TEKNIIKKA JA LIIKENNE

Kone- ja tuotantotekniikka

Koneautomaatio

INSINÖÖRITYÖ

MTS2-KOKOONPANOLINJAN TILAKONEOHJELMAN KEHITYSTYÖT

Työn tekijä: Samuli Aalto
Työn ohjaaja: Jari Savolainen

Työ hyväksytty: 5. 11. 2008

Jari Savolainen
TkT

ALKULAUSE

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoululle, sen koneautomaatiolaboratoriossa sijaitsevasta MTS2-kokoonpanolinjaston tilakoneohjelmasta syksyn 2007 ja kevään 2008 välisenä aikana. Haluan kiittää työn valvojaa ja ohjaajaa Jari Savolaista sekä koko koneautomaatiolaboratorion henkilökuntaa.

Helsingissä 1.9.2008

Samuli Aalto

OPINNÄYTETYÖN TIIVISTELMÄ

Työn tekijä: Samuli Aalto

Työn nimi: MTS2-kokoonpanolinjan tilakoneohjelman kehitystyöt

Päivämäärä: 1.9.2008 **Sivumäärä:** 29 s. + 6 liitettä

Koulutusohjelma: Kone- ja tuotantotekniikka

Ammatillinen suuntautuminen: Koneautomaatio

Työn ohjaaja: TkT Jari Savolainen

Tämän insinöörityön tarkoituksena oli tehdä korjauksia sekä täydennyksiä alkuperäiseen MTS2-kokoonpanolinjaston tilakoneohjelmaan sekä luoda ohjelman logiikasta esitettävä tilakonemalli. Linjalle on tehty ohjelma, jonka avulla linjaa pystyttiin ohjelman komennoin ohjaamaan. Ohjelma oli puutteellinen, jollaisena sitä ei voinut käyttää opetuksessa. Toimilaitteet ja anturit olivat valmiiksi kytketyt ja toimivat.

Aluksi ohjelmaan ja laitteistoon tutustuttiin ohjaajan avustamana. Ohjelman vikatilanteiden ja puutteiden löytämiseksi oli tehtävä testiajoja. Linjan ohjelma jakautui osiin, joita tuli testata ja muokata sekä erikseen että linkitettyinä. Linjan toiminnasta tuli hakea virhetilanteet ja tehdä tarvittavat muutostyöt ohjelman rakenteeseen.

Ohjelman esitystä varten tuli tutustua Metaedit+-ohjelmaan, jonka avulla luotiin esitys erikseen jokaiselle tilakoneelle. Tilakoneiden toimintavaiheet selitettiin sanallisesti kommentteina. Tehtävän tilakonemallin tuli esittää selkeästi ohjelman toiminnan eri vaiheet.

Kirjallisessa työssä kerrotaan linjan toimintaperiaate, linjan eri toimilaitteet ja linjastossa käytetyt kenttäväylät. Työssä esitellään ohjelma johon muutokset tehtiin. Ohjelmasta kerrotaan sen perustoiminnot, joilla linjaa ohjataan ja joita lisäämällä ja muokkaamalla voidaan linjaston toimintaa muuttaa.

Avainsanat: MTS2, C++, tilakone, Metaedit, ohjausjärjestelmä

ABSTRACT

Name: Samuli Aalto

Title: MTS2-assemblylines state maschine programs developmