteeksi tulee mm. käyttäjävaatimukset. Toimittajan tarjoukseen sisältyy toiminnallinen kuvaus sekä projekti ja laatusuunnitelma. Toiminnallinen kuvaus kuvaa lyhyesti laitteiston ja ohjelmiston rakenteen ja miten ne vastaavat asiakkaan vaatimuksiin. Dokumentteja tarkennetaan aina tarpeen mukaan neuvotteluiden aikana. Perussuunnitteluvaihe on yhteistyötä, mutta laajojen määrittelyiden tekemisestä voidaan sopia erikseen (SAS. 2001).

3.2.2 Suunnittelu

Suunnitteluvaiheen tarkoitus on tarkentaa määrittelyitä siten, että varsinaisen automaation toteutus voi alkaa. Tässä vaiheessa laiteriippumaton tieto voidaan tarkentaa tietylle laitteistolle. Kokonaisuuksien suunnittelusta edetään yksityiskohtien suunnitteluun, arkkitehtuurista, sovelluskuvauksen ja laitteistokuvauksen kautta yksityiskohtaiselle toiminta ja sekvenssikuvaus tasolle sekä näiden testaussuunnitelmat. Suunnitteluvaiheen vastuu on

yleensä vain toimittajalla toimittajan laatujärjestelmän mukaisesti. Keskustelut ajotavoista kuitenkin jatkuvat asiakkaan ja toimittajan välillä, sekä tarvittaessa kolmannen osapuolen välillä, joka saattaa vastata prosessisuunnittelusta (SAS. 2001).

3.2.3 Toteutus

Toteutusvaiheessa toimittaja hankkii, valmistaa ja kokoonpanee suunnitellun automaatiojärjestelmän laitteiston ja ohjelmiston. Vaiheen keskeinen osuus on tehdastesti (Factory Acceptance Test, FAT). Johon asiakas myös osallistuu. Toteutusvaiheen vastuu on toimittajalla toimittajan laadunhallintajärjestelmän mukaisesti, joka voi edelleen hankkia ulkopuoliselta valmistajalta soveltuvat kokonaisuudet. Vaiheen keskeiset syntyvät dokumentit ovat, asennus- ja käyttö-ohjeet, testaussuunnitelmat, tehdastestien dokumentaatio ja erilaiset raportit. Toteutusvaihe loppuu kun asiakas ja toimittaja hyväksyvät järjestelmän olevan valmis lähetettäväksi asiakkaan tiloihin (toimitushyväksyntä) (SAS. 2001).

3.2.4 Asennus

Asennusvaiheessa tehdastestattu järjestelmä ja siihen kuuluvat laitteet kuljetetaan, asennetaan ja kytketään paikalleen. Laitteistotestauksen jälkeen laitteisto on mekaanisesti ja sähköisesti toimiva. Kuljetuksen jälkeen asiakas suorittaa vastaanottotarkastuksen. Automaatiotoimittajan vastuulla on automaatioasennusten valvonta ja tarkastukset. Kriittisissä sovelluksissa myös asiakkaan tulee valvoa asennusten suorittamista. Asennuksen loppuessa järjestelmä on liitetty käyttöympäristöön ja automaatiojärjestelmä kenttälaitteisiin.

Leitteistotestauksessa instrumentit, piirit, laitteet ja ohjelmistot tarkistetaan alustavasti. Sen jälkeen järjestelmän tulee olla sähköisesti ja mekaanisesti piirustusten ja ohjeiden mukainen. Vaihe päättyy hyväksyttyyn asennusraporttiin, jolloin järjestelmä on valmis toiminnallista testausta varten, sekä käyttövastuu siirtyy käyttöhenkilökunnalle projekti- ja ylösajohenkilökunnalta (SAS. 2001).

3.2.5 Toiminnallinen testaus

Toiminnallisessa testauksessa varmistetaan automaatiojärjestelmän ja laitteiden toimivuus yksittäin ja kokonaisuutena. Tuloksena järjestelmä on toimittajan puolesta valmis tuotanto käyttöön. Tärkeimmät vaiheet ovat kylmä- ja kuumatestaus.

Kylmätestauksessa testataan järjestelmä mahdollisimman vaarattomilla materiaaleilla, esim. vedellä. Tässä vaiheessa testataan hälytykset, lukitukset, suojaukset yms. sekä normaalitilan yksittäiset toiminnot. Testauksen jälkeen järjestelmä on valmis ylösajoon, jossa varsinaiset tuotantomateriaalit otetaan käyttöön.

Kuumatestauksessa laitteiston kaikki sovellukset testataan mahdollisuuksien mukaan normaalilla materiaalilla. Tällä ja tarvittaessa erillisellä hyväksymistestauksella (Site Acceptance Testing, SAT) toimittaja osoittaa asiakkaalle järjestelmän olevan kuvauksen mukainen, virheet korjattuna ja valmis luovutukseen. Teknisesti järjestelmä on valmis tuotantoon ja takuuaika voi alkaa sopimuksen mukaan (SAS. 2001).

3.2.6 Kelpoistus

Automaatiojärjestelmän asiakkaalle luovutuksen jälkeen alkaa kelpoistusvaihe. Kelpoistuksen tavoite on osoittaa järjestelmän toimivan siten, että tuotettu tuote vastaa toistuvasti spesifikaatiota. Kelpoistuksen vastuu on asiakkaalla.

Automaation teknisellä loppukelpoistuksella osoitetaan, että järjestelmä on suunniteltu ja toteutettu vaatimusten mukaisesti. Keskeistä on dokumenttien tarkastus, toimittajan toiminnan ja testauksen arviointi, tarvittava kelpoistustestaus sekä kelpoistusraportin laatiminen ja arkistointi. Loppukelpoistusvaiheen aikana myös suorituskyky testataan, jossa todetaan suorituskykyvaatimusten toteutuminen. Kelpoistusvaiheessa havaitut virheet voivat estää järjestelmän hyväksynnän.

Tekninen loppukelpoistus päättyy hyväksyttyyn loppukelpoistusraporttiin ja syntynyt automaation sovellusversio voidaan ottaa tuotantokäyttöön. Automaatiojärjestelmän loppukelpoistus ei ole tuotannon kelpoistus vaan teknisen järjestelmän kelpoistus. Varsinaisen tuotantoprosessin asetukset ja viritykset tehdään prosessin kelpoistusvaiheessa tuotekohtaisesti ja useita panoksia tai pitkää jatkuvaa prosessia ajamalla. Joissain tapauksissa tuotteiden kelpoistaminen voikin tulla vuodenkin päästä (SAS. 2001).

3.2.7 Tuotanto

Kun järjestelmä on kelpoistettu hyväksytysti, alkaa tuotantovaihe jossa elinkaari jatkuu ylläpidolla ja useasti myös muutoksilla. Automaatiojärjestelmään kohdistuvat muutokset aiheuttavan uuden kelpoistuksen.

Jo suunnittelussa tulisi ottaa huomioon, että automaatiojärjestelmän elinkaari loppuu järjestelmän käytöstä poistoon ja hävittämiseen. Järjestelmän hallittuun uusimiseen tulisi siis varautua.

Tuotantovaiheen tehtävistä vastaa asiakas ja tarvittaessa asiakas käynnistää uusia projekteja omilla elinkaarillaan. Asiakas saattaa hankkia palveluita myös järjestelmätoimittajalta (SAS. 2001).

3.3 Automaation laatu

Automaation laatu siis perustuu toisaalta systemaattisiin toimintatapoihin ja toisaalta valvontaan ja testaamiseen. Laadun tuottamiselle tärkeää on ennakkosuunnittelu, suunnittelun rinnalla etenevä laadun varmistus ja kattava dokumentointi. Laatu riippuu myös johtamiskäytännöistä, menetelmistä, työkaluista sekä suunnittelijoiden ammattitaidosta ja vastuuntunnosta.

Järjestelmien monimutkaistuminen kasvattaa hallittujen toimintatapojen merkitystä. Automaatiosuunnittelun laatua tuotetaan hyvillä ratkaisuilla ja arvioimalla (testaus, katselmus) ratkaisuja. Varmistamalla, että laadunvarmistamiseksi olevaa ohjeistusta noudatetaan. Tärkein periaate laadunvarmistuksessa on kuitenkin, että varmistava taho on riippumaton tarkastettavan kohteen suunnittelijasta ja toteuttajasta (SAS. 2001).

3.4 Ohjelmistojen laatu

Sovellusten monimutkaistuessa suunnittelun hallinta on tullut vaikeammaksi. Kilpailu on ajanut panokset ohjelmistotuotannon ja ylläpidon tehokkuuteen.

Nykyään myös laatu on noussut tärkeäksi kysymykseksi. Ohjelmistojen laadun varmistaminen ei ole helppoa. Ohjelmistosuunnittelu on useiden ihmisten ja organisaatioiden luovaa työtä, jossa on monia epävarmuuksia. Teknisesti ohjelmistot eroavat mekaanisista järjestelmistä, joka vaikeuttaa laadun osoittamista. Ohjelmistot eivät vanhene, eivätkä vikaannu kuten mekaaniset laitteet ja pienikin muutos ohjelmakoodissa tai syötteissä (sisääntulot, sisäiset tilat yms.) voivat aiheuttaa arvaamattomia muutoksia ohjelmiston käyttäytymisessä. Vikaantuminen onkin käytännössä aina ihmisen virhe, joka voi syntyä jo määrittelyissä, toteutuksessa tai ylläpidon aikana. Sovelluksia on mahdotonta testata kattavasti, joten systemaattiset menetelmät ja osaavat suunnittelijat ovat elintärkeitä laadun aikaansaamiseksi. Kun sovelluksia kootaan yhä enemmän valmiita komponentteja (ohjelma- ja ohjelmalohkokirjastot yms.) käyttäen tulee versioiden yhteensopivuus ja laatu ongelmaksi. Tätä pyritään ratkaisemaan standardoimalla ohjelmistoja (SAS. 2001).

4 Standardi IEC61131

Kansainvälinen IEC 61131 Ohjelmoitavat ohjauslaitteet standardi kuvaa ohjelmoitavan logiikan toiminnan. Logiikoille on aikaisemminkin pyritty luomaan standardeja, mutta vasta IEC 61131 on saavuttanut teollisuuden hyväksynnän. Standardi on jaettu alla lueteltuihin osiin:

- IEC 61131-1 - Ohjelmoitavat ohjauslaitteet – Osa 1: Yleisinformaatio
- IEC 61131-2 - Ohjelmoitavat ohjauslaitteet – Osa 2: Laitevaatimukset
- IEC 61131-3 - Ohjelmoitavat ohjauslaitteet – Osa 3: Ohjelmointikielet
- IEC 61131-4 - Ohjelmoitavat ohjauslaitteet – Osa 4: Käyttäjän ohjeita
- IEC 61131-5 - Ohjelmoitavat ohjauslaitteet – Osa 5: Tiedonsiirto
- IEC 61131-6 - Ohjelmoitavat ohjauslaitteet – Osa 6: Toiminnallinen turvallisuus

- IEC 61131-7 - Ohjelmoitavat ohjauslaitteet – Osa 7: Fuzzy – ohjauksien ohjelmointi
- IEC/TR 61131-8 - Ohjelmoitavat ohjauslaitteet – Osa 8: Ohjausjärjestelmien ohjelmointikielten käyttö ja soveltaminen. Tekninen raportti

Ohjelmoinnin kannalta oleellisin on 61131-3, joka keskittyy ohjelmointiin ja voidaan nähdä paremminkin ohjeena kuin tiukkoina sääntöinä.

4.1 IEC61131 Ohjelman rakenne

Standardi kuvaa ohjelmarakenteen siten, että ohjelma koostuu osista. Näitä osia ovat program (ohjelma) function (funktio) ja function block (toimintalohko).

Ohjelma on ns. sisääntulopiste ohjelman suoritusta varten. Ohjelma yleensä koostuu pääosin muiden tyyppien (funktio, toimintalohko) kutsuista. Funktiolle voidaan antaa parametrejä, se ei muista edellistä suorituskertaa eli sillä ei ole omaa pysyvää muistia ja sen tulisi palauttaa sama arvo samoilla parametrien arvoilla. Toimintalohkoilla on oma muistinsa, näiden ulostulo siis riippuu paitsi annetuista parametreistä, myös sisäisen muistin tilasta.

	Program	Function Block	Function
VAR	*	*	*
VAR_INPUT	*	*	*
VAR_OUTPUT	*	*	
VAR_IN_OUT	*	*	
VAR_EXTERNAL	*	*	
VAR_GLOBAL	*		
VAR_ACCESS	*		

VAR on sisäinen toimintalohkon muuttuja, jonka tila pysyy kutsujen välillä. Lisäksi on VAR_TEMP, joka on kutsun sisällä väliaikaista käyttöä varten oleva muuttuja, jonka tilaa ei tiedetä, ennen kuin siihen kirjoitetaan.

VAR_INPUT tyypit ovat sisään luettavia parametrejä, VAR_OUTPUT ulos kirjoitettavia parametrejä ja VAR_IN_OUT parametrejä jotka luetaan kun toimintalohko kutsutaan, sekä kirjoitetaan ulos kun toimintalohkon suoritus loppuu.

Varsinaiseen ohjelmien, funktioiden ja toimintalohkojen ohjelmointiin käytetään IEC 61131-3 osan määrittämiä ohjelmointikieliä, nämä esitellään jälkeenpäin.

Muuttujien käyttö informaation säilytykseen ja välittämiseen on merkittävä osuus IEC 61131 standardia. Vanhimmissa järjestelmissä tietoon viitattiin muistiosoitteen perusteella. Tämä tapa vaatii huomattavaa työtä muistin hallinnan kannalta, koska muistiosoitteiden hallinnointi varsinkin suurissa järjestelmissä on virhealtista. Suoraa muistiosoitetta käytettäessä on tiedettävä tarvitun muistin koko, ettei käytä päällekkäistä muistialuetta eri datan tallentamiseen.

IEC 61131 standardin mukainen tapa käsitellä muuttujia on sellainen jossa muuttujilla on myös tietotyyppi (datatype). Nimi yksilöi muuttujan ja tietotyyppi kuvaa muuttujan sisällön ja millaisia arvoja se voi saada. Tämä periaatteessa vapauttaa ohjelmoijan muistin hallinnasta, aina näin ei kuitenkaan ole vaan valmistajien implementaatiot usein sisältävät myös absoluuttisen muistiosoitteen. Toisaalta jotkin sovellukset vaativat absoluuttisen muistiosoitteen käyttöä. Siitä huolimatta kuvaavien ja helposti muistettavien muuttujan nimien käyttö tehostaa ohjelmointityötä, koska muistin käytön ongelmat on ainakin minimoitu. IEC 61131 määrittää

tietotyypit joita voidaan käyttää, jotta eri valmistajien järjestelmät olisivat mahdollisimman yhdenmukaisia ja yhteensopivia (IEC 2013).

Boolean-tyypit	Kokonaisluvut etumerkillä	Kokonaisluvut ilman etumerkkiä	Liukuluvut	Aika, kesto, pvm, merkkijono
BOOL				TIME
BYTE	SINT	USINT		DATE
WORD	INT	UINT		TIME_OF_DAY
DWORD	DINT	UDINT	REAL	DATE_AND_TIME
LWORD	LINT	ULINT	LREAL	STRING

4.1.1 Muuttujatyypit

IEC61131 standardin tavoite on yhtenäistää ohjelmointiympäristöjä esimerkiksi vakioimalla ohjelmointikielet, ohjelmalohkot ja käytettävät tietotyypit. Kun peruselementit ovat eri ympäristöissä samat, on ohjelmoijien helpompi siirtyä eri ympäristöjen välillä. Käytännössä ympäristöt saattavat erota edelleenkin suuresti toisistaan, koska IEC61131 standardissa ei ole määritetty minimi vaatimuksia. Parhaiten järjestelmästä toiseen siirrettävä ohjelmointikieli onkin Structured Text sen tekstipohjaisuuden ja korkeatasoisuuden vuoksi (IEC 2013).

4.2 IEC61131-3 ohjelmointikielet

IEC61131-3 määrittää ohjelmointiin viisi ohjelmointikieltä joista kaksi on tekstimuotoisia ja kolme graafisia. Näiden lisäksi on yksi yleisesti käytössä oleva täydentävä kieli. Kieliä monissa järjestelmissä on siis yhteensä kuusi joista viisi varsinaisia IEC61131-3 standardin kieliä (IEC 2013).

Graafiset kielet:

- Ladder diagram (LAD)
- Functionblock diagram (FBD)
- Sequential Function Chart (SFC)

- Continuous Function Chart (CFC)

Tekstimuotoiset kielet:

- Instruction list (IL)
- Structured Text (ST)

Ohjelmoija voi valita eri käyttötarkoitukseen parhaiten soveltuvan kielen. On kuitenkin tärkeää muistaa automaatiojärjestelmän koko elinkaari ohjelmointikieltä valittaessa. Erityisesti korkeamman tason ST ja IL tuottavat ylläpitohenkilöstölle helposti vaikeuksia.

4.2.1 Instruction list

Instruction List on eniten assemblyä muistuttava ohjelmointikieli. Kieli on hyvin moneen taipuva ja erityisesti LAD ja FBD on usein mahdollista kääntää IL muotoon. Instruction List on rivi pohjainen kieli kuten kuviosta 3 näkee, eli yksi operaatio kuvataan yhdellä rivillä.

```
FUNCTION "GenLMS" : VOID

VERSION: 1.3
AUTHOR: Muotio
FAMILY: ASi

VAR_INPUT
    ASiListDataBlock : BLOCK_DB;
END_VAR

VAR_TEMP
    iCount, jCount :INT;
END_VAR

BEGIN

OPN #ASiListDataBlock;
LAR1 p#0.0;

//Tee puuttuvien orjien lista ASi master1 ja 2

l 2;
lop2: t jCount;

l 2;
lop1: t iCount;

l dbd[ar1,p#24.0]; //lataa lps
l dbd[ar1,p#0.0];  //lataa las
ad;
xod;
t dbd[ar1,p#32.0]; //siirrä lms

+ar1;

l #iCount;
loop lop1;

lar1 p#40.0;      //siirrä osoitin master 2 datojen kohdalle

l jCount;
loop lop2;

END_FUNCTION
```

Kuvio 3. IL (STL) lähdekoodi editori ja ohjelmakoodia, Simatic Manager V5.5

4.2.2 Structured Text

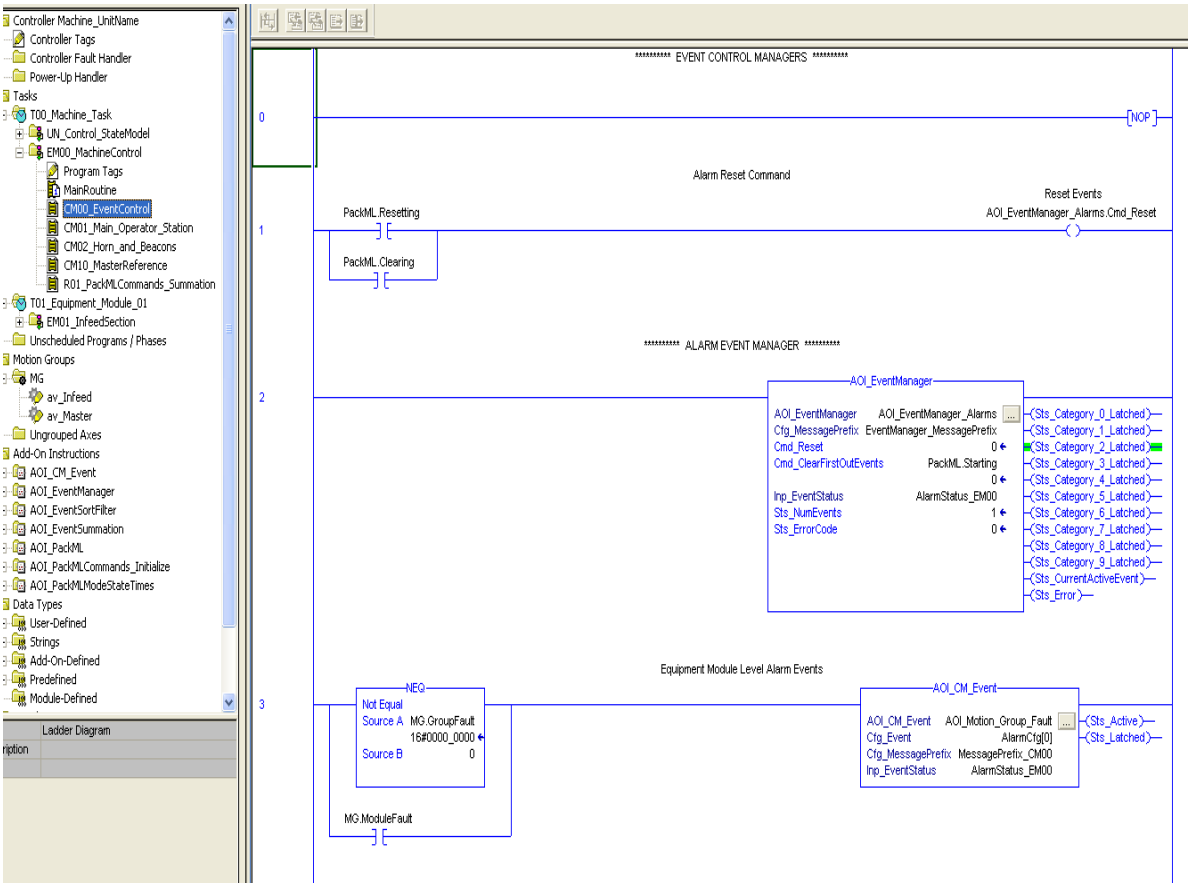
Structured Text on syntaksiltaan hyvin Pascalin tapainen kieli, täten se on myös korkeamman tason ohjelmointikieli (kuvio 4). ST soveltuu parhaiten matemaattisiin toimintoihin, tiedonsiirtoon liittyviin toimintoihin ja muihin monimutkaisia kontrollirakenteita tarvitseviin toimintoihin. Structured Text on monesti edukas verrattuna IL ohjelmointiin, erityisesti ohjelman kulun ohjaus, rakenteiden ryhmittely ja luettavuus on huomattavasti parempaa. Structured Text on tietyllä tapaa vapaamuotoinen kieli ja lause voi jatkua riviltä toiselle tai yksi rivi voi sisältää monta lausetta. Structured Text kieltä käytettäessä onkin kiinnitettävä huomiota ohjelmoinnin rakenteeseen, että luettavuuden etu säilyy.

```
(*
Valinta. Kun valinta vaihdetaan, odotetaan aika johon mennessä valinnan
tulee olla vaihtunut. Jos valinnan vaihto ei jostain syystä mene läpi, valittu valinta vaihdetaan takaisin aikaisemmaksi valinnaksi.
*)
#ValintaAjastinKäy := false;

CASE #State OF
    0: //Valinta valittu, odotetaan valinnan vaihtoa paneelilta tai vaihtumista metsolta

        //Valinta vaihtuu paneelilta
        IF #HmiValinta <> #VanhaHmiValinta THEN
            #State := 1;
        END_IF;

        //Valinta vaihtuu metsolta
        IF #MetsoValinta <> #HmiValinta AND #HmiValinta = #VanhaHmiValinta THEN
            #State := 2;
        END_IF;

    1: //Valinnan vaihto menossa paneelilta
        #ValintaAjastinKäy := true;
        //Aika kului, valinta ei mennyt läpi
        IF #ValintaAjastin.Q THEN
            #State := 2;
        END_IF;

        //Valinta meni läpi
        IF #MetsoValinta = #HmiValinta THEN
            #State := 0;
        END_IF;

    2: //Valinta vaihtuu metsolta
        #HmiValinta := #MetsoValinta;
        #State := 0;
```

Kuvio 4. ST (SCL) editori ja ohjelmakoodia, TIA Portal V13

4.2.3 Ladder Diagram

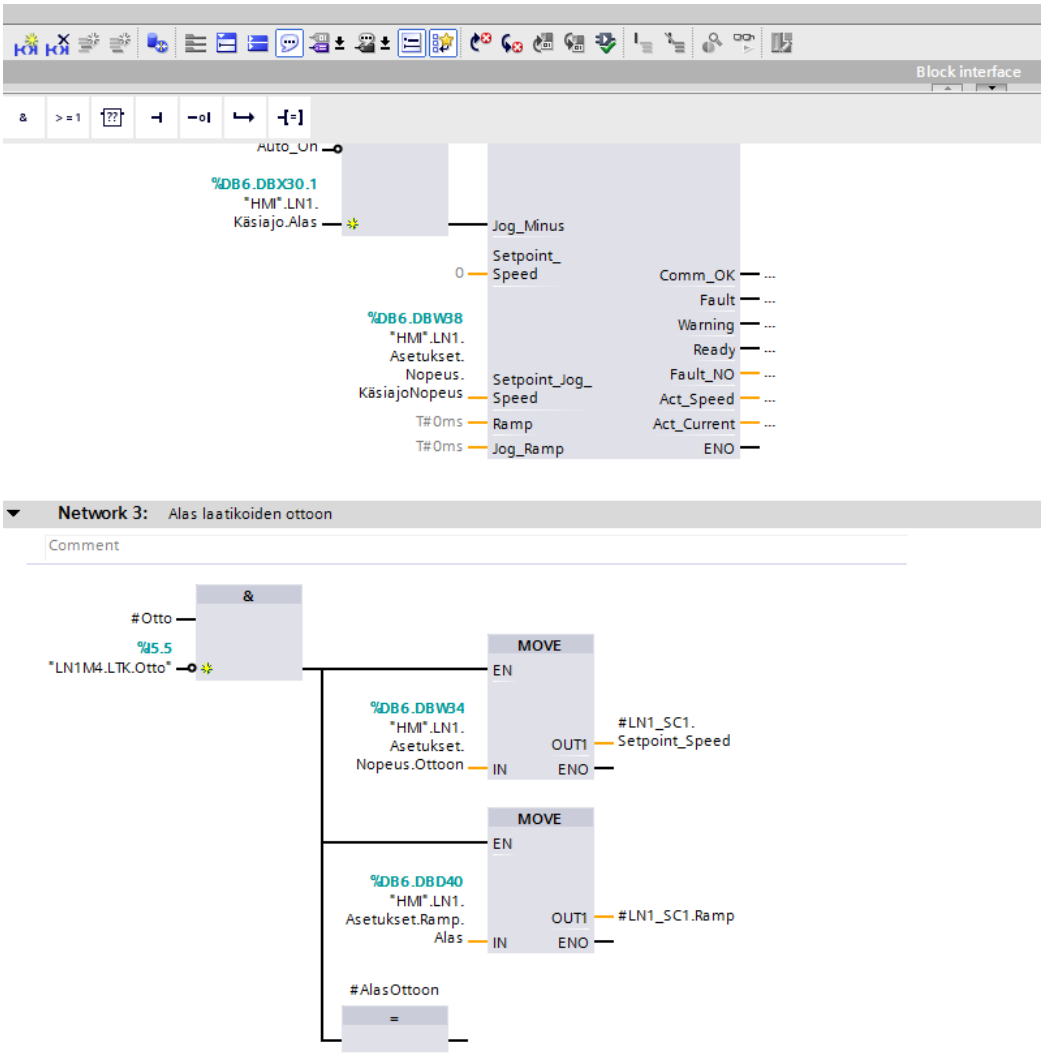
Ladder diagram kieli perustuu virtapiiri malliin, jossa kuvio 5:n mukaisesti yleensä vasemmassa reunassa on virtakisko ja oikeassa reunassa mahdollisesti nolla kisko sekä näiden välissä olevasta piiristä (rung, ”tikapuun askel”), ladder muistuttaakin tikapuita josta se onkin saanut nimensä. Virta kulkee vasemmalta oikealle riippuen elementtien tilasta. Ohjelma suoritetaan ylhäältä alas ja vasemmalta oikealle, hypyt ovat mahdollisia. Ladder diagram muistuttaa 90 astetta käännettyä piirikaaviota ja siksi onkin monesti helpoin sähköalan taustan omaaville henkilöille. Parhaimmillaan Ladder on yksinkertaisissa ohjauksissa, mutta sillä voidaan toteuttaa myös monimutkaisempia järjestelmiä luettavuuden kustannuksella.



Kuvio 5. Ladder editori ja ohjelmakoodia, RsLogix5000

4.2.4 Function Block Diagram

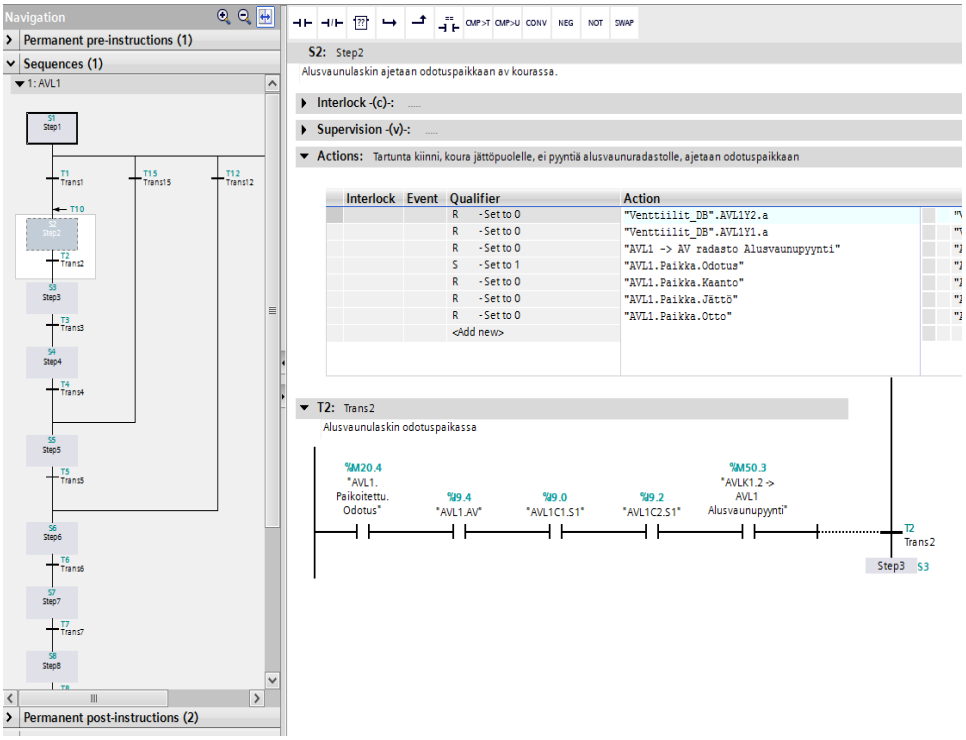
Function Block Diagram on graafinen kieli jossa ohjelmointi toteutetaan lohkoilla ja yhdistämällä näitä toisiinsa. Lohkot koostuvat sisääntuloista, itse lohkosta joka kuvaa sen toiminnan sekä lohkon lähdöistä kuten kuviosta 6 näkee. Ohjelman suoritus tapahtuu ylhäältä alas ja vasemmalta oikealle kuten Ladderissakin sekä hypyt ovat mahdollisia. Monissa ohjelmointiympäristöissä Ladderin ja Function Block Diagramin ero ei ole suuri, sillä Ladderiin yleensä voidaan lisätä toimilohkoja. Function Block Diagram sopii parhaiten suhteellisen yksinkertaisiin tehtäviin kuitenkin pysyen luettavampana kuin Ladder myös hieman monimutkaisemmissa ohjelmistoissa.



Kuvio 6. Function Block Diagram editori ja ohjelmakoodia, TIA Portal V13

4.2.5 Sequential Function Chart

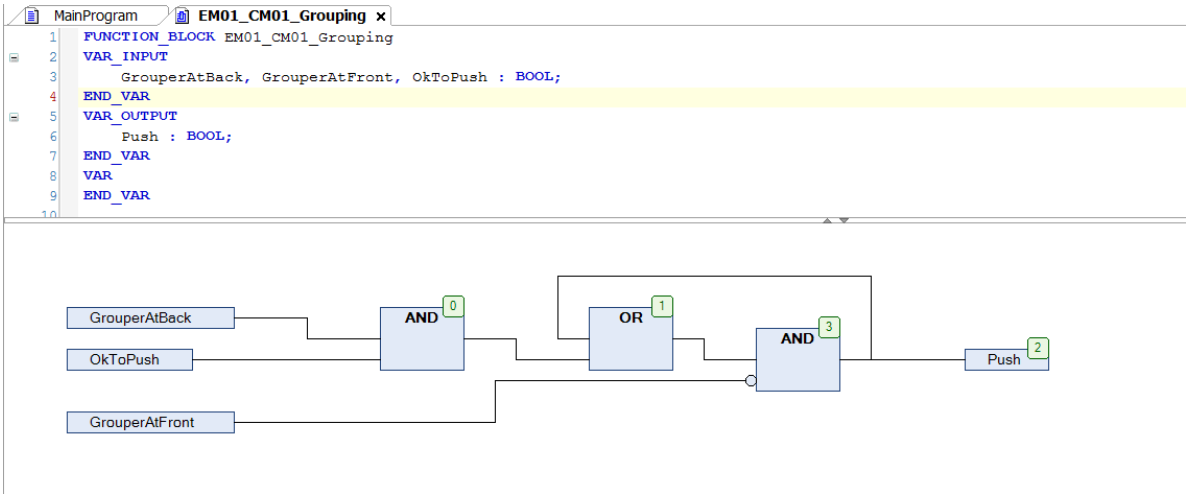
Sequential Function Chart (SFC) on graafinen ohjelmointikieli, joka on erityisesti suunniteltu sekvenssimuotoisen ohjelman kuvaamiseen. SFC koostuu askelista (Step), askelissa tapahtuvista toiminnoista (action) sekä siirtymä ehdoista (transition). Askeleen ollessa aktiivinen, kuvaa toiminnot mitä ohjelma tekee kyseisen vaiheen aikana. Siirtoehdot määrittävät mitkä askeleet aktivoituvat siirtoehtojen täyttyessä. SFC:ssä voi olla samaan aikaan suoritettavia askel haaroja (”simultaneous branch”) jossa useampi askel suoritetaan samaan aikaan ja seuraavaan askeleeseen siirrytään vasta kun kaikkien askeleiden siirtymäehdot ovat täyttyneet, tai muotoisista askel haaroista (”alternate branch”) jossa suoritetaan ohjelman perusteella jokin askelreitti suorittamatta jotain toista, haarojen suluista, hypyistä sekä sekvenssin lopusta (kuvio 7). Kerran suoritettavan sekvenssin loppuun asetetaan sekvenssin lopetus, jatkuvasti toimivaan sekvenssiin asetetaan hyppy askeleeseen josta sekvenssi jatkaa uudelle kierrokselle.



Kuvio 7. SFC (Graph7) editori ja ohjelmakoodia, TIA Portal V13

4.2.6 Continuous Flow Chart

Continuous Flow Chart on IEC61131-3:a täydentävä graafinen ohelmointikieli. Se muistuttaa lähinnä Function Block Diagrammia, joka voidaan vapaasti sijoitella ohjelmointialueelle (chart). Kielessä kuvataan lohkojen suoritusjärjestys sekä takaisinkytkentätiedot ovat mahdollisia, kuten kuviossa 8 on esitetty. Kieli sopii erittäin hyvin prosessien säätöpiirien ja vastaavien jatkuvien toimintojen ohjelmointiin.



Kuvio 8. CFC editori ja ohjelmakoodia, Codesys V3.5

5 Standardit ja ohjeet

5.1 ISA88

Tuotanto operaatiot voidaan yleensä luokitella kuuluvan yhteen kolmesta vaihtoehdosta, diskreetti, jatkuva tai panos.

Panosprosesseissa yhdistetään raaka-aineet halutuissa määrissä lopputuotteen tai välituotteen muodostamista varten. Sekoitus tai reaktio vaiheen aikaan uuden raaka-aineen syöttö pysähtyy. Uusi panos voidaan aloittaa vasta kun edellinen panos on valmis ja valmistunut tuote on siirretty eteenpäin.
Yleensä panoksien teossa käytetään säiliöitä tai sekoittajia joihin raaka-aineet annostellaan ja kun halutut aineet on valmisteltu, siirretään seuraavaan osaan prosessia, jossa mahdollisesti alkaa uusi vaihe.
Panos prosesseissa voidaan tehdä erilaisia tuotteita muuttamalla:

- Raaka-aineita
- Määriä
- Aikoja
- Energiaa (lämpötilat, paineet jne.)

ISA-88 on työkalu joustavuuden maksimoimiseen.

Jatkuvassa prosessissa on jatkuva raaka-aine virta sekä jatkuva valmiin tuotteen virta. Jossain tapauksissa jatkuvan prosessin käynnistäminen vaatii enemmän työtä kuin sen ylläpito.
Esimerkiksi sähkön tuotanto, vesijohtoveden tuotanto ja öljynjalostus ovat jatkuvatoimisia prosesseja.
Myös näissä tapauksissa muutostarpeita voi olla, esimerkiksi öljynjalostuksessa talvidieselin tuotanto vs kesädieselin tuotanto.

ISA-88 toimii myös tällaisissa tapauksissa työkaluna joustavan rakenteen luomiseen

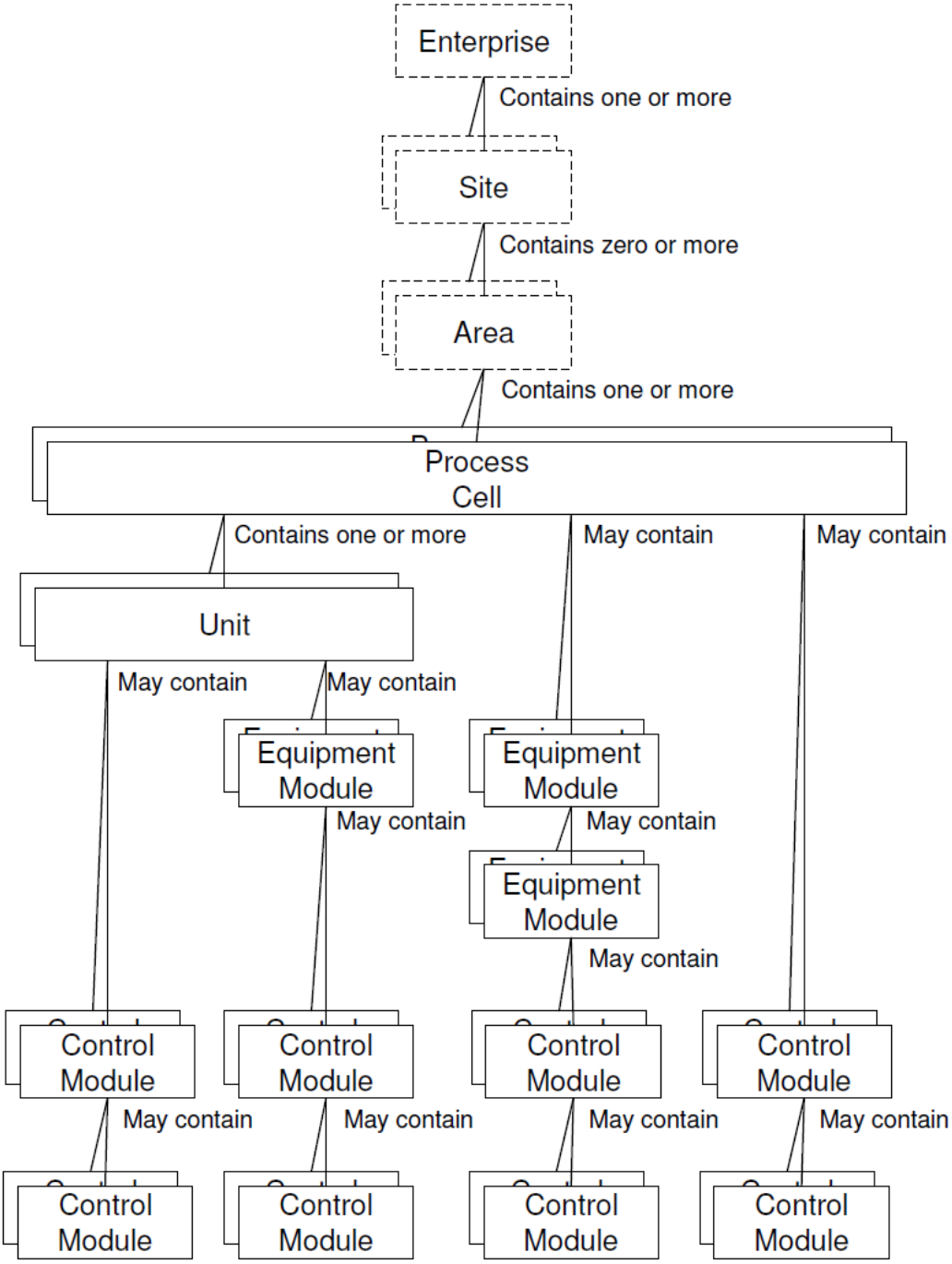
Diskreetissä prosessissa tuotteita tehdään osa kerrallaan, pala palalta. Yksi osa tai jokin tietty määrä osia siirtyy työvaiheesta toiseen. Jokainen työvaihe tuo lisäarvoa. Esimerkiksi autojen valmistus on diskreetti prosessi jossa tuotannon on oltava joustava mahdollistaakseen asiakkaiden tarpeisiin vastaamisen. Tyypillisiä diskreetin prosessin työasemia ovat erilaiset kuljettimet, robotit, manipulaattorit ja pakkauslaitteet.

ISA-88 on alkujaan panoprosessin suunnitteluun ja mallintamiseen kehitetty standardi, sitä voidaan kuitenkin hyödyntää myös jatkuvan ja diskreetin prosessin kuvaamiseen. ISA-88 ei ole ohjelmisto standardi vaan sitä voidaan käyttää myös manuaalisesti ohjatun prosessin kuvaukseen.
ISA-88 tarjoaa johdonmukaisen terminologian ja standardit panosprosessin mallintamiseen määrittämällä fyysisen mallin, proseduurit ja reseptit. Standardilla haluttiin vastata ongelmiin kuten yleisen panosprosessin mallintamisen puute, käyttäjävaatimusten teon ja kommunikoinnin vaikeus, automaatio toimittajien integroinnin vaikeus sekä itse panoprosessin konfiguroinnin vaikeus. Yleisesti siis ongelmiin joita eri suunnittelualat (esim. prosessisuunnittelu vs. automaatiosuunnittelu) ajattelevat eri tavalla ja on tarve päästä puhumaan asioista yhteisellä kielellä.
Standardi määrittää prosessimallin missä on määritellyt prosessin vaiheet, jotka koostuvat määritellyistä prosessioperaatioista, jotka vuorostaan koostuvat määritellyistä prosessi toiminnoista.
ISA88 sisältää osat:

- ISA-88.01, Batch Control Part 1 : Models and Terminology

- ISA-88.02, Batch Control Part 2: Data Structures and Guidelines for Languages
- ISA-88.03, Batch Control Part 3: General and Site Recipe Models and Representation
- ISA-88.04, Batch Control Part 4: Batch Production Records
- ISA-88.05, Part 5: Implementation, Models & Terminology for Modular Equipment Control (kehityksessä)

Erityisesti diskreetin prosessin kannalta kiinnostavia ovat Part 1, 5 sekä TR2 (ISA88). ISA88 jakaa kuvattavan prosessin osiin seuraavasti kuvion 9 esittämällä tavalla.



Kuvio 9. Fyysinen malli (ANSI-ISA88.01)

Kuvion 9 esittämä fyysinen malli lähtee ”enterprise” tasolta, jonka tulee sisältää ”site” taso, joka voi sisältää ”area” tasoja, joka voi sisältää ”process cell” tasoja, jonka on sisällettävä vähintään ”unit” taso, joka voi sisältää

”equipment module” tasoja jotka voivat sisältää ”control module” tasoja. Osa tasoista voidaan jättää käyttämättä poislukien ”unit” taso (ISA88.01).

5.2 ISA-88 TR2 (tekninen raportti)

ISA88 komitea on määrittänyt standardi sarjan panos prosessien kuvaamiseen käytettäviä malleja ja konsepteja. Näitä standardeja ei kuitenkaan ole tehty pakkauskoneita ja muita diskreettejä laitteistoja silmälläpitäen. ISA-88 standardien kehittyessä, voidaan standardimallia laajentaa koskemaan koko tehdasta siten, että mallinnus tapahtuu samalla tavalla oli kyse sitten panosprosessista, pakkausautomaatiosta tai varastoinnista. ISA-88 TR2 avaa ajattelua miten ISA-88 standardia voidaan käyttää käsittämään kaikki automaattiset koneet.

ISA-88 TR2 on informatiivinen ja esittää esimerkein miten ISA-88 standardia voidaan soveltaa käytettäväksi yhteisen ylemmän tason ohjelmistoarkkitehtuurin tai layoutin rakentamiseen. Termit ja määritykset seuraavat ISA-88 standardia niin pitkälle kuin mahdollista, mutta mallit ovat laajennettuja versioita ISA-88:n malleista.

Teknisen raportin tarkoitus on:

- Selittää funktionaalinen tila ohjelmointimalli automaattisille laitteille
- Esittää yleiset määritykset ja terminologia
- Selittää miten käyttää tila ohjelmointimallia automaattisille laitteille
- Antaa todellisia esimerkkejä sekä pohjia
- Esittää yhteinen rakenne ja malli tageille

ISA-88 TR2 on tehty soveltaen OMAC workgroupin PackML ohjeistusta (ISA-TR88.00.02).

5.2.1 PackML

PackML on OMAC:n packaging workgroup:in kehittämä standardi. PackML pääsi versioon 3 asti. Tämän jälkeen PackML yhdistettiin ISA88 standardeihin ISA88-TR2 avulla. Tällä hetkellä puhuttaessa PackML:stä tulee huomata, että tarkoitetaan juuri ISA88-TR2:ta. PackML nimikkeellä kuitenkin löytyy huomattava määrä erilaisia resursseja. Näistä esimerkkinä useat valmistajat ovat tuottaneet valmiita pohjia ohjelmointiympäristöihinsä. (OMAC).

6 Modulaarisuus

Modulaarisuuden tarkoitus on parantaa ohjelmistojen laatua, yhdenmukaisuutta sekä uudelleen käytettävyyttä. Tämän saavuttamiseksi henkilöiden tulee ymmärtää tarpeeksi laajasti ongelmat joihin pyritään vastaamaan, sekä työkalut joilla ongelmiin pyritään vastaamaan. Aiemmin on esitetty standardit sekä automaation elinkaari.
Modulaarisen sovellusohjelmoinnin hyödyt
Modulaarinen sovellusohjelmointi hyödyttää sovelluskehityksestä vastaavaa yritystä, sekä loppukäytön tekevää yritystä.

Loppukäyttäjät saavat esimerkiksi seuraavia hyötyjä:

Loppukäyttäjän tarve	Modulaarisuuden vastaus
Tuotannon joustavuus, tuotannon muokkaus mahdollisimman kustannustehokkaasti ja nopeasti tarpeiden muuttuessa.	Laitteiston ja toimintojen erottaminen toisistaan sekä muut modulaarisuutta lisäävät konseptit.
Parempi tuotanto tehokkuus. Jatkuvien parannustöiden teko ja	Yhteinen kieli ja tapa esittää laitteiden tilatiedot, mikä on

seuranta tuotannon informaatio järjestelmien avulla.	kriittinen edellytys mittausjärjestelmän standardi rajapinnan tekemiseen.

Ohjelmistoja tuottava yritys saa esimerkiksi seuraavia hyötyjä:

Tuottajan tarve	Modulaarisuuden vastaus
Markkinoille saantiaika	Modulaarinen ohjelmointi mahdollistaa valmiita kirjastoja, sekä mahdollistaa keskittymisen laitesidonnaisiin ohjausmoduuleihin. Joka vastaavasti nopeuttaa kehitystyötä.
Laatu	Modulaarisuus helpottaa eri ohjelmisto-osien testausta, kun eri osat ovat erotettu toisistaan ja muutos johonkin moduuliin ei vaikuta muihin.
Kokonaiskustannukset	Kun modulaarisuutta käytetään kattavasti, laskee kokonaiskustannus suunnittelun, tuottamisen sekä käyttöönoton osalta.

Modulaarisen ohjauksen konseptit

Aikaisemmin kuvatut standardit tukevat erityisesti automaation lähtökohdista tehtyä modulaarista ohjelmointia. Näiden lisäksi tietokone

sovellusten ohjelmoinnissa ollaan hyvin pitkällä OOP konseptissa ja ohjelmistojen kuvauksissa. Monesti tietokoneen sovellukset ovat huomattavasti monimutkaisempia ja suurempi kuvaustarkkuus onkin tarpeen. Osa logiikkavalmistajista on alkanut osassa asioita lähentymään tietokonemaailmaa, esimerkiksi Beckhoff:in viimeisin logiikkaympäristä Twincat 3 on tehty MS Visual Studion päälle, sekä tukee myös C++ ja Matlab ohjelmointia logiikkakäyttöön. Siemensillä on uusi pc pohjainen ohjelmoitava logiikka, jossa ajetaan rinnan logiikka käyttöjärjestelmää sekä pc käyttöjärjestelmää ja näiden sovellukset voivat keskustella toistensa kanssa.

7 Sovellussuunnittelu ja toteutus

Tölkkikuljettimien tehtävä on siirtää täytetyt tölkit täyttökoneen ulostulosta jälkipakkauskoneen sisääntuloon. Jälkipakkauskone pakkaa tölkit suurempiin yksiköihin kuten maitorullakot. Kuljettimien tulee siirtää tölkit mahdollisimman tehokkaasti. Tehokkaasti tarkoittaa tässä tilanteessa sitä, että kuljettimien tulee siirtää normaali olosuhteissa tölkit siten, ettei jonoa pääse kertymään. Epänormaaleissa tilanteissa taas tulee mahdollistaa uusien tölkkien saapuminen kuljettimille mahdollisimman aikaisessa vaiheessa jotta käytettävissä oleva koneaika tulee mahdollisimman hyvin käytettyä. Käytännössä siis esim. jälkipakkauksessa olleen häiriön vuoksi kun tölkkikuljetin linja täyttyy tölkeistä, tulee tölkkivirta hallita siten, ettei jononpaine kasva liian suureksi ja näin aiheuta uutta häiriötä häiriön poistumisen jälkeen, mutta kuitenkin siten, että uusia tölkkejä pystytään ottamaan mahdollisimman nopeasti häiriön poistumisen jälkeen.

7.1 Automaatiosuunnittelu

Tölkkikuljettimien ohjaussovellus ei ole niin monimutkainen eikä haastava kokonaisuus, että koko elinkaarimallin kaikkia vaiheita olisi ollut järkevää soveltaa käytäntöön siinä laajuudessa jossa elinkaarimalli ne esittää. Päävaiheet on kuitenkin aina syytä pitää samana ja vaiheiden poistamisen sijaan tuleekin harkita vaiheiden varsinainen sisältö.

Määrittelyvaihe oli hyvin lyhyt ja sisälsi käytännössä vain toimeksiannon sekä esisuunnittelun. Toisin kuin elinkaarimallissa mainitaan, että esi- ja perussuunnittelu tehtäisiin laitteistoriippumattomasti. Tein suunnittelun suoraan laitteistoriippuvaisena, koska suuri osa järjestelmästä oli jo olemassa sekä olimme jo päättäneet osan komponenteista. Komponenttivalintoihin vaikutti olemassa olevan laitekannan lisäksi varaosien saatavuus, tarve jatkossa pienentää varaosavaraston arvoa sekä tietenkin ylläpidon osaaminen. Varsinaisia käyttäjävaatimuksia järjestelmän yksinkertaisuuden takia ei tehty.

Suunnitteluvaihe oli hyvin lyhyt ja koostui lähinnä ohjelman osien määrittelystä ja toiminnan suunnittelusta.

Toteutusvaiheessa teimme sähkökuvat ja kytkentälistat tarvitulle tasolle, tilasimme tarvittavat osat kuten kuljettimien moottorit, valokennot ja väylämoduulit. Toteutusvaihe sisälsi myös varsinaisen sovelluksen ensimmäisen version suunnittelun.

Asennusvaihe oli hyvin mielenkiintoinen muuttuneiden tarpeiden takia. Sovellus oli kuitenkin onnistuttu osioimaan ja eristämään toiminta siten, että muutokset olivat vaivattomia tehdä.

Toiminnallinen testaus ja asennus vaihe venyi pitkäksi, koska yhden linjan kuljetin otettiin käyttöön ensin siten, että siltä pystyttiin ajamaan myös vanhoille kuljettimille. Tämä tehtiin siksi, että uusi jälkipakkauslaite oli ensimmäinen tyyppiään ja haluttiin ylläpitää tuotantokyky tilanteissa joissa uuteen jälkipakkauslaitteeseen tulee estävä vika tai virhetila.
Ensimmäinen käyttöönotettu jälkipakkauslaite otettiin käyttöön vielä ns. väärältä linjalta, eli ensimmäinen käyttö oli ns. ristiinajo jossa pakkauskoneen 1 puolelta ajettiin 2 puolen jälkipakkauslaiteeseen.
Seuraavassa vaiheessa otettiin käyttöön toinen jälkipakkauslaite ja pakkauskoneen toinen linja siirrettiin uusiin kuljettimiini, sekä siirrettiin kohtaa josta liityttiin vanhoihin kuljettimiin.
Viimeisessä vaiheessa, joka oli uusien jälkipakkauslaitteiden ylösajon ongelmien ja viivästyksien takia useita kuukausia myöhemmin kuin alkuperäinen suunnitelma, poistettiin liityntä vanhoihin kuljettimiin sekä päästiin tekemään viimeiset toiminnalliset muutokset ja asetuksien asetukset.

Kelpoistusvaihe on käytännössä hyvin yksinkertainen, koska järjestelmäkin on pieni. Asetukset tehtiin muutettavaksi paneelista, jolloin viimeinenkin hienosäätö pystyttiin tekemään mekaanisen asentajan toimesta.

Kuljettimet ovat nyt jatkuvassa tuotantokäytössä ja saatujen kokemusten perusteella on päätetty jatkaa samalla mallilla eteenpäin kattaen myös muuttuneet vaatimukset. Jatkossa siis tulee asennus ja testausvaiheeseen käytännössä samat muuttuvat vaatimukset joissa jälkipakkauslaitteet otetaan vaiheittain käyttöön ja liityntä vanhoihin kuljettimiin ja siten mahdollisuus ajaa myös vanhoihin jälkipakkauslaitteisiin säilytetään.

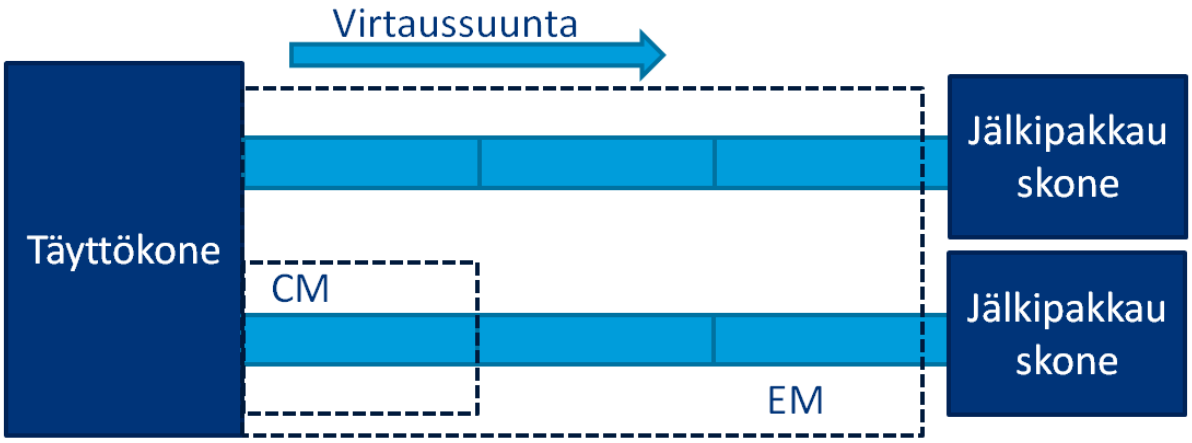
7.2 Sovellus

Sovelluksen perustana on aiemmin esiteltyjen standardien oma soveltaminen työn kohteeseen. Käytännössä standardit on tehty suurempia kokonaisuuksia ajatellen ja siitä syystä usein huomattavan raskaita toteuttaa eikä niiden suoranaista käyttöä voida perustella pieniin kohteisiin. Onkin tärkeää pyrkiä soveltamaan juuri olennaiset osat standardien ajatusmaailmasta kuhunkin kohteeseen, mitä suurempi kokonaisuus sitä syvemmin standardi on sovellettavissa käyttöön.

Sovelluksessa on myös pyritty minimoimaan ulospäin näkyvien muuttujien määrä, tästä käytettäisiin pc ohjelmointi puolella kapselointi käsitettä. Eli muuttuja pyritään pitämään näkyvissä vain oleelliseen osaan ohjelmaa ja vain oleellinen tieto siirtämään muun sovelluksen käyttöön. Paras esimerkki tästä on toteutettu yksittäisen kuljettimen ohjausmoduuli.

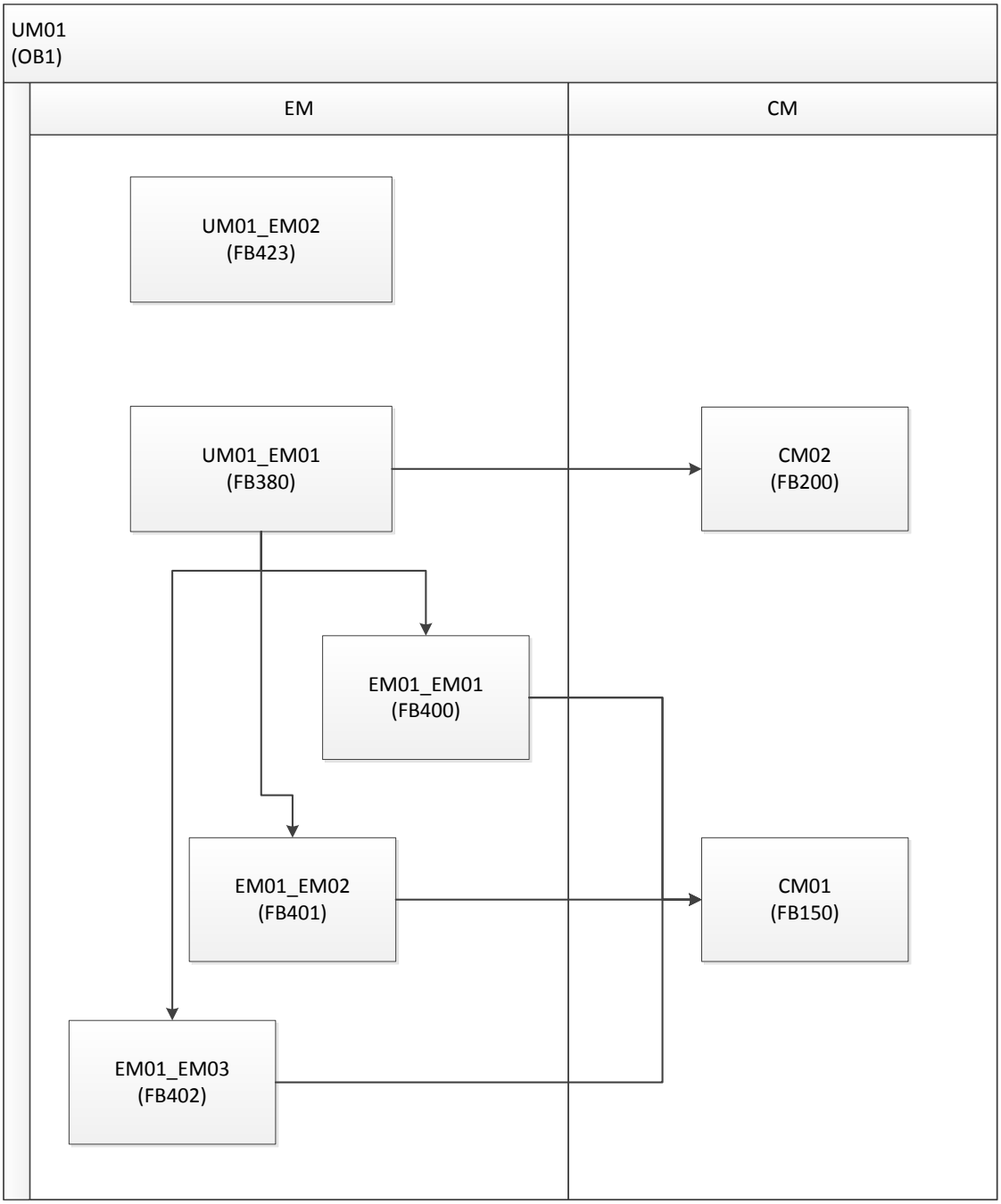
Eri ajovaihtoehtojen toteutus on pyritty tekemään siten, että se pysyy mahdollisimman ylläpidettävänä. Tätä ei ajan käytöllisistä syistä pystytty aivan loppuun asti pohtimaan, mutta ainakin modulaarisuus ja kapselointi on pyritty toteuttamaan.

Sovelluksen perustoiminta on kahdessa moodissa, automaatti ja manuaali. Automaattimoodissa on lisäksi kaksi eri vaihetta, tuotanto sekä pesu. Tähän en toteuttanut modemanageria jonka PackML/ISA88 TR2 esittää, koska laitteisto on hyvin yksinkertainen. Normaalisti järjestelmä pidetään aina automaatilla.



Kuvio 10. Moduulijaon periaatekuva

Kuviosta 10 nähdään miten moduulit liittyvät fyysiseen systeemiin. Yksittäisen kuljettimen ohjaukseen on oma control module (CM) ja kokonaisuus on oman equipment modulen (EM) sisällä.



Kuvio 11. Sovelluksen lohkojen periaate kutsuhierarkia ja lohkotyypit

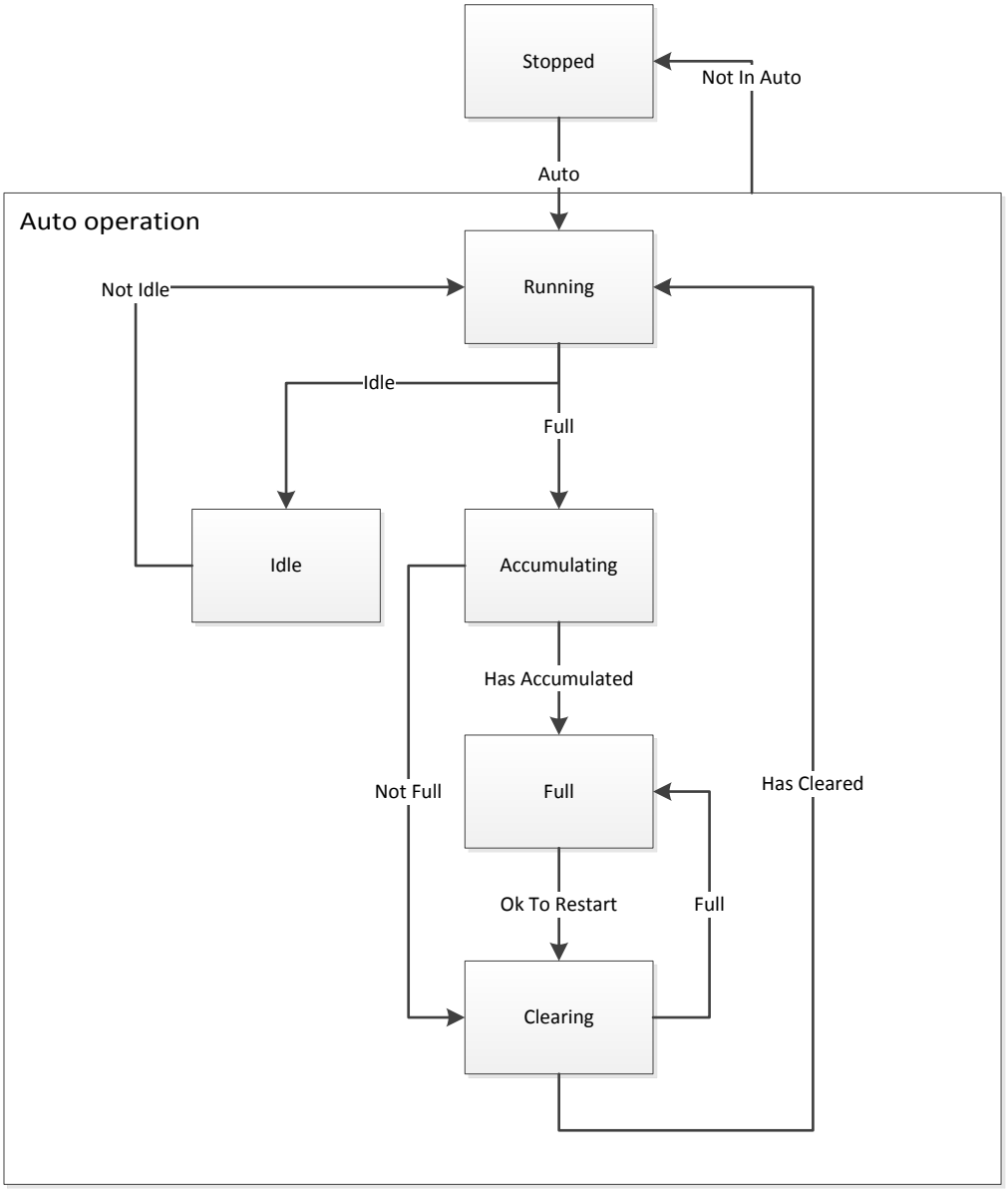
Tarkemmin jaoteltuna moduulit on jaoteltu pääpiirteittäin siten, että on kuljetin laitteisto (equipment module, EM01) tämä sisältää kuljettimia sekä kuljettimien ohjausmoduuleita (control module, CM01) kuvion 11 esittämällä tavalla. Kuljettimien EM sisältää lisäksi saman alueen kattavia reitin valinnan

toteuttavia moduuleita (EM01_EM0x). Näiden lisäksi on pienempiä ohjausmoduuleita, joista osa on sovelluksen sisään rajautuvia ja osa suoraan laitteeseen liittyviä kuten CM02, joka on yksittäisen moottorin ohjauksen toteuttava moduuli.

Sovelluksen perusrakenteessa on pyritty jakamaan toimintalogiikka siten, että rajapinna olisivat mahdollisimman minimaalisia sekä logiikka mahdollisimman eristetty ja riippumaton ympäröivästä ohjelmistosta. Tärkein lohko on yksittäisen kuljettimen ohjauslohko, tämä ohjauslohko siis toteuttaa vain ja ainoastaan automaattitilan ohjauslogiikan, eikä ole suoraan kytketty varsinaiseen moottoria ohjaavaan lähtöön. Lisäksi on eräänlaiset reitin määrittävät lohkot sekä muutamia apulohkoja kuten automaatti ja manuaali ajotilan erottavat moottorin ohjauslohko.

7.2.1 Kuljetinmoduulin ohjelmointi

Suunnittelemalla ensin varsinainen tilakone ja sen jälkeen keskittymällä tilan vaihdon tarpeisiin, pystytään ratkaistavaa ongelmaa pitämään mahdollisimman pienenä. Laadukkaan sovelluksen suunnittelussa pitäisikin esisijaisesti pyrkiä pitämään kerralla ratkaistavat ongelmat mahdollisimman pieninä, tämä mahdollistaa mahdollisimman virheettömän sovelluksen tekemisen sekä helpottaa ottamaan kaiken juuri käsiteltävänä olevan ongelman kannalta tärkeän huomioon. Näistä sitten rakentuu suurempi kokonaisuus. Aina tason noustessa tulisi suunnittelun tapahtua sopivalla abstraktiotasolla, jolloin ihmisen käsittelykyky pysyy huomattavasti paremmin mukana ja virheiden mahdollisuus pienenee.



Kuvio 12. Kuljetinmoduulin tilakone

Kuljetinmoduulin suunnittelu lähti siten liikkeelle, että kuvattiin miten kuljettimen tulisi toimia kuvion 12 esittämällä tavalla. Ensin kuvattiin normaali toiminta, joka siis koostuu vain kolmesta tilasta ”Stopped”, ”Running” ja ”Idle”. Tästä laajennettiin ns. epänormaaleihin tiloihin, eli tiloihin jotka tulevat esiin kun jälkipakkauslaite ei pysty ottamaan tuotetta vastaan esimerkiksi häiriön takia. Alun perin suunnitelmassa ei ollut siirtymää ”Clearing” tilasta ”Full” Tilaan takaisin, tämä aiheutti

tehottomuutta tapauksissa joissa jälkipakkauslaite pysähtyi useasti peräkkäin ilman, että kuljettimet olivat ehtineet tyhjentymään tarpeeksi joten se lisättiin.

Ohjelmiston olennaisin osa, kuljetinmoduulin ohjauslohko on toteutettu yksinkertaisella tilakoneella. Ohjelmointikieli on valittu ylläpidettävyyden takia. Graph7 olisi ollut hyvin raskas kokoon nähden, muut kielet paitsi SCL taas olisivat tuottaneet hyvin vaikeaselkoisen lopputuloksen joten SCL on paras vaihtoehto ohjauslogiikan toteuttamiseen.

```
Running: //RUNNING
IF IsIdle THEN
  NewState := Idle;
END_IF;
IF IsFull THEN
  NewState := Accumulating;
END_IF;
IF NOT Auto THEN
  NewState := Stopped;
END_IF;
```

Kuvio 13. Esimerkki signaalin käytöstä tilakoneen ohjaukseen

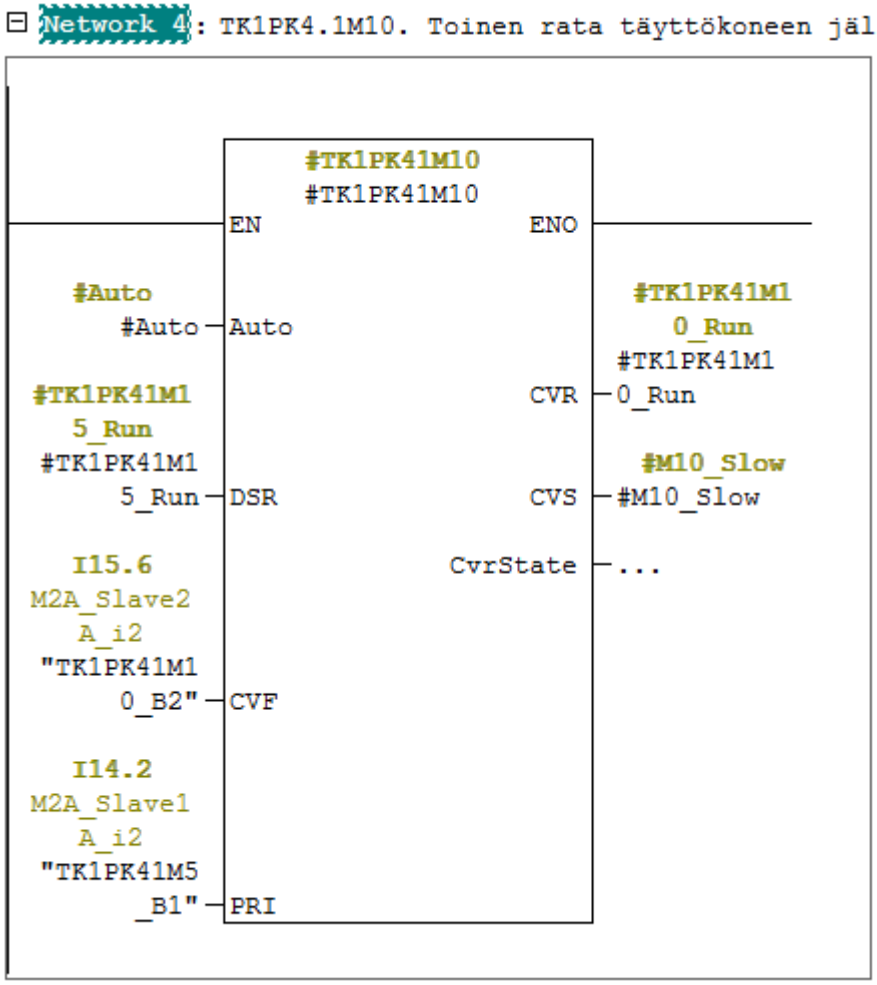
Perustoiminnallisuus on toteutettu tilakoneena käyttäen CASE kontrollilausetta. Yksittäisille tiloille on annettu vakiot (constant) jolloin CASE lausekkeessa tilaa voidaan ilmaista selkokielisesti numeron sijaan kuvion 13 esittämällä tavalla. Yksittäiset tilat on pyritty pitämään mahdollisimman kompakteina ja yksinkertaisina.

```
IdleTimer(IN :=NOT PRI AND NOT CVF, PT:=Settings.IdleTime);
IsIdle:=IdleTimer.Q;
```

Kuvio 14. Esimerkki signaalin luonnista, tyhjäkäynti signaali

Itse signaalit muodostetaan omassa osiossaan ja tilan käsittelyyn liittyy pelkästään signaalien tuoma tieto. Tämä on myös hyvä esimerkki ongelman rajaamisesta mahdollisimman pieneksi. Kuvion 14 esimerkki näyttää vastauksen kysymykseen ”onko kuljetin toimeton”. Vastaus on ”kun

kuljettimelle ei ole tullut tuotetta, eikä kuljetin ole täysi, kuljetin on toimeton kun haluttu aika on kulunut”.



Kuvio 15. Kuljettimen ohjauslogiikan kutsu. Control Modulen kutsu

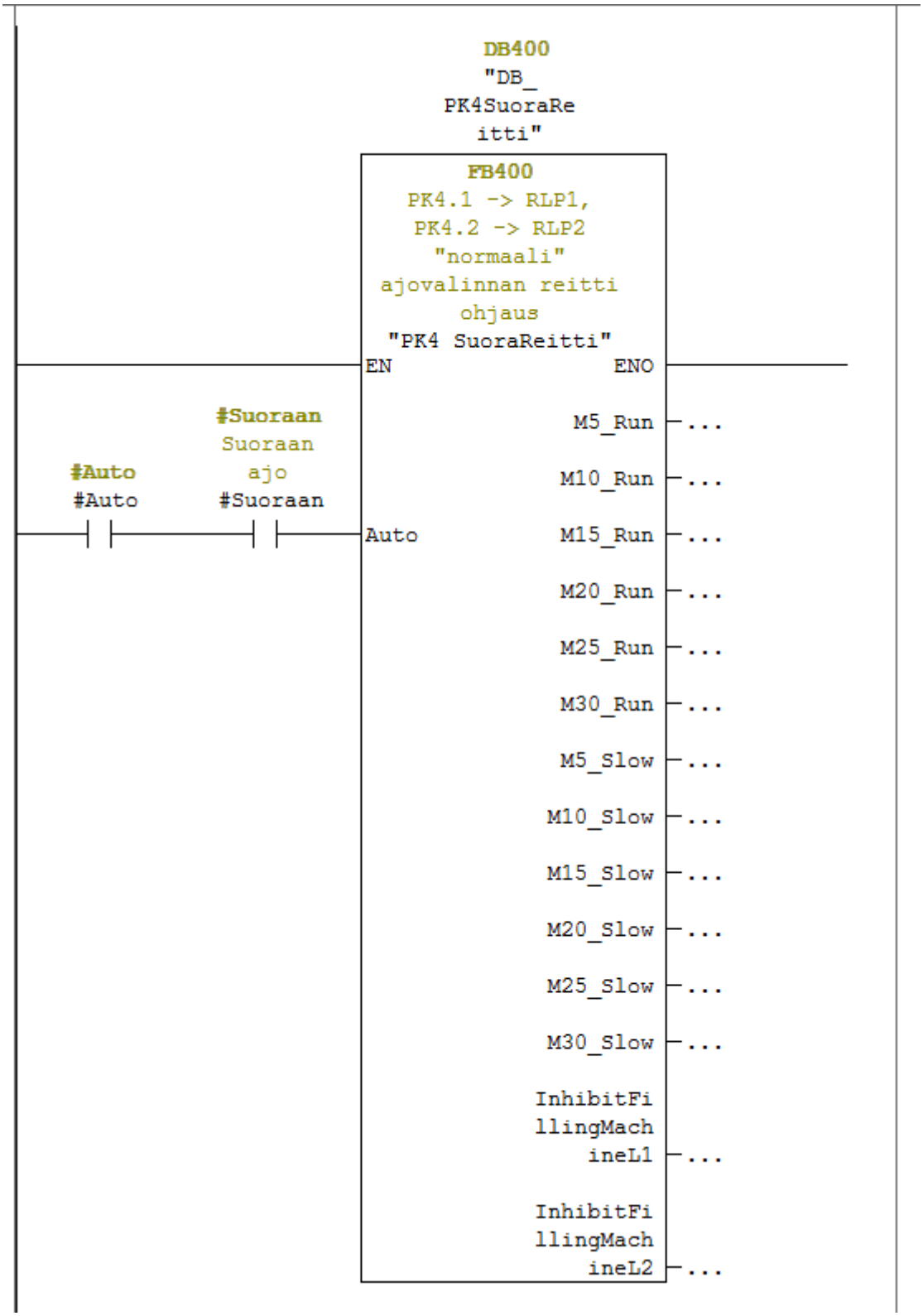
Kuljetinmoduulin ohjauslogiikka ulospäin paljastaa vain tilansa sekä onko kyseisen kuljettimen tarkoitus pyöriä nopeasti, hiljaa vai olla seis kuvion 15 esittämällä tavalla. Kuvion 15 CVR (conveyor run) on käynti pyynti, CVS (conveyor slow speed) on hitaastipyörimis pyynti, esimerkiksi käytössä kun ruuhkaa puretaan, Auto on automaatti päälle, DSR (downstream run) on kuljettimen jälkeen olevan kuljettimen käyntitieto, CVF (conveyor full) on kuljettimen täyteen tulon ilmaisevan valokennon signaali ja PRI (product

incoming) on kuljettimelle tulevan materiaalin ilmaisevan valokennon tai pakkauskoneelta tuleva signaali.

7.2.2 Reitin ohjaus

Suuri ongelmakohta vanhoissa kuljetinjärjestelmissä on ollut eri reitti valintojen tuoma kompleksisuus ohjauksiin. Yleensä toteutukset ovat olleet hyvin lähtökeskeisiä. Kuljettimen moottorin ohjauksen lähes kaikki ohjaus on toteutettu muutamaan ladder networkiin. Tähän kun lisää eri reittivalinnat ja kiinteät vaihtumattomat viiveet, jotka on pyritty sovittamaan kaikkiin ajovalintoihin sopivaksi. On toiminta huono kompromissi eri reittien tarpeiden kesken.

Tämän työn tarkoitus oli siis myös pyrkiä ratkaisemaan osaltaan tätä ongelmaa ja löytää ratkaisu jolla eri reittien ohjaus voitiin erottaa toisistaan.



Kuvio 16. Reitin kutsu (”resepti”)

Sovelluksessa eriytettiin omiin lohkoihinsa jokainen mahdollinen resepti eli ajovalinta kuvion 16 esittämällä tavalla. Tällä saatiin erotettua kaikki reitinvalintaan liittyvät asetukset sekä toiminnot erilleen. Kuitenkin alla toimii sama kuljetinlohko, jota muuttamalla perustoiminnallisuus muuttuu. Tämä mahdollistaa vaivattoman ylläpidon.

8 Pohdinta

Tässä työssä pääsin tekemään sovellussuunnittelua hyvin tuntemaani prossessin osaan. Vuosien kokemuksen perusteella minulla oli jo selvä kuva siitä miten tölkkikuljettimien ohjauksen tulisi toimia. Minua on aina kiinnostanut erilaiset automaatiojärjestelmät sekä erityisesti miten saada mahdollisimman helposti ylläpidettävä lopputulos, tähän varmasti on vaikuttanut kunnossapidon taustani ja selvä kuva siitä, kuinka kallista sekavien ja laadullisesti huonojen järjestelmien ylläpito on.

Minulle oli käytettävä Simatic Manager ohjelmointiympäristö sekä käytettävät tekniikat muutoin on hyvin tuttua. Näiden kanssa siis ei ollut mitään ongelmia. Radaston moottorit ja anturit ovat ASi väylässä, joka on toteutettu Pepperl Fuchsin ASi tupla masterilla, joka on kytketty Siemens S7-300 logiikkaan Profibus DP väylällä. Ohjauspaneeli on kytketty ethernetillä.

Oma vahvuuteni on aina ollut ongelmien ratkaisussa ja saan juuri siitä parhaan tyydytyksen työssäni. Tässä työssä pääsinkin hyvin ruokkimaan myös sitä puolta. Asetettu ongelma oli sopivasti rajautuva uuden soveltamiseen käytäntöön järkevällä työmäärällä.

Opinnäytetyöni aiheen valintaa pidän onnistuneena, sillä pääsin hyvin syventämään osaamistani jo vahvalla pohjalla olevan aihealueen suhteen. Työn tein käytännössä itsenäisesti, kuitenkin keskustellen kollegoideni kanssa eri ratkaisuista työn edetessä. Suuri hyöty oli tietenkin siitä, että ympäristö on vuosien saatossa tullut hyvin tutuksi.

Ohjelmiston käyttöönotto sujui todella hyvin ja modulaarisen rakenteen vuoksi oli jatkuvasti muuttuviin vaatimuksiin helppo mukautua. Alkujaan vaatimus oli suora linja pakkauskoneen pakkauslinjalta jälkipakkauslinjaan ilman mitään ristiinajo tai muihin radastoihin ajo vaatimuksia. Projektin edetessä vaatimukset kehittyivät siten, että ensin tuli vaatimus ristiinajoon ja myöhemmin lähempänä käyttöönottoa tuli vielä vaatimus väliaikaisesta mahdollisuudesta ajaa vanhaan tölkkikuljetinosaan vanhojen jälkipakkauslaitteiden käytön mahdollistamiseksi. Asetusten eli pääsääntöisesti eri tilojen vaihtumiseen liittyvien aikojen lopullisen asettamisen suoritti mekaanisen puolen asentaja ja olenkin tyytyväinen myös saavutettuun käytettävyyteen.

Uudella ohjelmoinnin ajatusmallilla säästettiin noin 70% antureista, vältettiin ratapainejarrujen käyttö ja tietyissä tilanteissa tapahtuva tölkin pohjan rikkoontuminen saatiin loppumaan.

Jatkossa kun pakkauslaitteiston ja siten tölkkikuljettimien uudistus etenee pakkaussalin osalta, tulen kuitenkin kehittämään vielä yhden version ohjelmasta ratkaistakseni asioita jotka jäivät vielä liian pienelle huomiolle. Nyt kiinnitin suurimman huomion varsinaisen radan toimintalogiikan tekoon ja hyväksi saamiseen sekä osioiden eristämiseen toisistaan siten, että rajapinnat

olisivat pieniä. Rajapintoja on kuitenkin mielestäni vielä mahdollista minimoida ja selkiyttää siten, että lopputulos on vielä yksinkertaisempi ja helpompi ylläpitää. Lisäksi yksinkertaisempi kuljetinlinjan järjestyksen konfigurointi on asia johon tulen vielä kiinnittämään huomiota.

Lähteet

Automaatiosovellusten ohjelmistokehitys. 2005. Suomen Automaatioseura ry.

Laatu automaatiossa - parhaat käytännöt. 2001. Suomen Automaatioseura ry.

IEC 61131-3:2013 Programmable controllers – Part3: Programming languages

ISA88.01, Batch Control, Part 1: Models and Terminology. The Instrumentation, Systems and Automation Society 2010.

ISA-TR88.00.02, Machine and Unit States: An Implementation Example of ISA-88. The Instrumentation, Systems and Automation Society 2008.

ISA. Kotisivut 2015. Viitattu 4.4.2015. http://www.isa.org/

OMAC. Kotisivut 2015. Viitattu 20.4.2015 http://www.omac.org/

Valio Oy. Kotisivut 2015. Viitattu 12.3.2015. http://www.valio.fi/

Ylén J-P, Ventä O, Tommila T, Lappalainen J, Hirvonen J, Karhela T, Paljakka M, Lehtinen H, Heilala J, Peltonen J, Malm T, Valkonen J, Voho P. Automaatio liiketoimintaprosessien tukena. Tekesin katsaus. TEKES 2010.

Liitteet

Liite 1. Tölkkikuljetinten sovellus

```
1  FUNCTION_BLOCK "PackageConveyor"
2
3
4     VAR_INPUT
5        Auto : BOOL;
6        DSR : BOOL; //DOWNSTREAM RUNNING, Conveyor after or group packagin machine
7        CVF : BOOL; //CONVEYOR FULL SIGNAL, Signal indication for this device that its full
8        PRI : BOOL; //PRODUCT INCOMING, Signal indication for this device that product is incomi
   ng (Photocell or filling machine)
9     END_VAR
10
11    VAR_OUTPUT
12       CVR : BOOL; //CONVEYOR RUNNING, Signal out, this device is running
13       CVS : BOOL; //CONVEYOR SLOW SPEED, Signal out, this device on slow speed
14       CvrState : INT;
15    END_VAR
16
17    VAR
18       State : Int;
19       OldState : Int;
20       NewState : Int;
21       FullTimer : TON;
22       IdleTimer : TON;
23       AccumulationTimer : TON;
24       ClearingTimer : TON;
25       RestartTimer : TON;
26       Settings : STRUCT
27          FullTime : TIME := T#20s;
28          IdleTime : Time := T#20s;
29          AccumulationTime : Time := T#20s;
30          ClearingTime : Time := T#20s;
31          RestartDelay : Time := t#20s;
32       END_STRUCT;
33       IsFull : Bool;
34       IsIdle : Bool;
35       HasAccumulated : Bool;
36       HasCleared : Bool;
37       OkToRestart : BOOL;
38    END_VAR
39
40    CONST
41     Stopped := 0;
42     Running := 1;
43     Idle    := 2;
44     Accumulating := 3;
45     Full    := 4;
46     Clearing := 5;
47    END_CONST
48
49 BEGIN
50     (*
51     Program module for packagin conveyor 'device'.
52
53     DSR : DOWNSTREAM RUNNING //Conveyor after or group packagin machine
54     CVR : CONVEYOR RUNNING //Signal out, this device is running
55     CVF : CONVEYOR FULL SIGNAL //Signal indication for this device that its full
56     PRI : PRODUCT INCOMING //Signal indication for this device that product is incoming
57
58     AUTO : AUTO MODE //Sets this device to automatic operation, 'on'
59
60     *)
61
62     (* Main State Handler*)
63
64     CASE State OF
65       Stopped: //STOPPED
66       IF Auto THEN
67         NewState := Running;
68       END_IF;
69       (*
70       DEVICE IS NOT IN AUTO MODE
71
```

```
72        ->RUNNING, IF AUTO MODE
73        *)
74
75        Running: //RUNNING
76        IF IsIdle THEN
77          NewState := Idle;
78        END_IF;
79        IF IsFull THEN
80          NewState := Accumulating;
81        END_IF;
82        IF NOT Auto THEN
83          NewState := Stopped;
84        END_IF;
85
86        (*
87
88        TRANSITIONS
89        ->IDLE, IF EMPTY FOR TIME
90        ->ACCUMULATIN, IF FULL FOR TIME
91        ->STOPPED, IF AUTO MODE REMOVED
92        *)
93
94        Idle: //IDLE
95        IF NOT IsIdle THEN
96          NewState := Running;
97        END_IF;
98        IF NOT Auto THEN
99          NewState := Stopped;
100       END_IF;
101
102       (*
103       TRANSITIONS
104       ->RUNNING, IF PRODUCT ARRIVING
105       ->STOPPED, IF AUTO MODE REMOVED
106       *)
107
108       Accumulating: //ACCUMULATING
109       IF HasAccumulated THEN
110         NewState := Full;
111       END_IF;
112       IF NOT IsFull THEN
113         NewState := Clearing;
114       END_IF;
115       IF NOT Auto THEN
116         NewState := Stopped;
117       END_IF;
118       (*
119        TRANSITIONS
120       ->FULL, IF FULL FOR TIME
121       ->CLEARING, IF NOT FULL
122       ->STOPPED, IF AUTO MODE REMOVED
123       *)
124
125       Full: //FULL
126       IF OkToRestart THEN
127         NewState := Clearing;
128       END_IF;
129
130       IF NOT Auto THEN
131         NewState := Stopped;
132       END_IF;
133       (*
134        TRANSITIONS
135       ->CLEARING, IF DSR
136       ->STOPPED, IF AUTO MODE REMOVED
137       *)
138
139       Clearing: //CLEARING
140       IF HasCleared THEN
141         NewState := Running;
142     END_IF;
143     IF IsFull THEN
```

```
144         NewState := Full;
145         END_IF;
146       IF NOT Auto THEN
147         NewState := Stopped;
148       END_IF;
149       (*
150       TRANSITIONS
151       ->RUNNING, IF NOT FULL AND TIME
152       ->STOPPED, IF AUTO MODE REMOVED
153       *)
154
155     END_CASE;
156
157
158
159     FullTimer(IN :=CVF AND (State = NewState OR NewState = Full OR NewState = Accumulating), P
    T:=Settings.FullTime); //If conveyor full for period of time, set full signal
160     IsFull := FullTimer.Q;
161
162     IdleTimer(IN :=NOT PRI AND NOT CVF, PT:=Settings.IdleTime); //If product not incoming and
    is not full, set idle signal
163     IsIdle:=IdleTimer.Q;
164
165     AccumulationTimer(IN := State = Accumulating AND NOT DSR, PT:=Settings.AccumulationTime);
166     HasAccumulated:=AccumulationTimer.Q;
167
168     ClearingTimer(IN := State = Clearing  AND NOT CVF, PT:=Settings.ClearingTime);
169     HasCleared:=ClearingTimer.Q;
170
171     RestartTimer(IN:= State = Full AND DSR, PT:=Settings.RestartDelay);
172     OkToRestart:=RestartTimer.Q;
173
174     //Is running?
175     CVR := (State = Running) OR (State = Accumulating) OR (State = Clearing);
176     CVS := (State = Accumulating) OR (State = Clearing);
177
178
179     OldState := State;
180     CvrState := State;
181     State := NewState;
182
183 END_FUNCTION_BLOCK
184
185
```

FB380 - <offline>

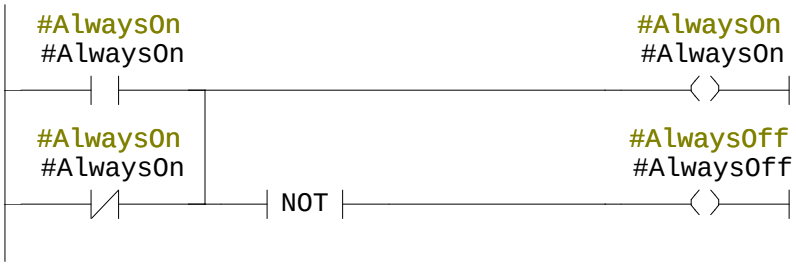
"PK4 Radat" PK4 koneen ratojen reitit ja auto/manu ohjaukset

Name: **Family:**
Author: Muotio **Version:** 0.1
Block version: 2
Time stamp Code: 10/30/2014 08:13:24 AM
Interface: 10/30/2014 08:01:45 AM
Lengths (block/logic/data): 02346 02030 00010

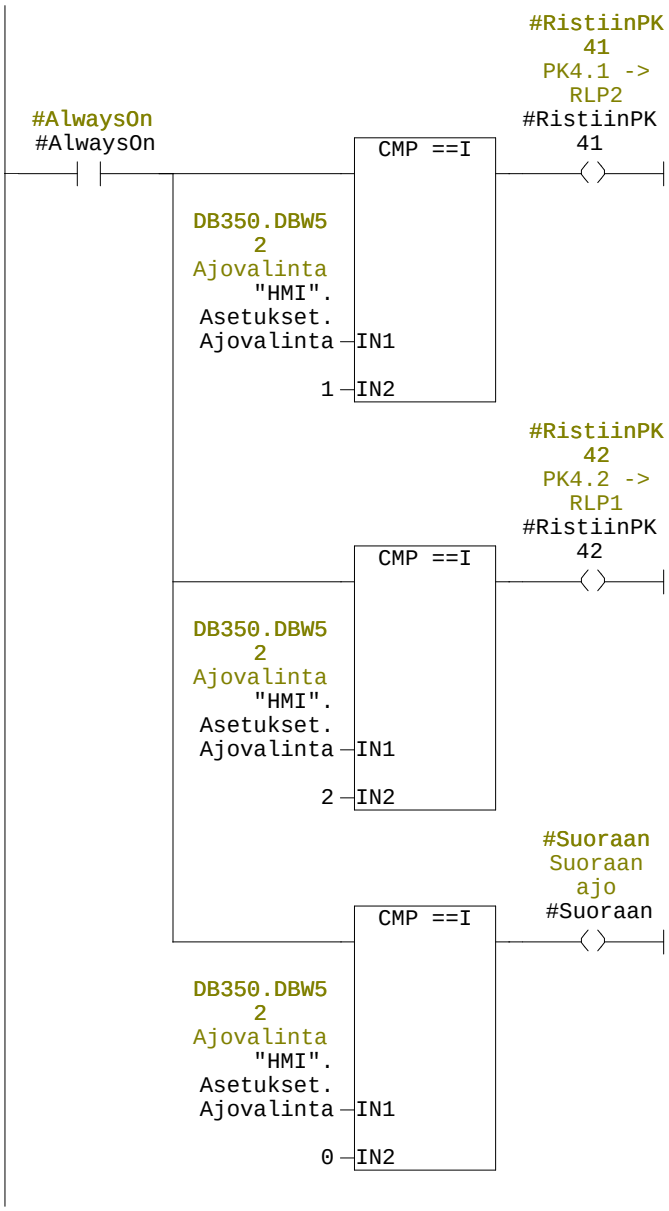
Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Auto	Bool	0.0	FALSE	
OUT		0.0		
IN_OUT		0.0		
STAT		0.0		
Suoraan	Bool	2.0	FALSE	Suoraan ajo
RistiinPK41	Bool	2.1	FALSE	PK4.1 -> RLP2
RistiinPK42	Bool	2.2	FALSE	PK4.2 -> RLP1
M5	SEW_DRC_ASi	4.0		
M10	SEW_DRC_ASi	10.0		
M15	SEW_DRC_ASi	16.0		
M20	SEW_DRC_ASi	22.0		
M25	SEW_DRC_ASi	28.0		
M30	SEW_DRC_ASi	34.0		
PesuPyoritysaika	TON	40.0		
PesupyoritysReuna	Bool	62.0	FALSE	
PesupyoritysPaalla	Bool	62.1	FALSE	
TEMP		0.0		
AlwaysOn	Bool	0.0		
AlwaysOff	Bool	0.1		
Dummy	Bool	0.2		

Block: FB380 PK4 Ratakentän ohjaukset

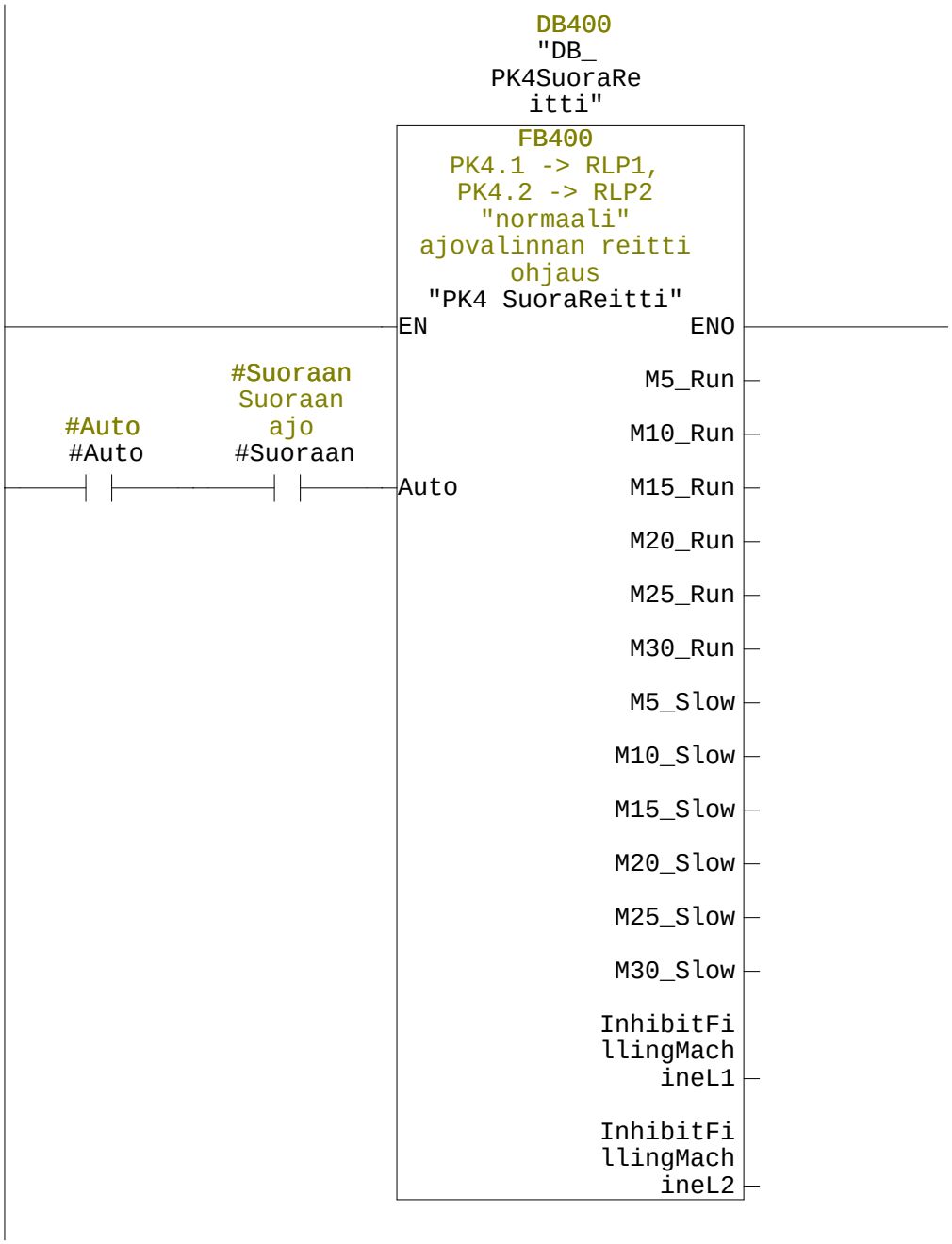
Network: 1 AlwaysOn, AlwaysOff



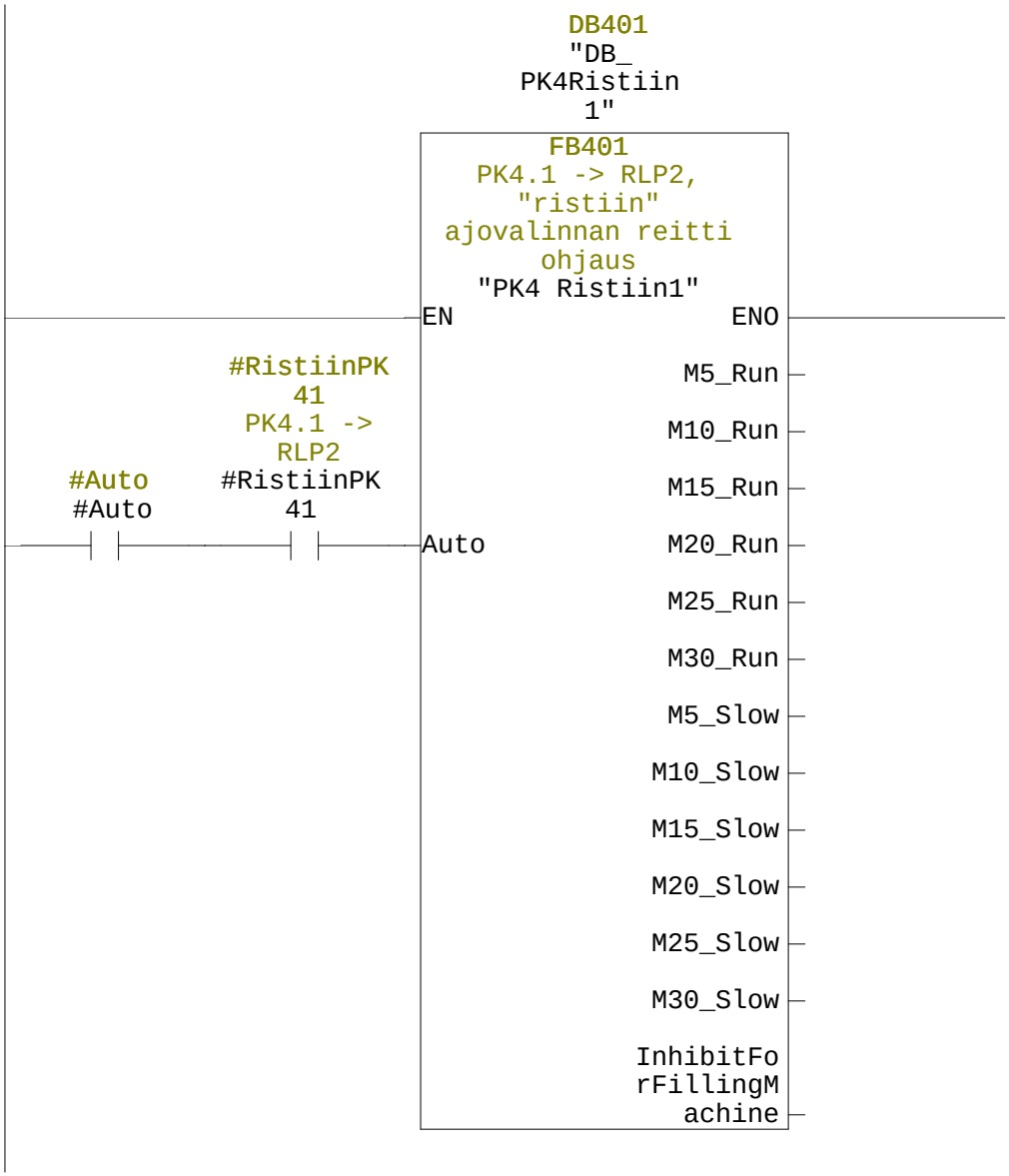
Network: 2 Ajovalinta



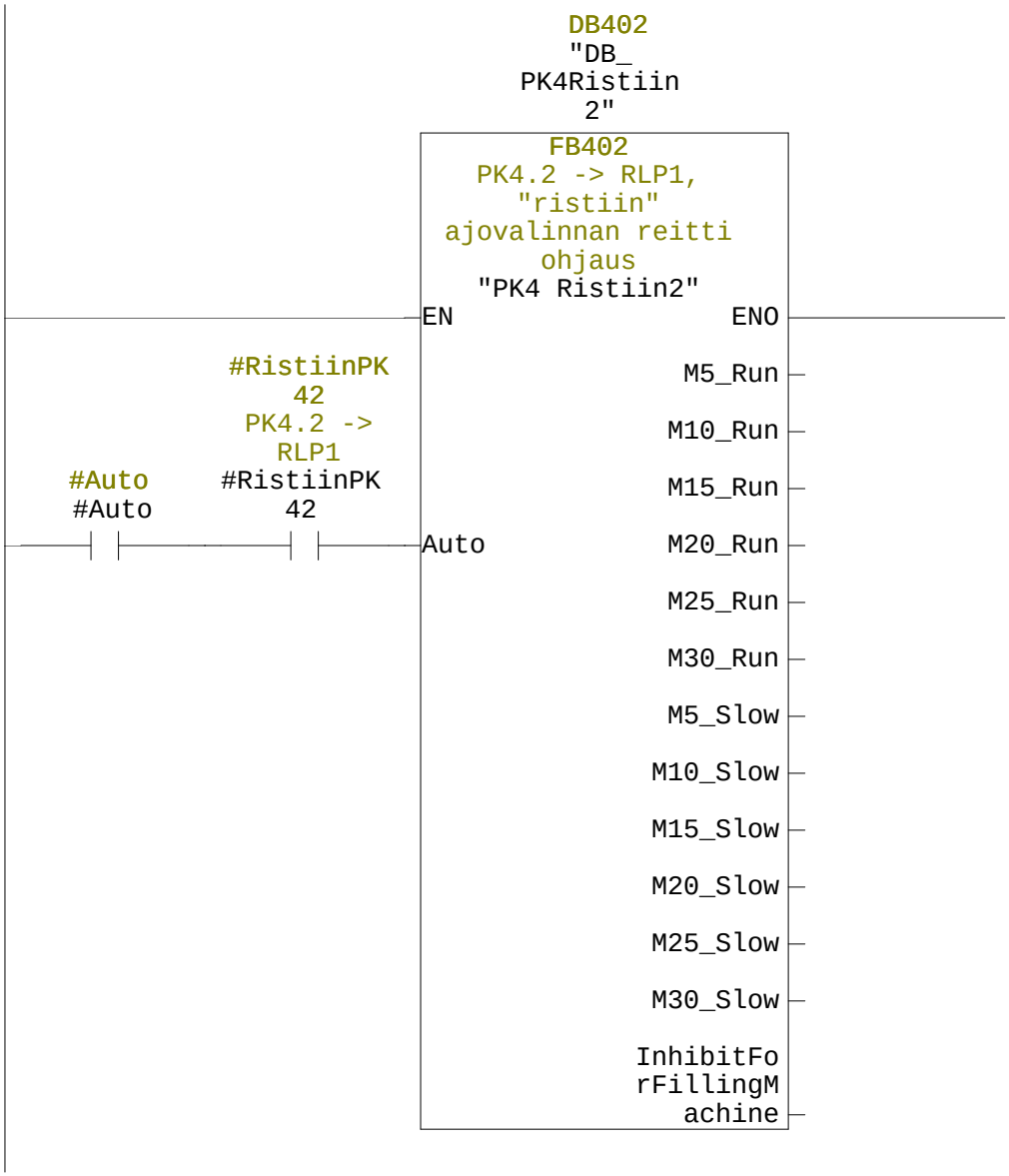
Network: 3 Suorareittiohjaus, PK4.1 -> RLP1 ja PK4.2 -> RLP2



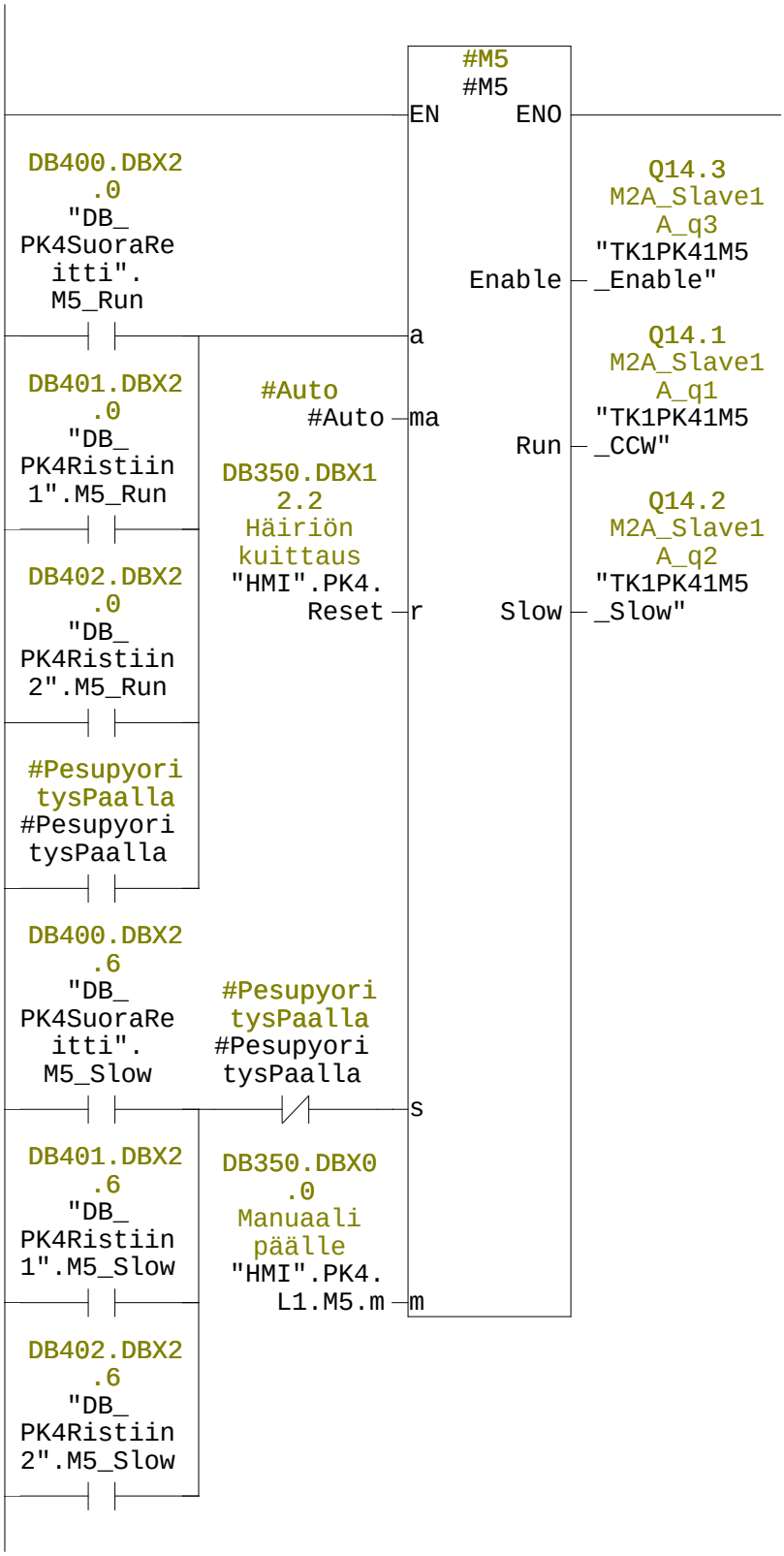
Network: 4 Ristiinreittiohjaus, PK4.1 -> RLP2



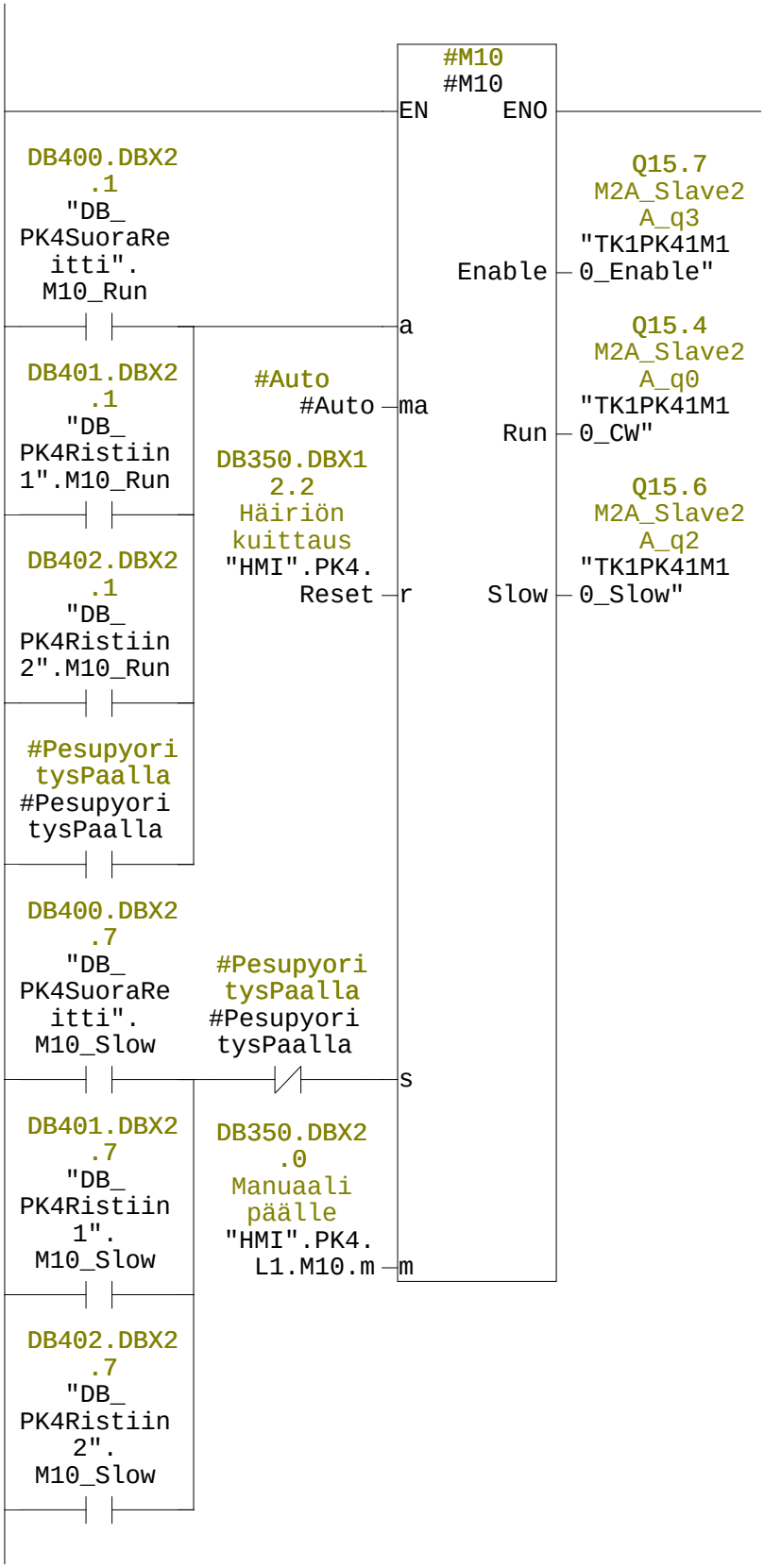
Network: 5 Ristiinreittiohjaus, PK4.2 -> RLP1



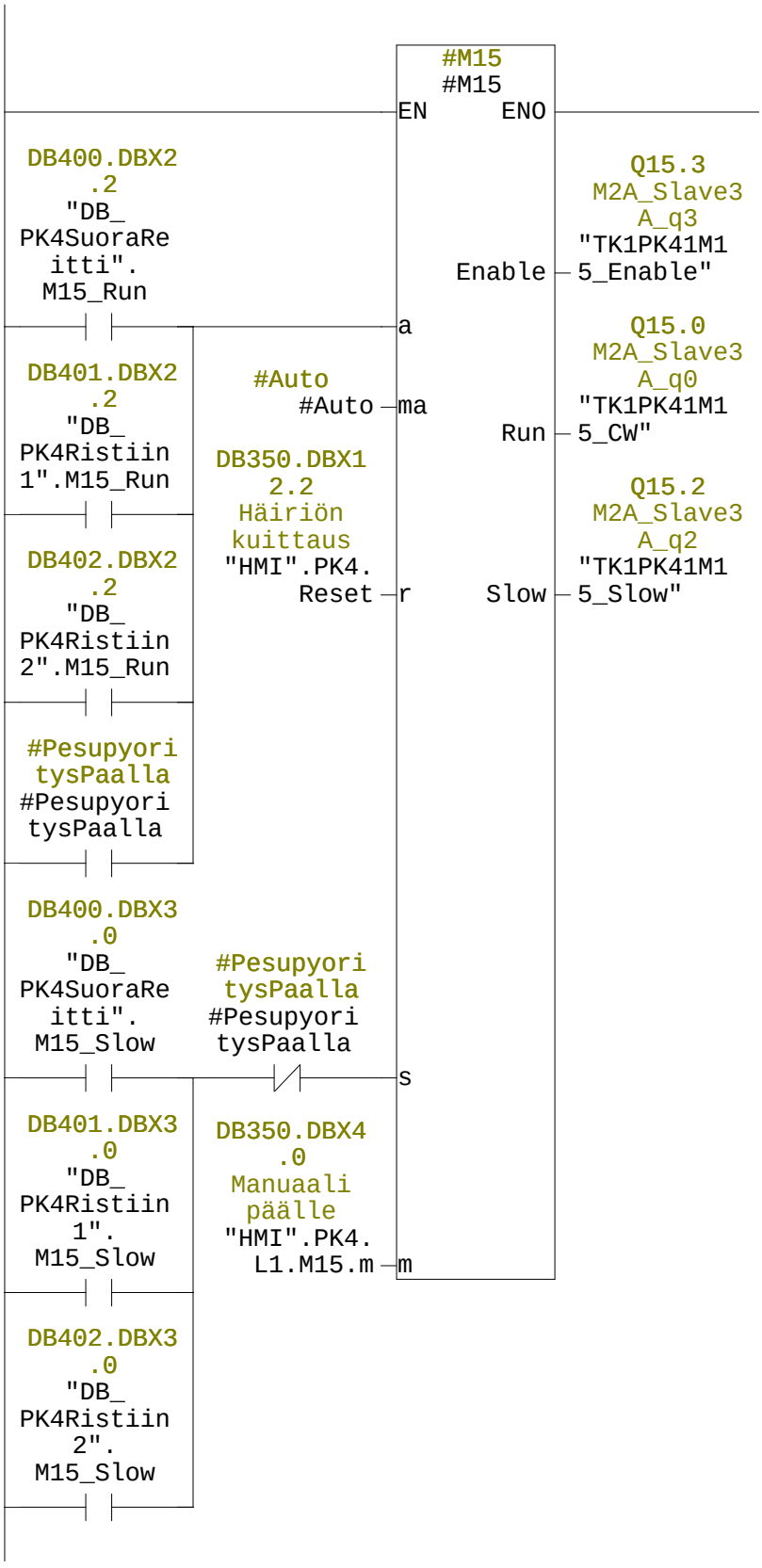
Network: 6 TK1PK4.1M5



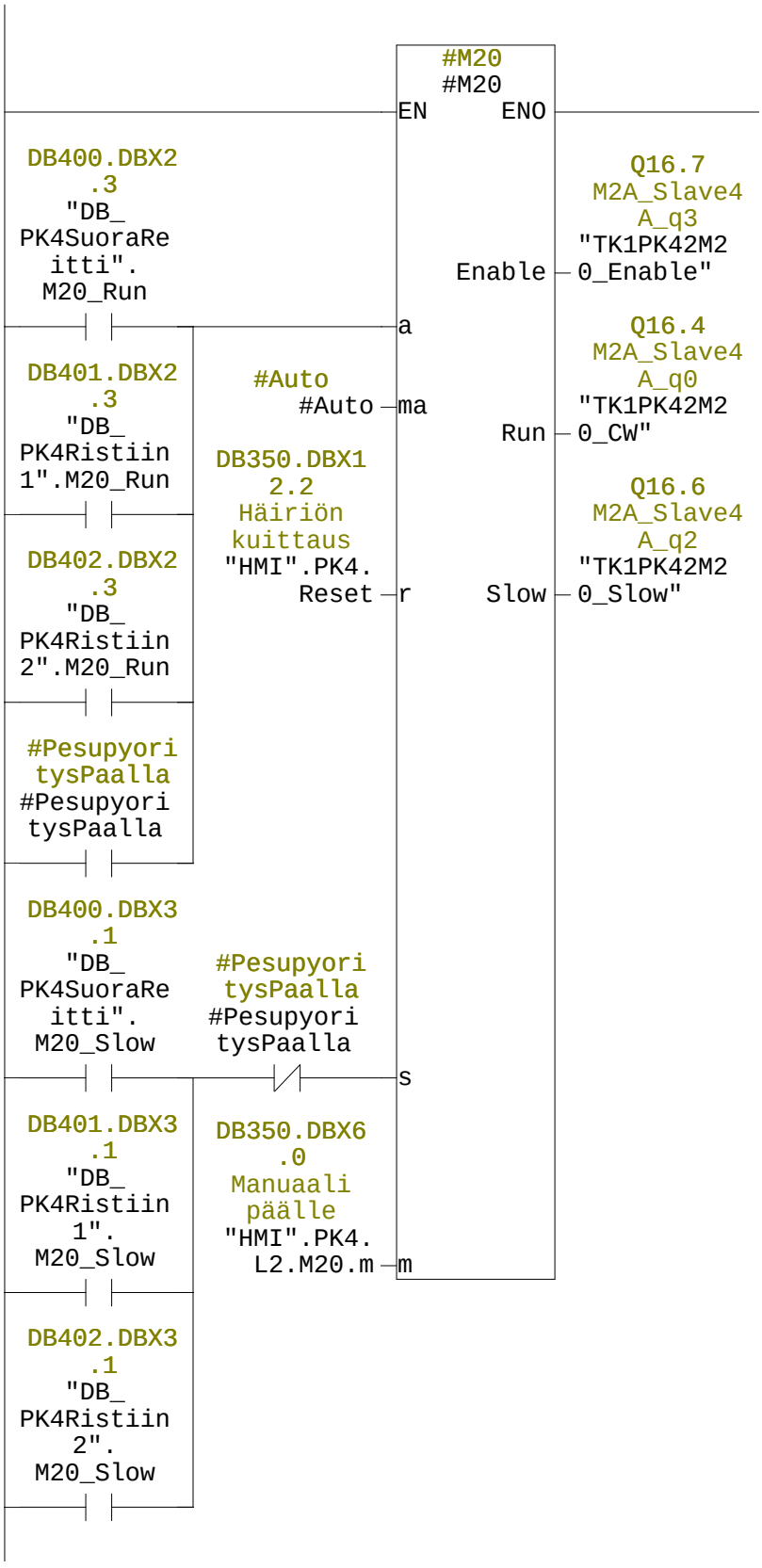
Network: 7 TK1PK4.1M10



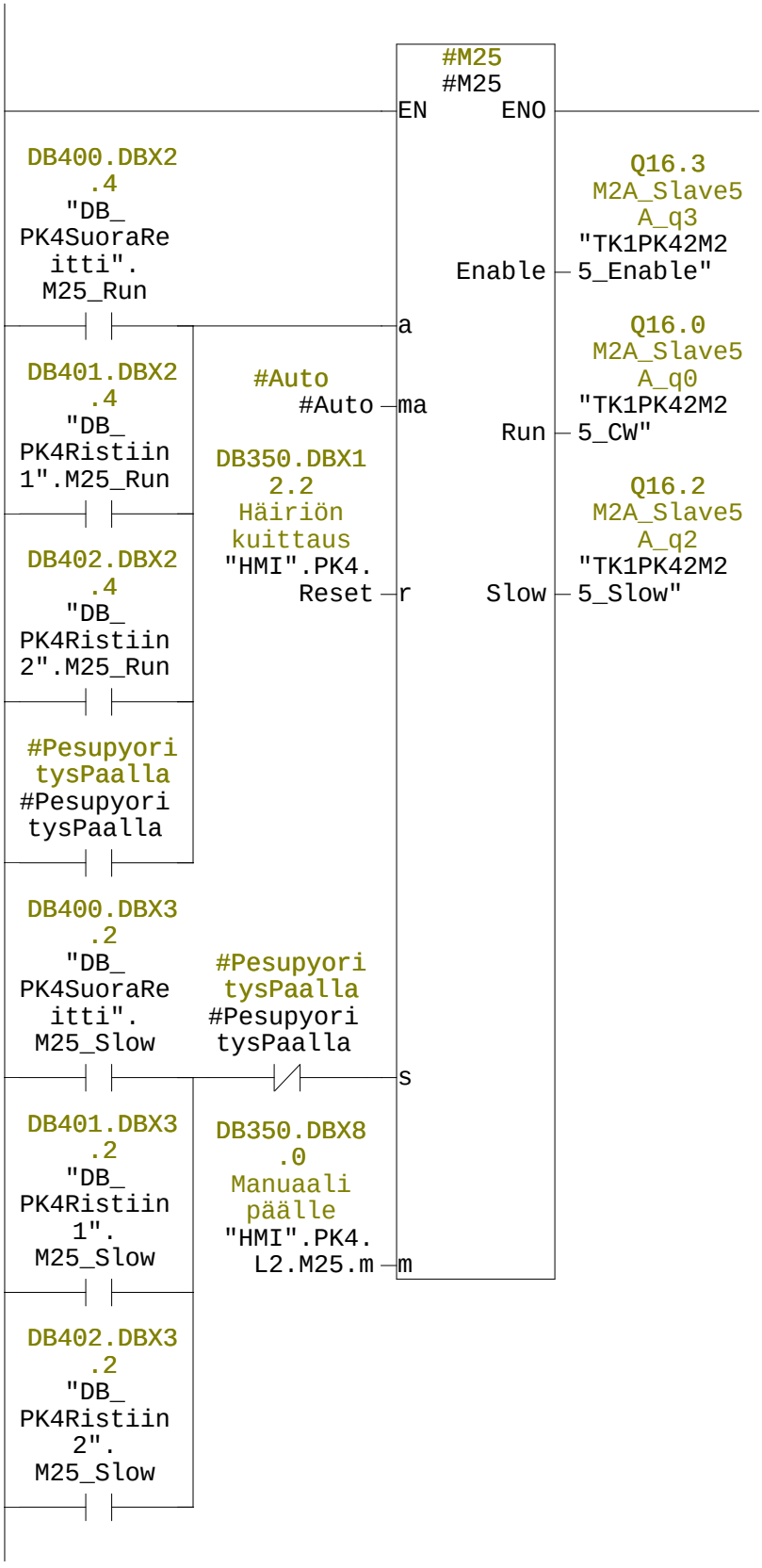
Network: 8 TK1PK4.1M15



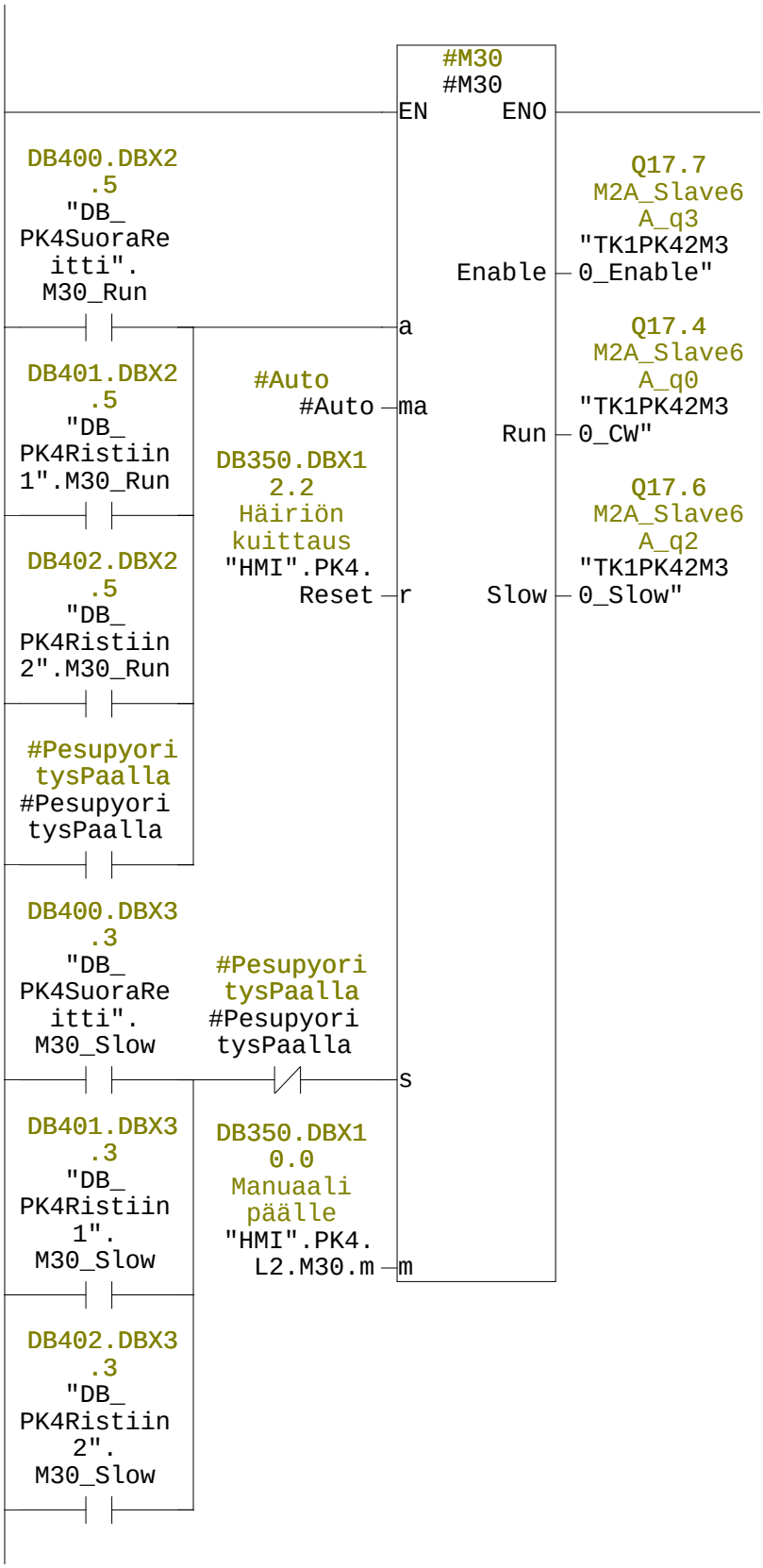
Network: 9 TK1PK4.2M20



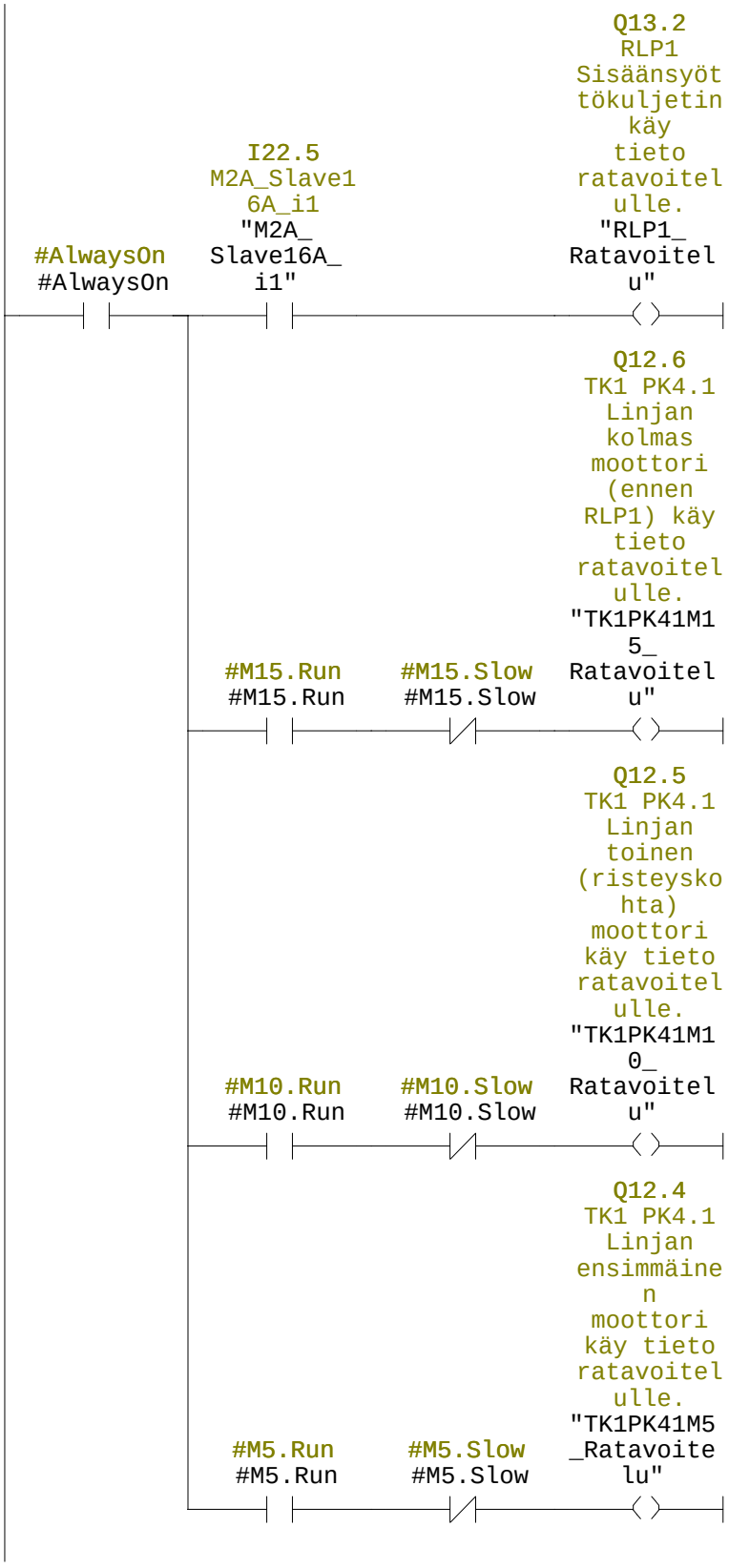
Network: 10 TK1PK4.2M25



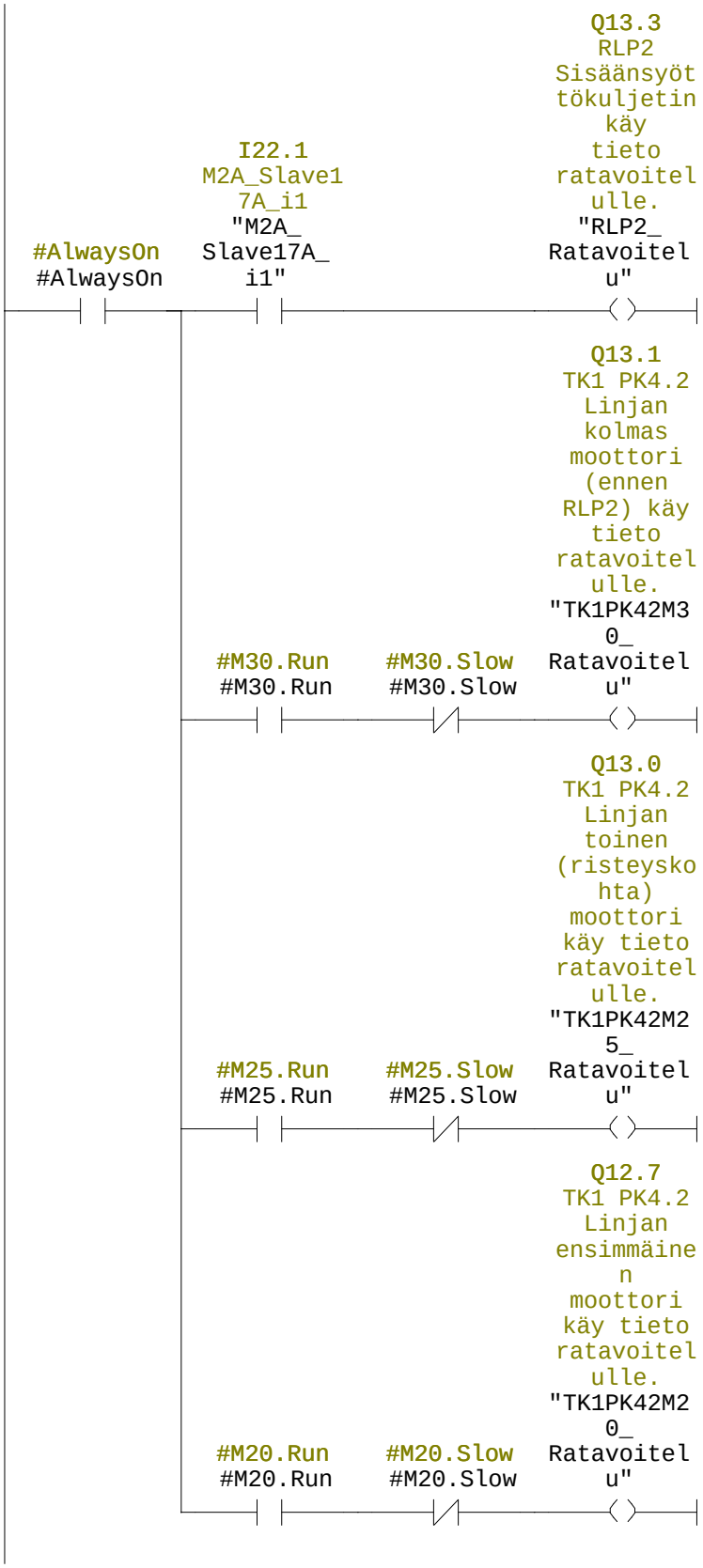
Network: 11 TK1PK4.2M30



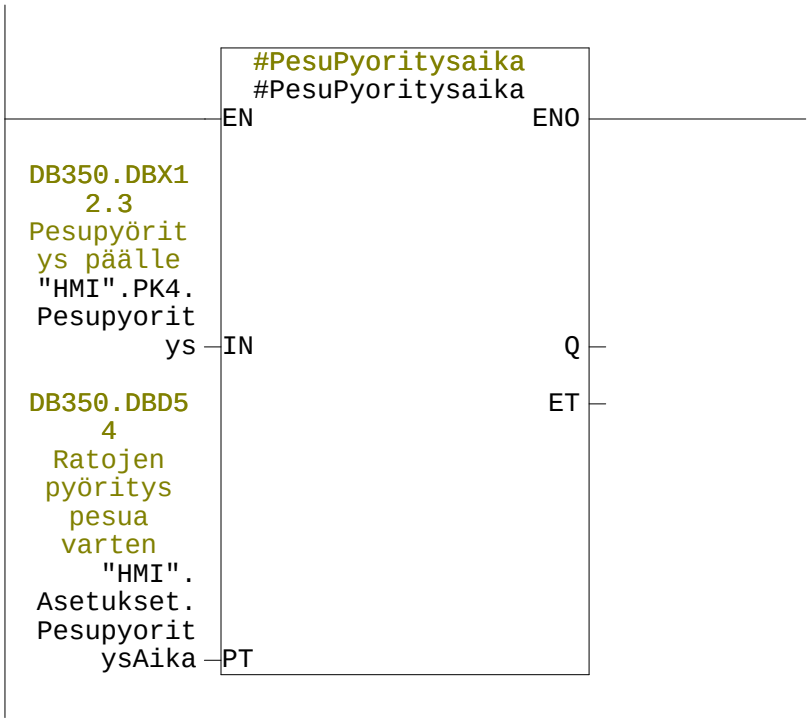
Network: 12 Ratavoitelutiedot PK4.1



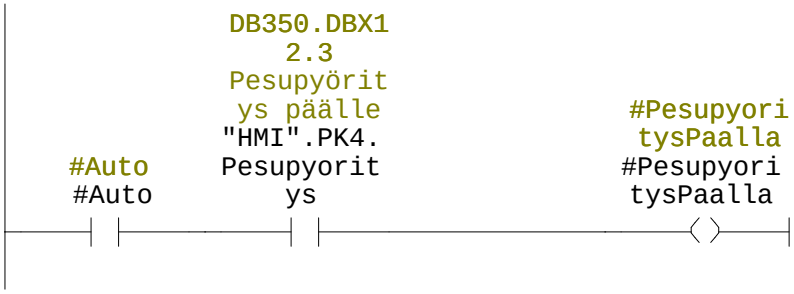
Network: 13 Ratavoitelutiedot PK4.2



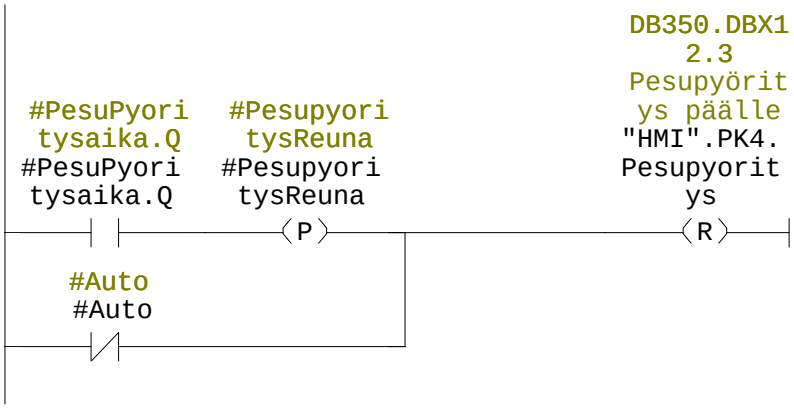
Network: 14 Pesupyöritysaika



Network: 15 Pesupyöritys päällä



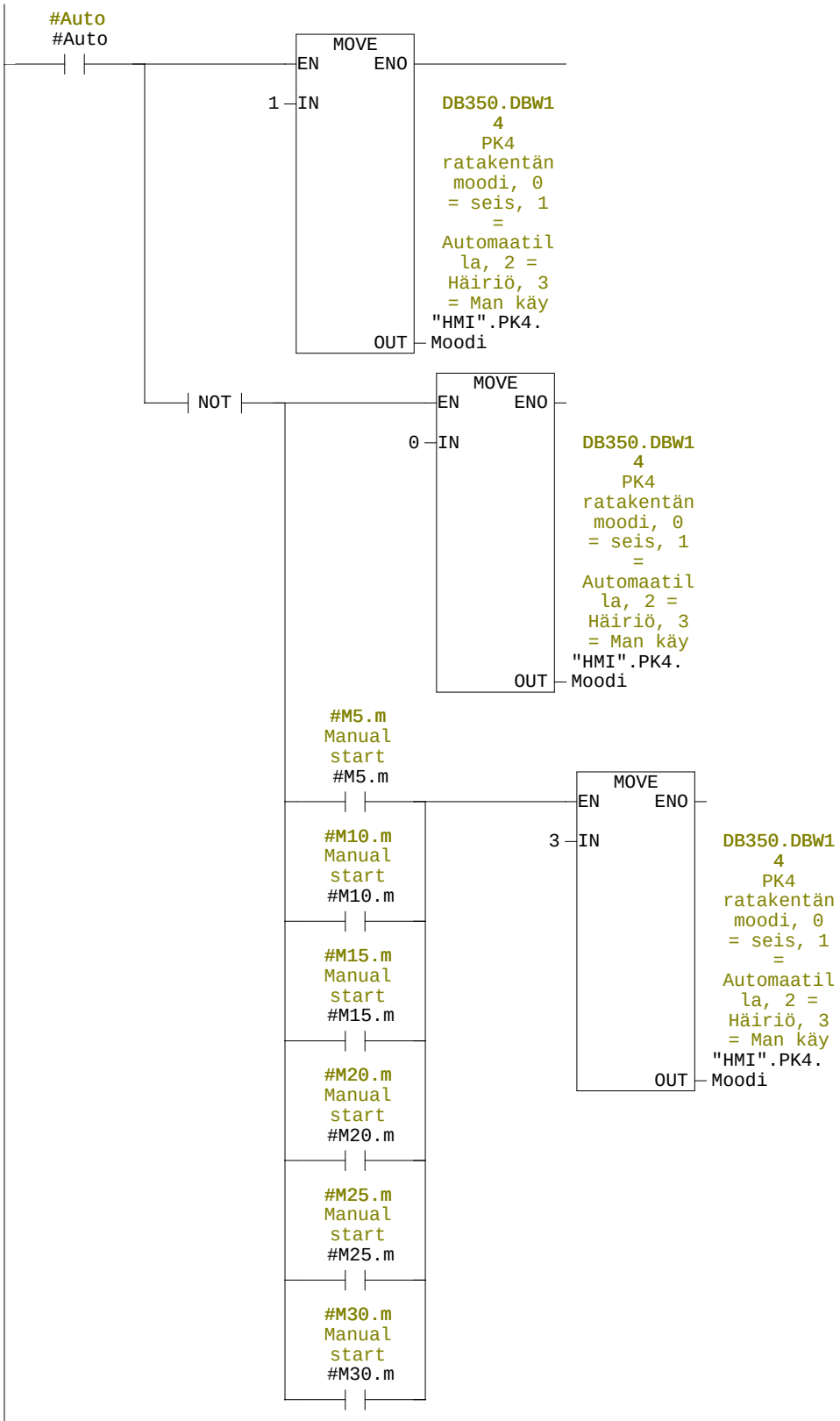
Network: 16 Pesupyöritys päällä painikkeen resetointi



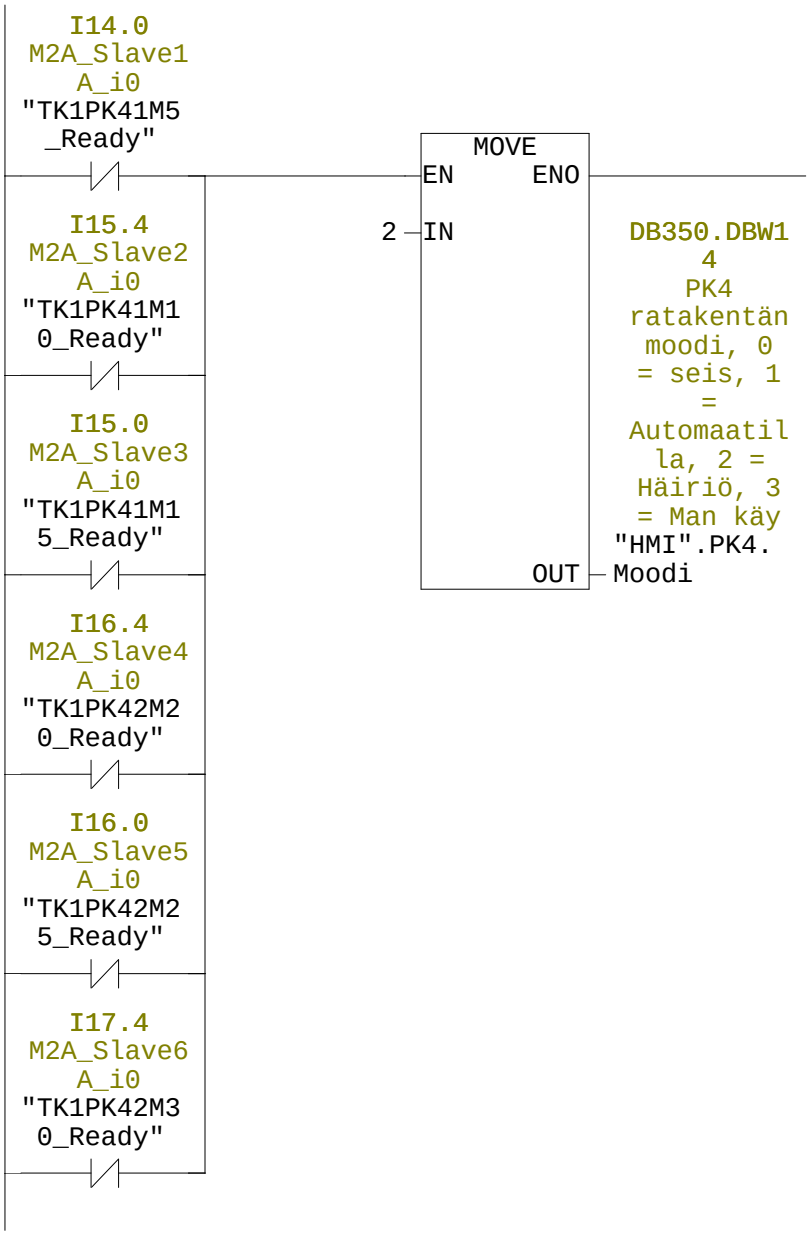
Network: 17 Jäljellä oleva pesupyöritysaika

```
//Jos pesupyöritys ei ole päällä, siirretään 0 jäljellä olevaan aikaan
//Jos pesupyöritys on päällä, vähennetään kulunut aika kokonaisajasta ja
//siirretään jäljellä olevaan aikaan

      AN    #PesupyoritysPaalla               #PesupyoritysPaalla
      JC    zero

      L     "HMI".Asetukset.PesupyoritysAika  DB350.DBD54        -- Ratojen pyöritys pesua varte
                                              n
      L     #PesuPyoritysaika.ET              #PesuPyoritysaika.ET
      -D

      JU    end
zero: L     T#0MS
end:  T     "HMI".PK4.PesupyoritystaJaljella  DB350.DBD16        -- Jäljellä oleva pesupyöritysa
                                              ika
```

Network: 18 Moodi näytölle



Network: 19 Jokin moottori häiriössä



FB400 - <offline>

"PK4 SuoraReitti" PK4.1 -> RLP1, PK4.2 -> RLP2 "normaali" ajovalinnan reitti ohjaus

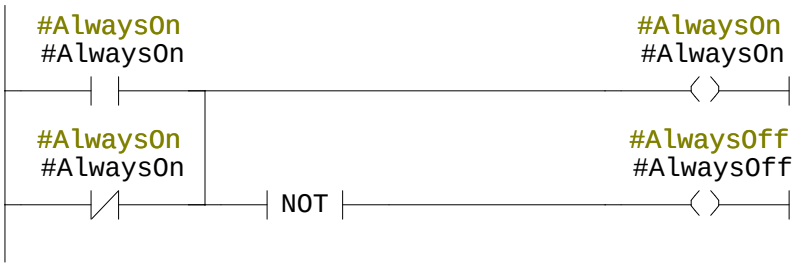
Name: **Family:**
Author: Muotio **Version:** 0.1
Block version: 2
Time stamp Code: 06/25/2014 11:16:56 AM
Interface: 05/21/2014 03:48:38 PM
Lengths (block/logic/data): 02080 00902 00010

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Auto	Bool	0.0	FALSE	
OUT		0.0		
M5_Run	Bool	2.0	FALSE	
M10_Run	Bool	2.1	FALSE	
M15_Run	Bool	2.2	FALSE	
M20_Run	Bool	2.3	FALSE	
M25_Run	Bool	2.4	FALSE	
M30_Run	Bool	2.5	FALSE	
M5_Slow	Bool	2.6	FALSE	
M10_Slow	Bool	2.7	FALSE	
M15_Slow	Bool	3.0	FALSE	
M20_Slow	Bool	3.1	FALSE	
M25_Slow	Bool	3.2	FALSE	
M30_Slow	Bool	3.3	FALSE	
InhibitFillingMachineL1	Bool	3.4	FALSE	
InhibitFillingMachineL2	Bool	3.5	FALSE	
IN_OUT		0.0		
STAT		0.0		
TK1PK41M5	PackageConveyor	4.0		
TK1PK41M10	PackageConveyor	148.0		
TK1PK41M15	PackageConveyor	292.0		
TK1PK42M20	PackageConveyor	436.0		
TK1PK42M25	PackageConveyor	580.0		
TK1PK42M30	PackageConveyor	724.0		
TK1PK41M5_Run	Bool	868.0	FALSE	
TK1PK41M10_Run	Bool	868.1	FALSE	
TK1PK41M15_Run	Bool	868.2	FALSE	
TK1PK42M20_Run	Bool	868.3	FALSE	
TK1PK42M25_Run	Bool	868.4	FALSE	
TK1PK42M30_Run	Bool	868.5	FALSE	
TEMP		0.0		
AlwaysOn	Bool	0.0		
AlwaysOff	Bool	0.1		

Name	Data Type	Address	Initial Value	Comment
PK41RLP2	Bool	0.2		
PK42RLP1	Bool	0.3		

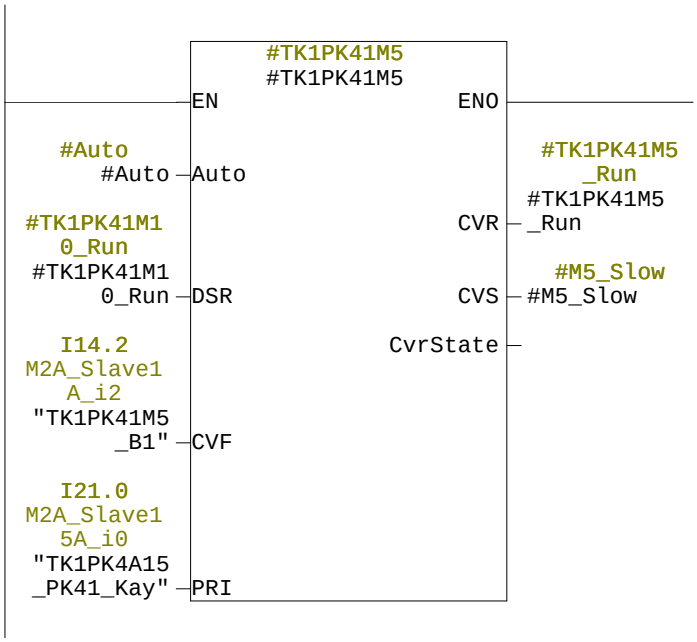
Block: FB400

Network: 1 AlwaysOn, AlwaysOff

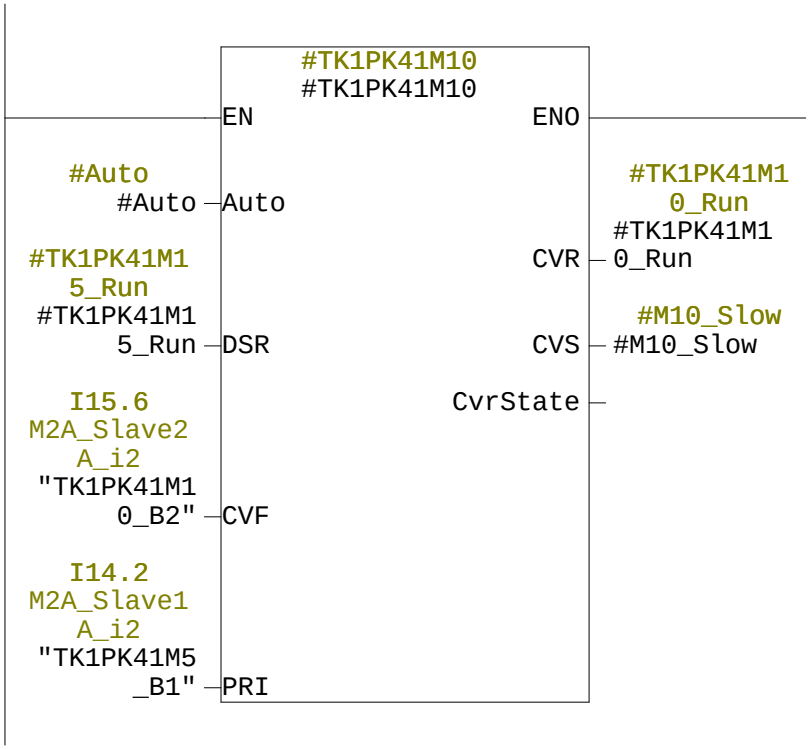


Network: 2 PK4.1 -> RLP1

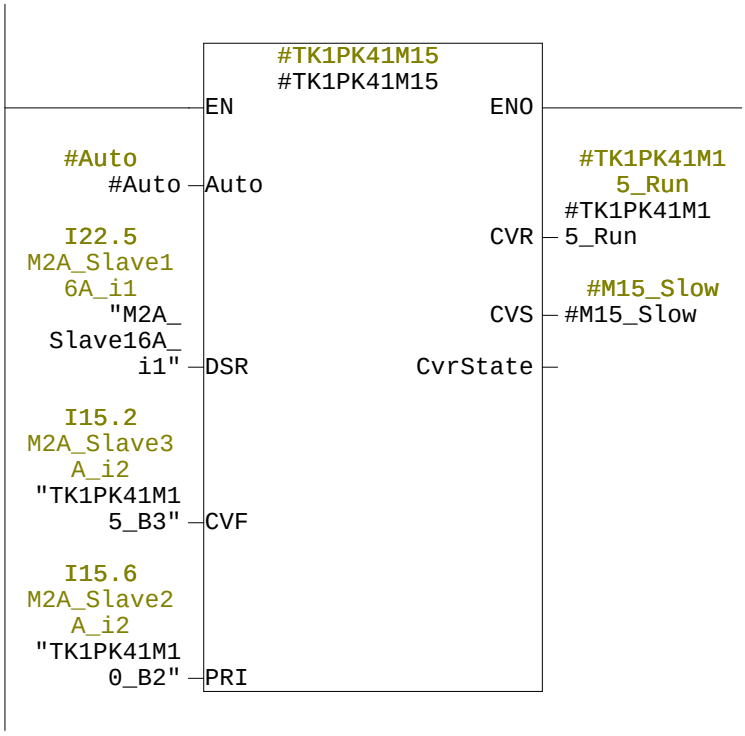
Network: 3 TK1PK4.1M5. Ensimmäinen rata täyttökoneen jälkeen



Network: 4 TK1PK4.1M10. Toinen rata täyttökoneen jälkeen

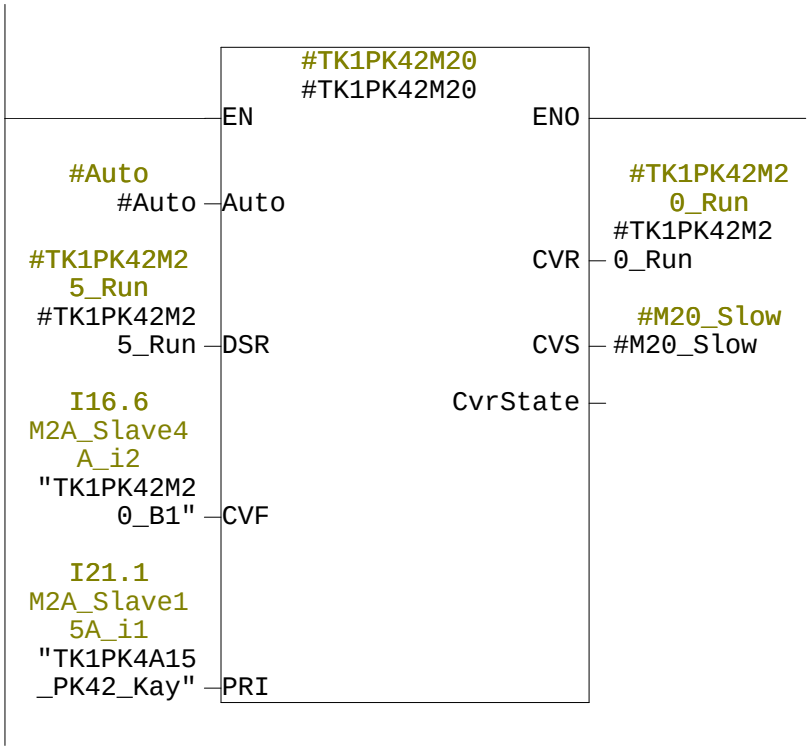


Network: 5 TK1PK4.1M15. Kolmas rata täyttökoneen jälkeen.

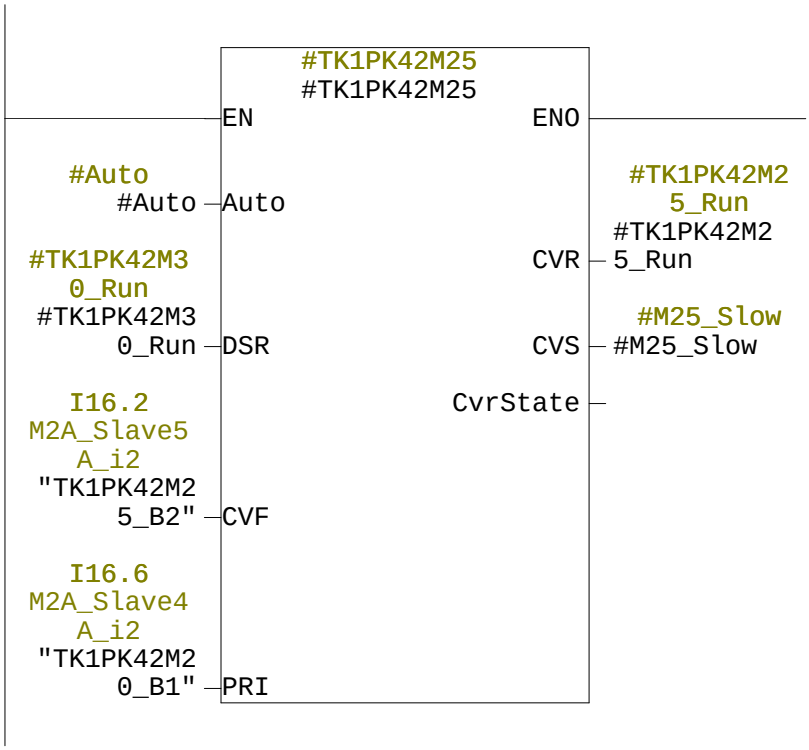


Network: 6 PK4.2 -> RLP2

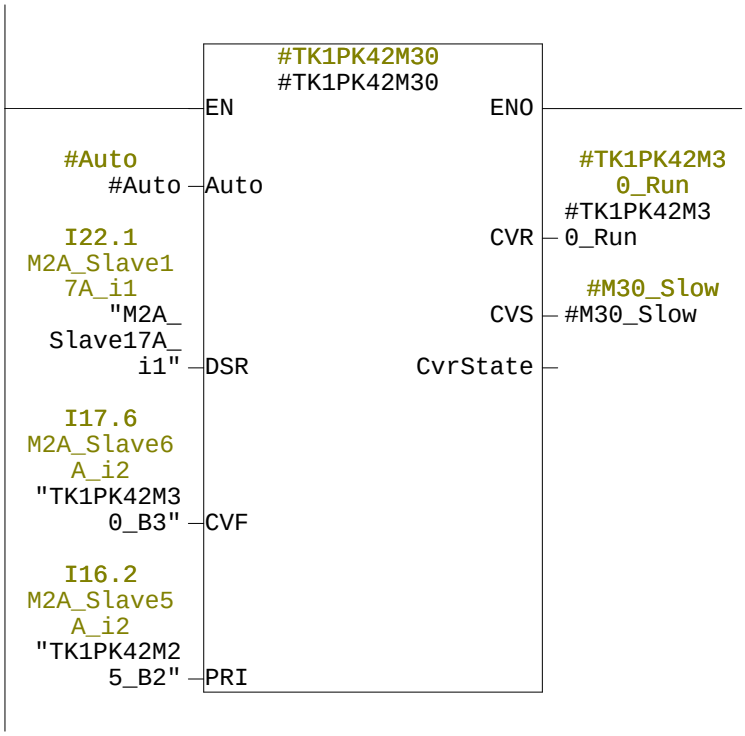
Network: 7 TK1PK4.2M20. Ensimmäinen rata täyttökoneen jälkeen



Network: 8 TK1PK4.2M25. Toinen rata täyttökoneen jälkeen



Network: 9 TK1PK4.2M30. Kolmas rata täyttökoneen jälkeen.



Network: 10 Pakkaus ei ok signaali linja 1

```
O(
L     #TK1PK41M5.CvrState       #TK1PK41M5.CvrState
L     3
>=I
)
O(
L     #TK1PK41M10.CvrState      #TK1PK41M10.CvrState
L     3
>=I
)
O(
L     #TK1PK41M15.CvrState      #TK1PK41M15.CvrState
L     3
>=I
)
=     #InhibitFillingMachineL1  #InhibitFillingMachineL1
```

Network: 11 Pakkaus ei ok signaali linja 2

```
O(
L     #TK1PK42M20.CvrState      #TK1PK42M20.CvrState
L     3
>=I
)
O(
L     #TK1PK42M25.CvrState      #TK1PK42M25.CvrState
L     3
>=I
)
O(
L     #TK1PK42M30.CvrState      #TK1PK42M30.CvrState
L     3
>=I
)
=     #InhibitFillingMachineL2  #InhibitFillingMachineL2
```

Network: 12 Moottori käy

```
A     #TK1PK41M5_Run   #TK1PK41M5_Run
=     #M5_Run          #M5_Run

A     #TK1PK41M10_Run  #TK1PK41M10_Run
=     #M10_Run         #M10_Run

A     #TK1PK41M15_Run  #TK1PK41M15_Run
=     #M15_Run         #M15_Run

A     #TK1PK42M20_Run  #TK1PK42M20_Run
=     #M20_Run         #M20_Run

A     #TK1PK42M25_Run  #TK1PK42M25_Run
=     #M25_Run         #M25_Run

A     #TK1PK42M30_Run  #TK1PK42M30_Run
=     #M30_Run         #M30_Run
```

FB401 - <offline>

"PK4 Ristiin1" PK4.1 -> RLP2, "ristiin" ajovalinnan reitti ohjaus

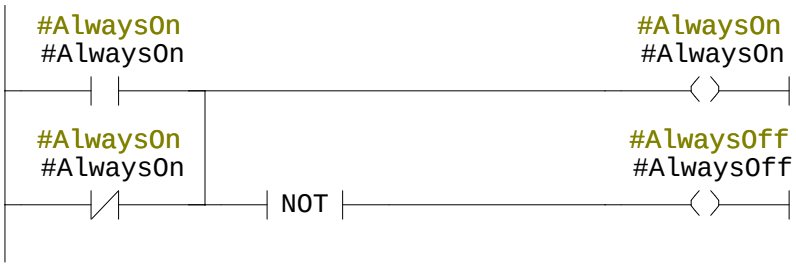
Name: **Family:**
Author: Muotio **Version:** 0.1
Block version: 2
Time stamp Code: 06/25/2014 11:17:03 AM
Interface: 04/25/2014 03:15:55 PM
Lengths (block/logic/data): 01936 00762 00012

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Auto	Bool	0.0	FALSE	
OUT		0.0		
M5_Run	Bool	2.0	FALSE	
M10_Run	Bool	2.1	FALSE	
M15_Run	Bool	2.2	FALSE	
M20_Run	Bool	2.3	FALSE	
M25_Run	Bool	2.4	FALSE	
M30_Run	Bool	2.5	FALSE	
M5_Slow	Bool	2.6	FALSE	
M10_Slow	Bool	2.7	FALSE	
M15_Slow	Bool	3.0	FALSE	
M20_Slow	Bool	3.1	FALSE	
M25_Slow	Bool	3.2	FALSE	
M30_Slow	Bool	3.3	FALSE	
InhibitForFillingMachine	Bool	3.4	FALSE	
IN_OUT		0.0		
STAT		0.0		
TK1PK41M5	PackageConveyor	4.0		
TK1PK41M10	PackageConveyor	148.0		
TK1PK41M15	PackageConveyor	292.0		
TK1PK42M20	PackageConveyor	436.0		
TK1PK42M25	PackageConveyor	580.0		
TK1PK42M30	PackageConveyor	724.0		
TK1PK41M5_Run	Bool	868.0	FALSE	
TK1PK41M10_Run	Bool	868.1	FALSE	
TK1PK41M15_Run	Bool	868.2	FALSE	
TK1PK42M20_Run	Bool	868.3	FALSE	
TK1PK42M25_Run	Bool	868.4	FALSE	
TK1PK42M30_Run	Bool	868.5	FALSE	
TEMP		0.0		
AlwaysOn	Bool	0.0		
AlwaysOff	Bool	0.1		
PK41RLP2	Bool	0.2		

Name	Data Type	Address	Initial Value	Comment
PK42RLP1	Bool	0.3		
Dummy	Int	2.0		

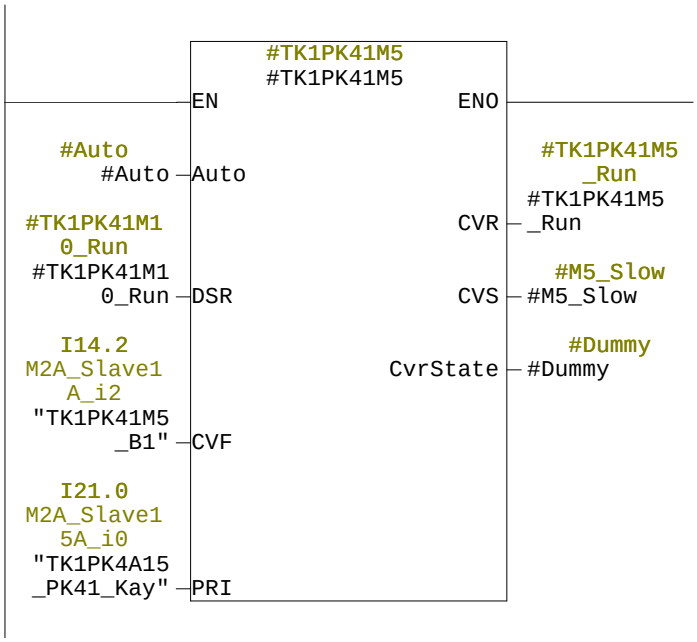
Block: FB401

Network: 1 AlwaysOn, AlwaysOff



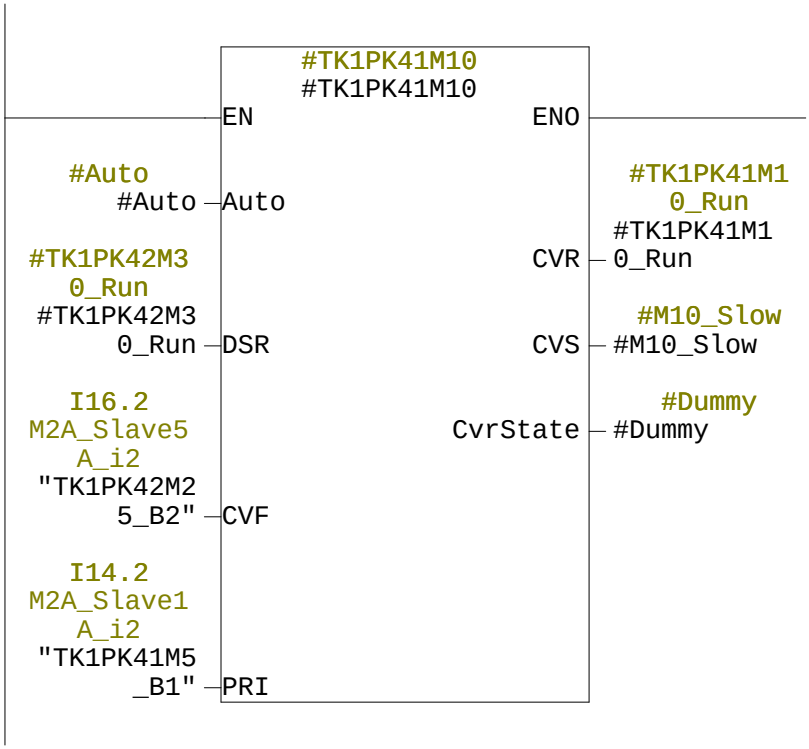
Network: 2 PK4.1 -> RLP2, PK4.2 ei ole käytettävissä tässä ajovalinnassa

Network: 3 TK1PK4.1M5. Ensimmäinen rata täyttökoneen jälkeen

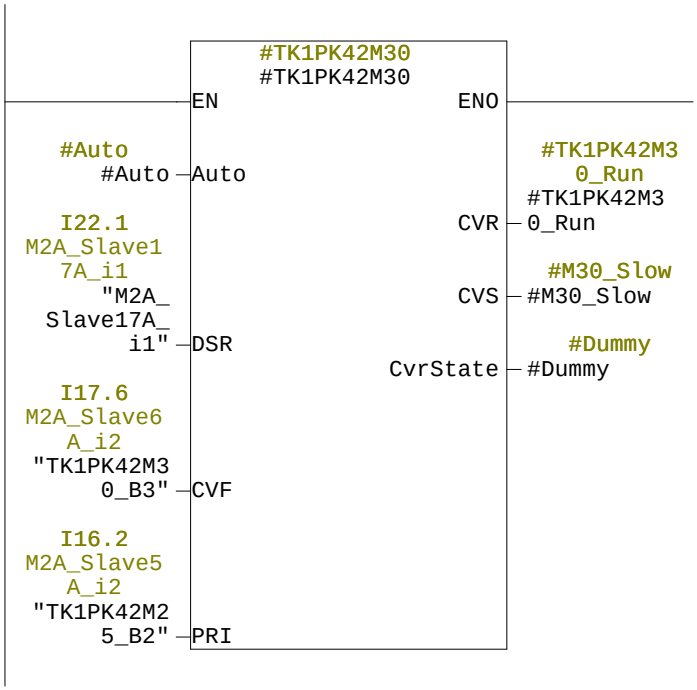


Network: 4 TK1PK4.1M10 ja TK1PK4.2M25. Toinen rata täyttökoneen jälkeen.

Risteyskohta, radan tulokenno on TK1PK4.1M10 tulokenno ja poistumiskenno on radan TK1PK4.2M25 poistumiskenno. Kumpaakin rataa ohjataan samaan aikaan.

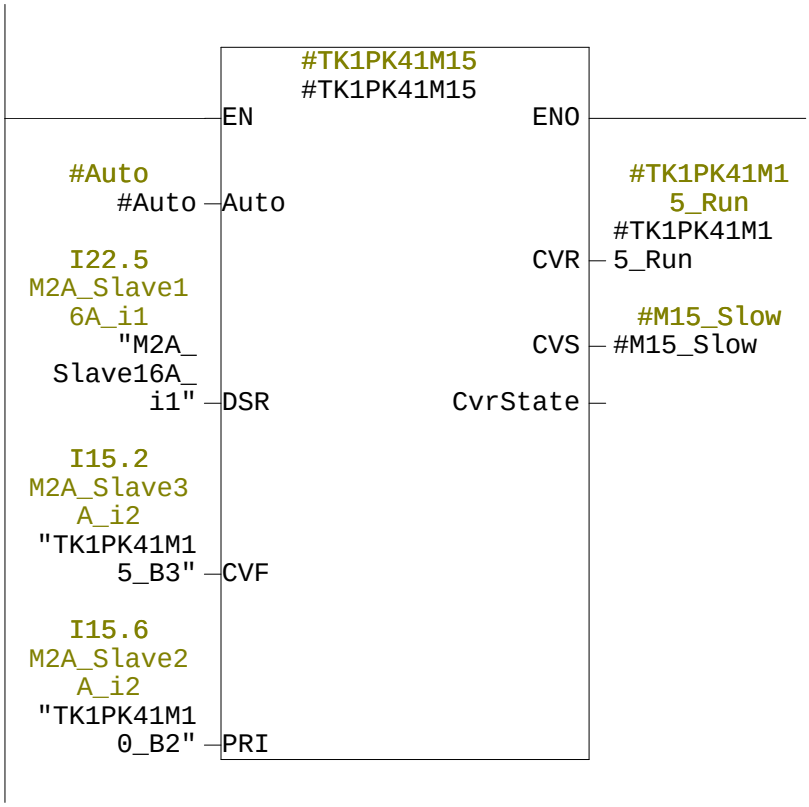


Network: 5 TK1PK4.2M30. Kolmas rata täyttökoneen jälkeen.

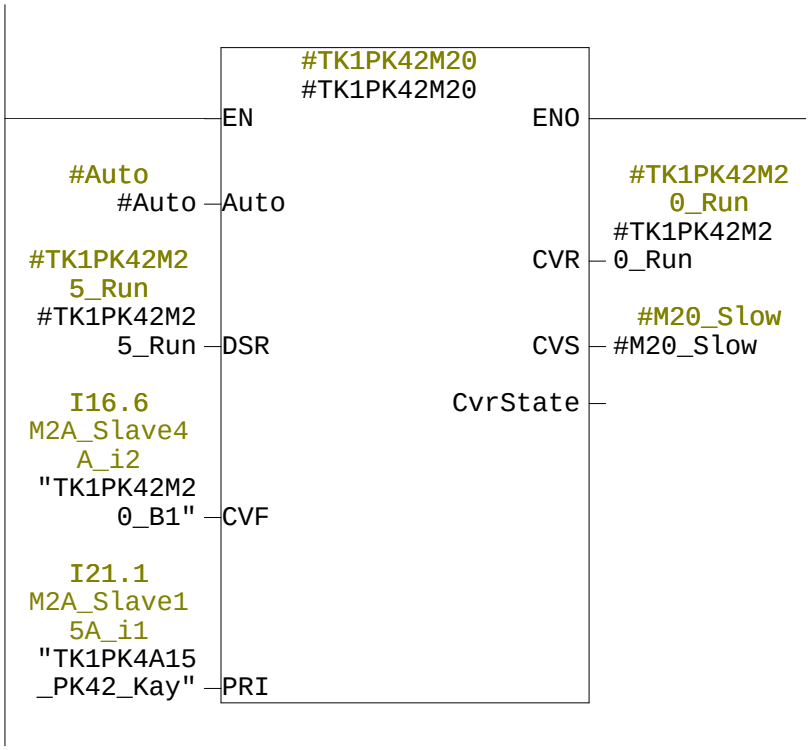


Network: 6 Käyttämättömät radat.

Network: 7 TK1PK4.1M15



Network: 8 TK1PK4.2M20



Network: 9 Pakkaus ei ok signaali

```
O(
L     #TK1PK41M5.CvrState        #TK1PK41M5.CvrState
L     3
>=I
)
O(
L     #TK1PK41M10.CvrState       #TK1PK41M10.CvrState
L     3
>=I
)
O(
L     #TK1PK42M30.CvrState       #TK1PK42M30.CvrState
L     3
>=I
)
=     #InhibitForFillingMachine  #InhibitForFillingMachine
```

Network: 10 Moottori käy

```
A     #TK1PK41M5_Run   #TK1PK41M5_Run
=     #M5_Run          #M5_Run

A     #TK1PK41M10_Run  #TK1PK41M10_Run
=     #M10_Run         #M10_Run
=     #M25_Run         #M25_Run
```

```
      A     #TK1PK42M30_Run  #TK1PK42M30_Run
      =     #M30_Run         #M30_Run

      A     #TK1PK41M10.CVS  #TK1PK41M10.CVS    -- CONVEYOR SLOW SPEED, Signal out, this device
                             on slow speed
      =     #M25_Slow        #M25_Slow



//Käyttämättömät radat
      A     #TK1PK41M15_Run  #TK1PK41M15_Run
      =     #M15_Run         #M15_Run

      A     #TK1PK42M20_Run  #TK1PK42M20_Run
      =     #M20_Run         #M20_Run
```

FB402 - <offline>

"PK4 Ristiin2" PK4.2 -> RLP1, "ristiin" ajovalinnan reitti ohjaus

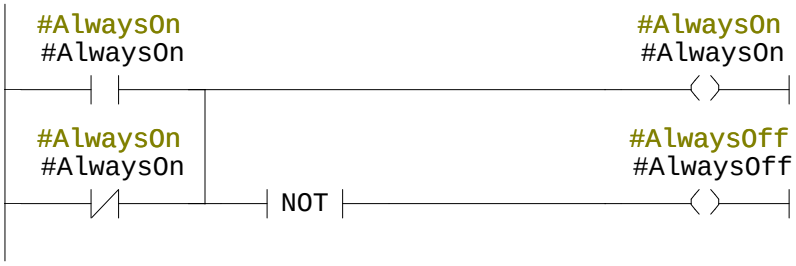
Name: **Family:**
Author: Muotio **Version:** 0.1
Block version: 2
Time stamp Code: 06/25/2014 11:17:08 AM
Interface: 04/25/2014 03:16:54 PM
Lengths (block/logic/data): 01910 00738 00010

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Auto	Bool	0.0	FALSE	
OUT		0.0		
M5_Run	Bool	2.0	FALSE	
M10_Run	Bool	2.1	FALSE	
M15_Run	Bool	2.2	FALSE	
M20_Run	Bool	2.3	FALSE	
M25_Run	Bool	2.4	FALSE	
M30_Run	Bool	2.5	FALSE	
M5_Slow	Bool	2.6	FALSE	
M10_Slow	Bool	2.7	FALSE	
M15_Slow	Bool	3.0	FALSE	
M20_Slow	Bool	3.1	FALSE	
M25_Slow	Bool	3.2	FALSE	
M30_Slow	Bool	3.3	FALSE	
InhibitForFillingMachine	Bool	3.4	FALSE	
IN_OUT		0.0		
STAT		0.0		
TK1PK41M5	PackageConveyor	4.0		
TK1PK41M10	PackageConveyor	148.0		
TK1PK41M15	PackageConveyor	292.0		
TK1PK42M20	PackageConveyor	436.0		
TK1PK42M25	PackageConveyor	580.0		
TK1PK42M30	PackageConveyor	724.0		
TK1PK41M5_Run	Bool	868.0	FALSE	
TK1PK41M10_Run	Bool	868.1	FALSE	
TK1PK41M15_Run	Bool	868.2	FALSE	
TK1PK42M20_Run	Bool	868.3	FALSE	
TK1PK42M25_Run	Bool	868.4	FALSE	
TK1PK42M30_Run	Bool	868.5	FALSE	
TEMP		0.0		
AlwaysOn	Bool	0.0		
AlwaysOff	Bool	0.1		
PK41RLP2	Bool	0.2		

Name	Data Type	Address	Initial Value	Comment
PK42RLP1	Bool	0.3		

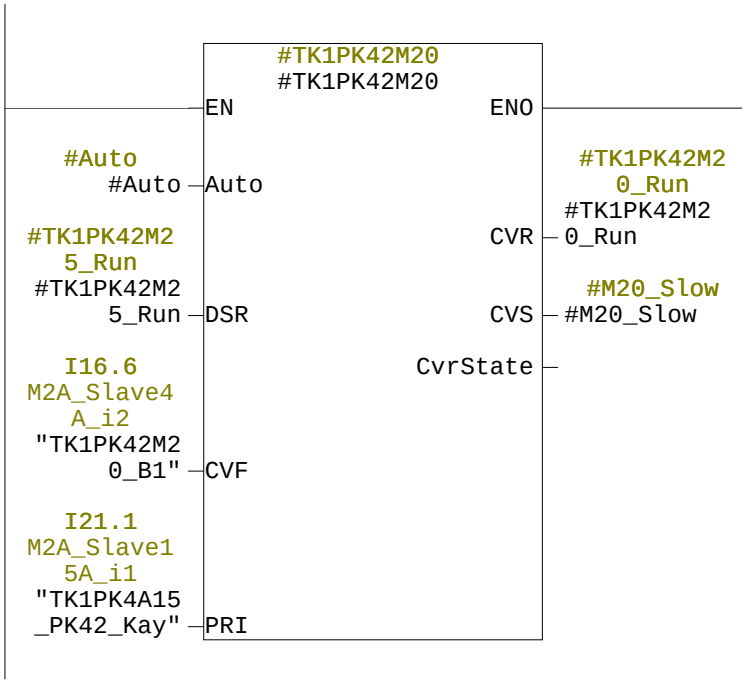
Block: FB402

Network: 1 AlwaysOn, AlwaysOff



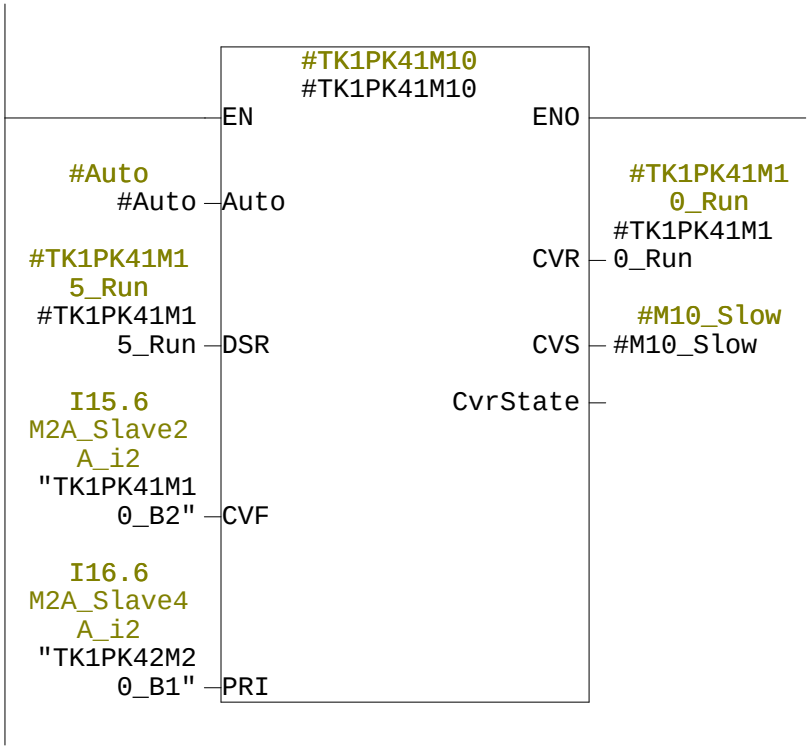
Network: 2 PK4.2 -> RLP1, PK4.1 ei ole käytettävissä tässä ajovalinnassa

Network: 3 TK1PK4.2M20. Ensimmäinen rata täyttökoneen jälkeen

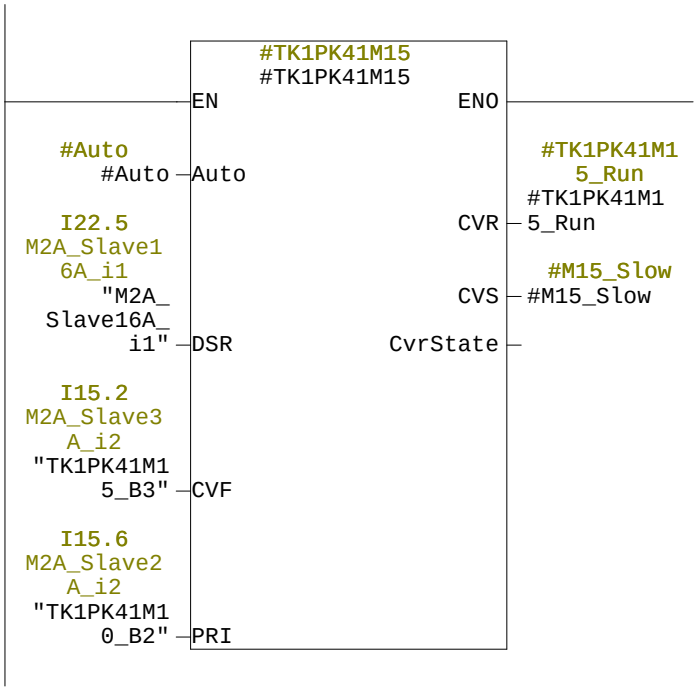


Network: 4 TK1PK4.1M10. Toinen rata täyttökoneen jälkeen

Risteyskohta, radan tulokenno on TK1PK4.2M25 tulokenno ja poistumiskenno on radan TK1PK4.1M10 poistumiskenno. Kumpaakin rataa ohjataan samaan aikaan.

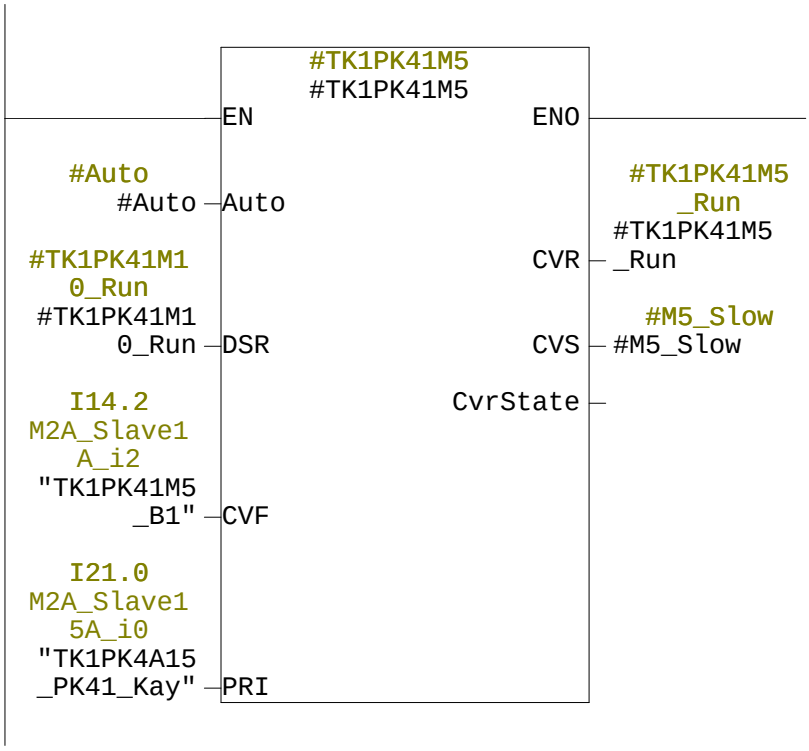


Network: 5 TK1PK4.1M15. Kolmas rata täyttökoneen jälkeen.

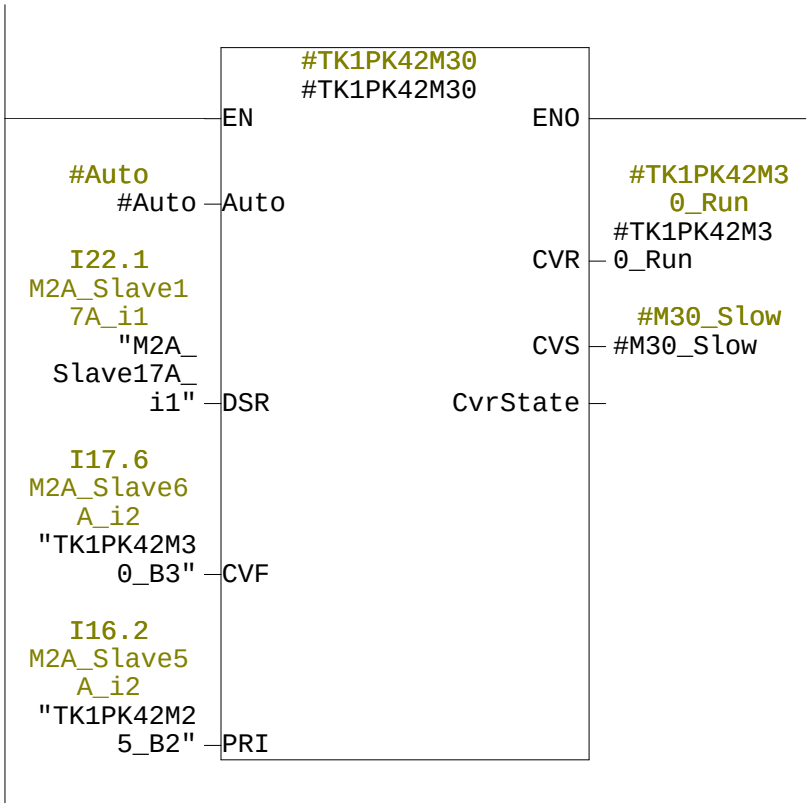


Network: 6 Käyttämättömät radat

Network: 7 TK1PK4.1M5. Ensimmäinen rata täyttökoneen jälkeen.



Network: 8 TK1PK4.2M30. Kolmas rata täyttökoneen jälkeen.



Network: 9 Pakkaus ei ok signaali

```
O(
L     #TK1PK42M20.CvrState       #TK1PK42M20.CvrState
L     3
>=I
)
O(
L     #TK1PK41M10.CvrState       #TK1PK41M10.CvrState
L     3
>=I
)
O(
L     #TK1PK41M15.CvrState       #TK1PK41M15.CvrState
L     3
>=I
)
=     #InhibitForFillingMachine  #InhibitForFillingMachine
```

Network: 10 Moottori käy

```
      A     #TK1PK42M20_Run  #TK1PK42M20_Run
      =     #M20_Run         #M20_Run

      A     #TK1PK41M10_Run  #TK1PK41M10_Run
      =     #M10_Run         #M10_Run
      =     #M25_Run         #M25_Run

      A     #TK1PK41M15_Run  #TK1PK41M15_Run
      =     #M15_Run         #M15_Run

      A     #TK1PK41M10.CVS  #TK1PK41M10.CVS    -- CONVEYOR SLOW SPEED, Signal out, this device
                             on slow speed
      =     #M25_Slow        #M25_Slow


//Käyttämättömät radat

      A     #TK1PK41M5_Run   #TK1PK41M5_Run
      =     #M5_Run          #M5_Run

      A     #TK1PK42M30_Run  #TK1PK42M30_Run
      =     #M30_Run         #M30_Run
```

FB423 - <offline>

"PK4 JPEIOK" PK4:lle ruuhka ja jälkipää ei ok tiedot

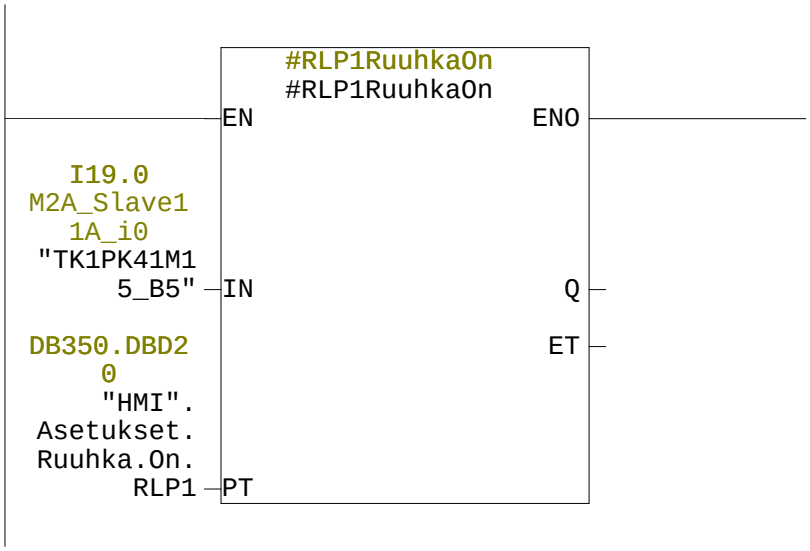
Name: **Family:**
Author: Muotio **Version:** 0.1
Block version: 2
Time stamp Code: 10/29/2014 12:51:01 PM
Interface: 04/07/2014 02:31:48 PM
Lengths (block/logic/data): 01170 00874 00010

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Auto	Bool	0.0	FALSE	
OUT		0.0		
IN_OUT		0.0		
STAT		0.0		
RLP1RuuhkaOn	TON	2.0		
RLP1RuuhkaOff	TON	24.0		
RLP1JpEiOkOn	TON	46.0		
RLP1JpEiOkOff	TON	68.0		
RLP2RuuhkaOn	TON	90.0		
RLP2RuuhkaOff	TON	112.0		
RLP2JpEiOkOn	TON	134.0		
RLP2JpEiOkOff	TON	156.0		
TEMP		0.0		
Dummy	Bool	0.0		
PK41ReitinMoottoritOK	Bool	0.1		
PK42ReitinMoottoritOK	Bool	0.2		

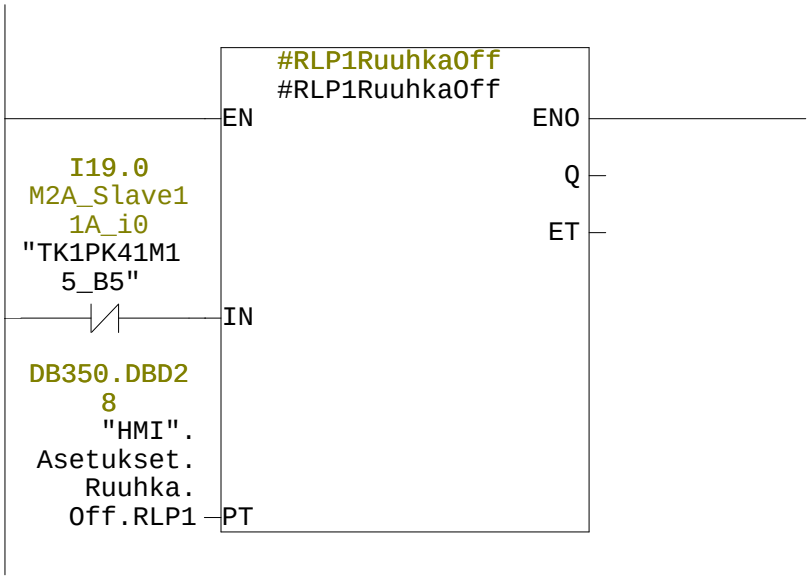
Block: FB423 Jälkipää ok tiedot PK4 koneelle

Network: 1

Network: 2 Ruuhka päälle RLP1 menevä rata

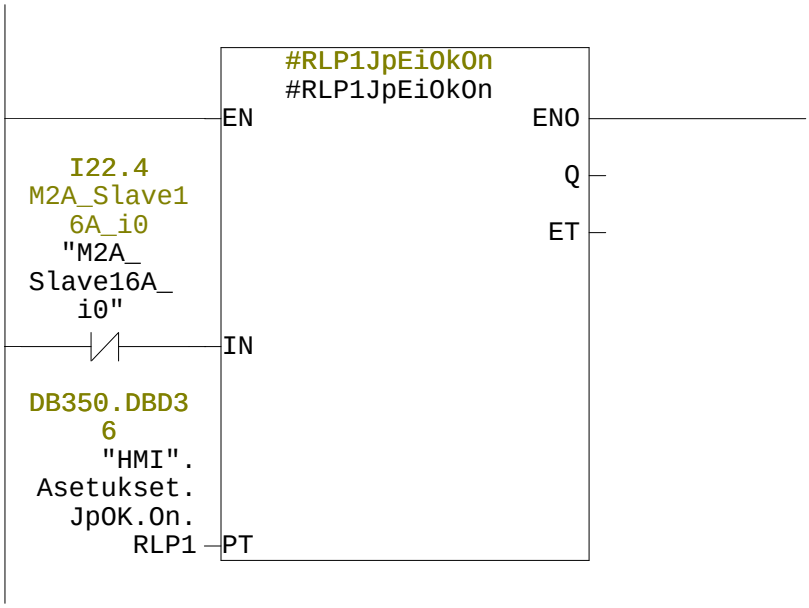


Network: 3

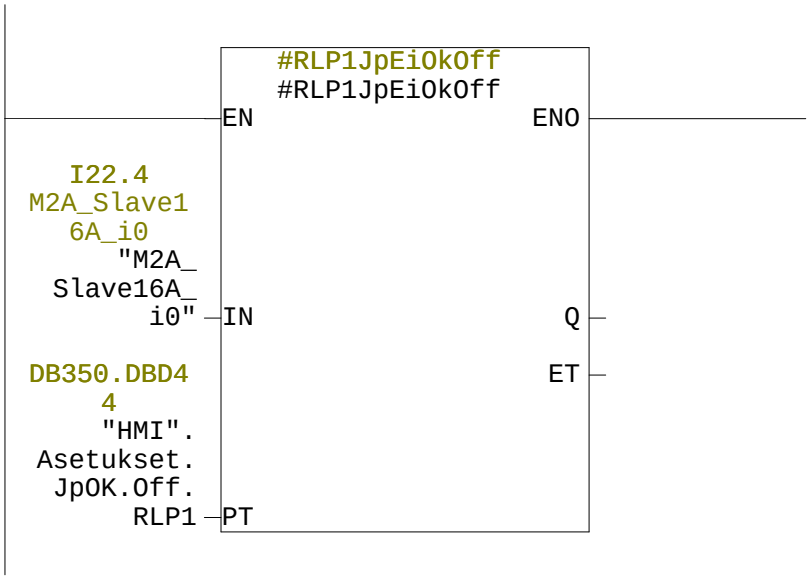


Network: 4

Network: 5 RLP1 ok

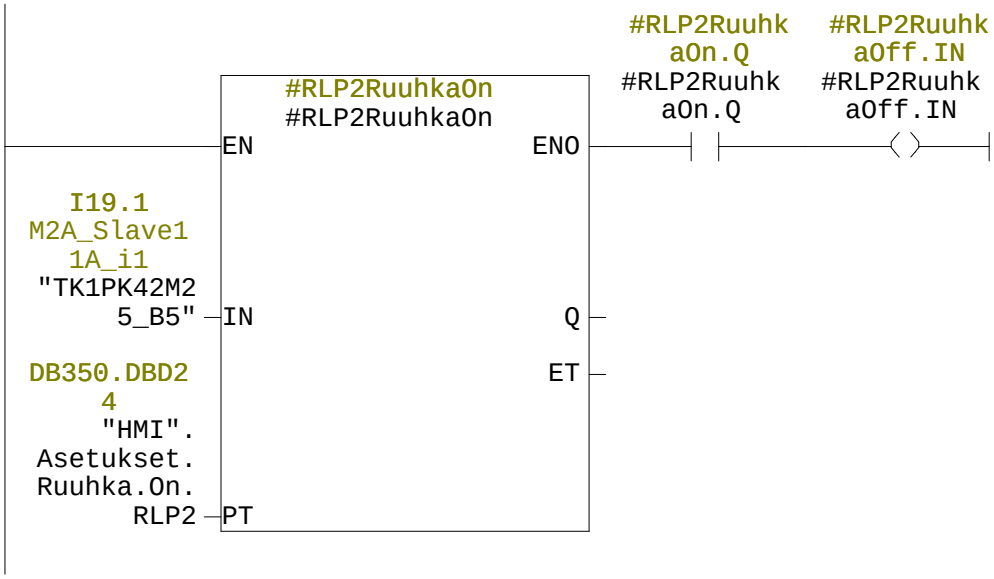


Network: 6

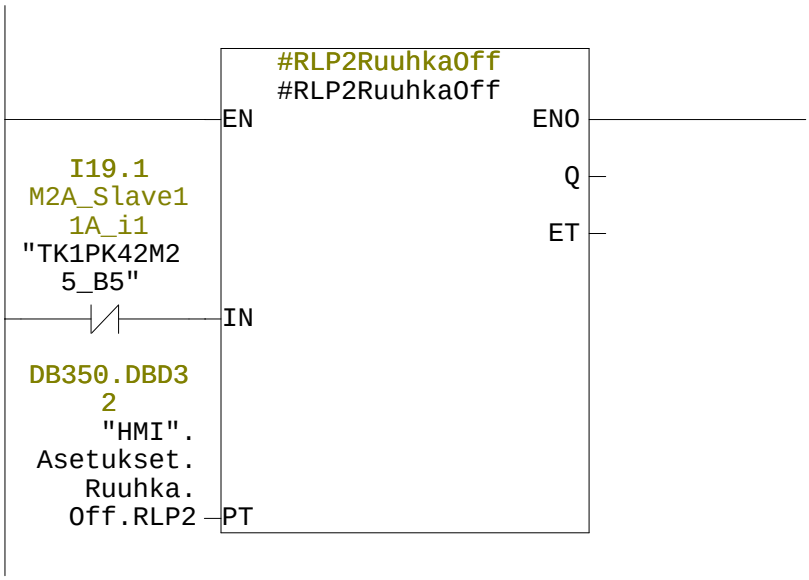


Network: 7

Network: 8 Ruuhka päälle RLP2 menevä rata

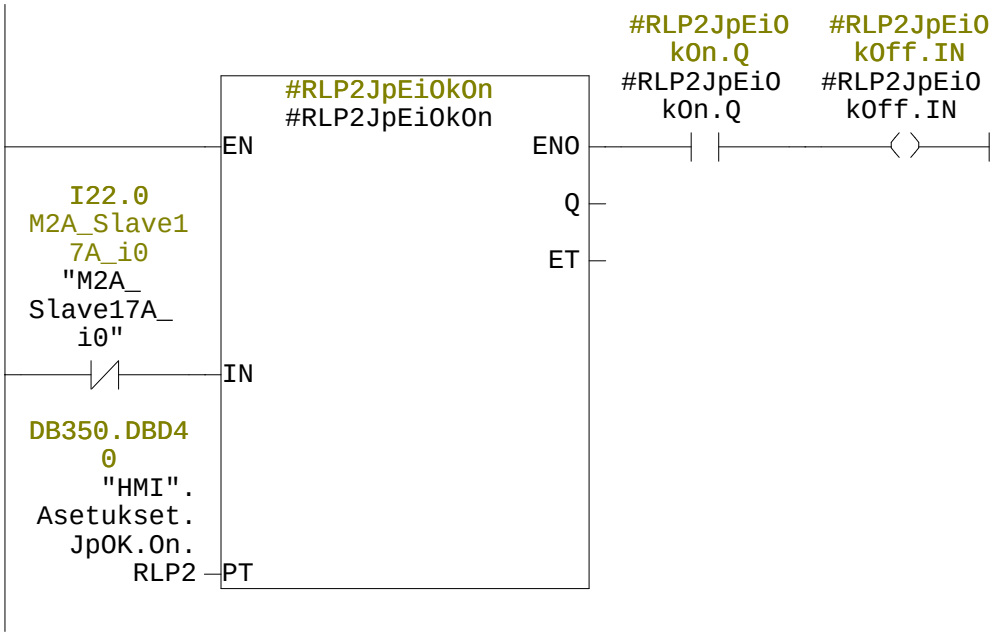


Network: 9

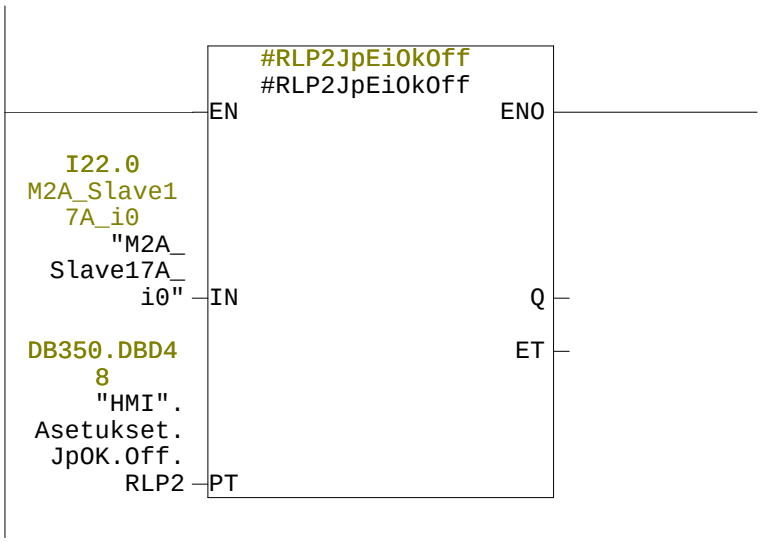


Network: 10

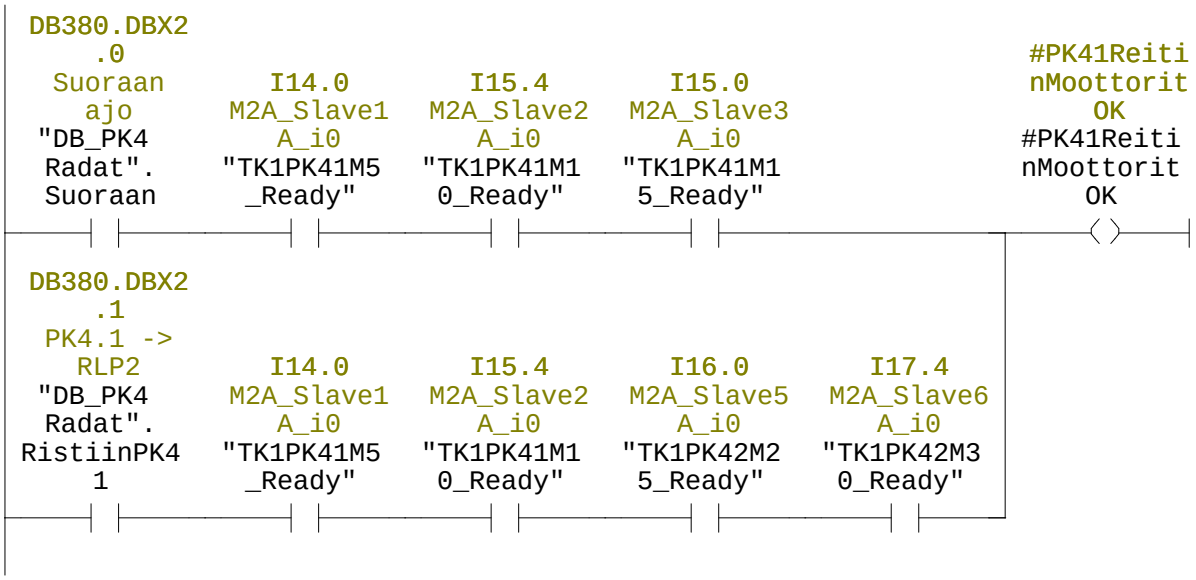
Network: 11 RLP2 ok



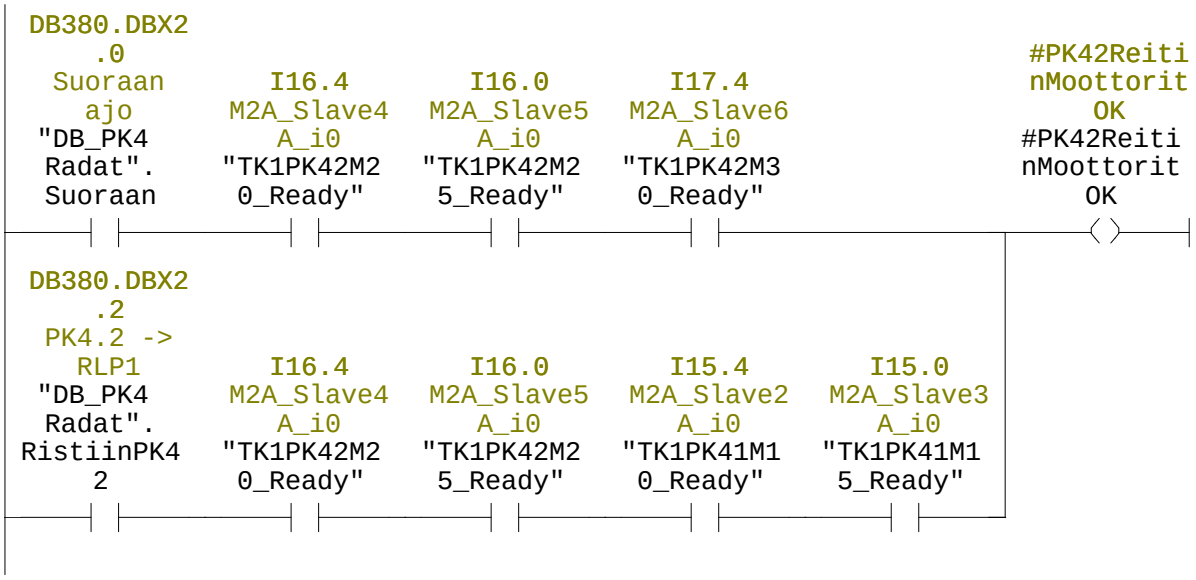
Network: 12



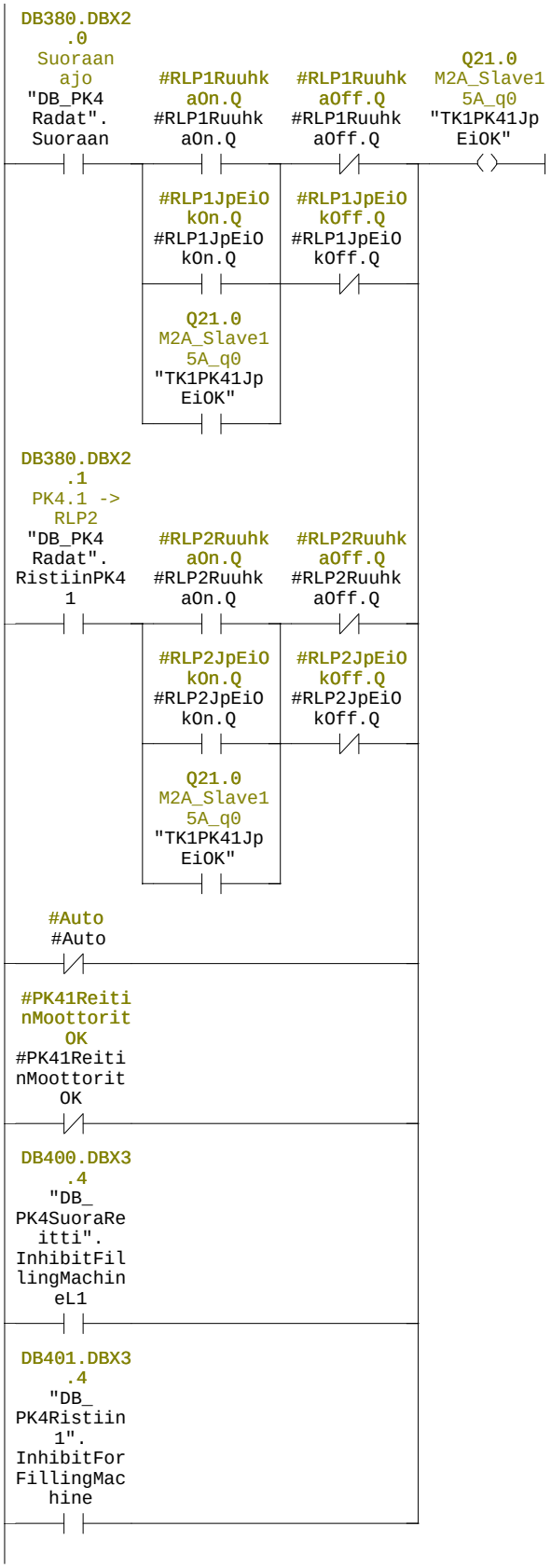
Network: 13 PK4.1 reitin moottorit ok



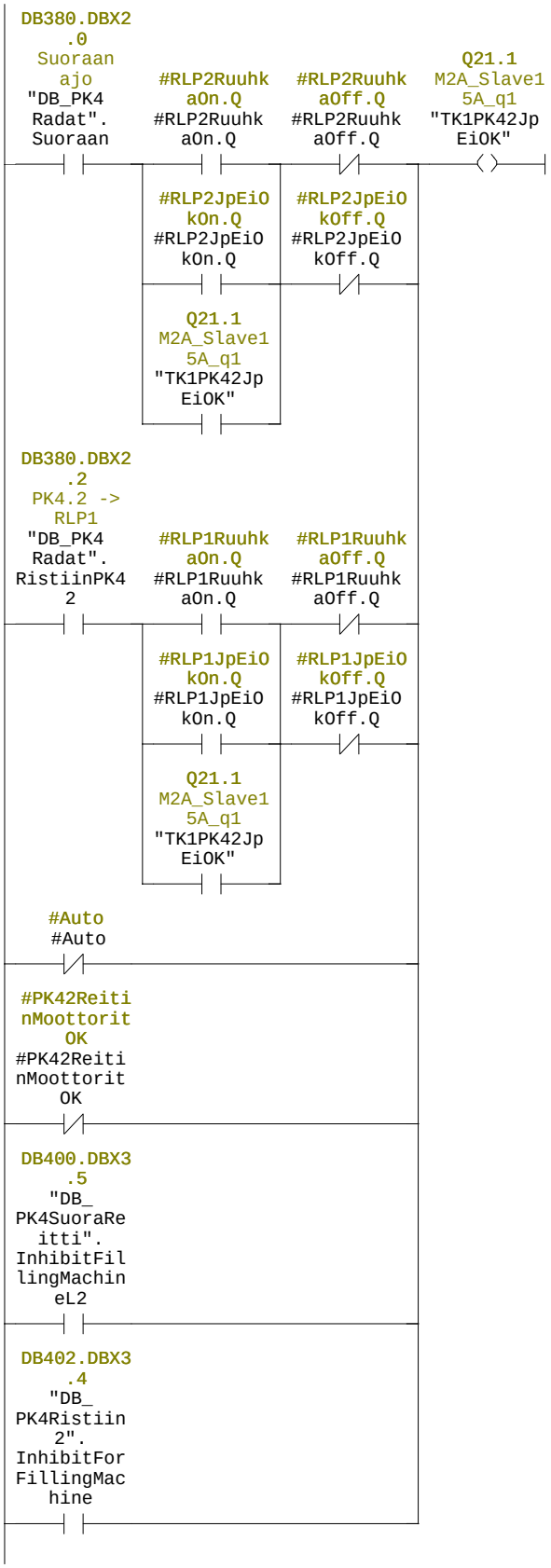
Network: 14 PK4.2 reitin moottorit ok



Network: 15 PK4.1 Ruuhka päällä tai jälkipää ei ok



Network: 16 PK4.2 Ruuhka päällä tai jälkipää ei ok



Network: 17 TK1PK4.1M15-B3 valokenno eteenpäin RLP1:lle

Network: 18 TK1PK4.2M30-B3 valokenno eteenpäin RLP2:lle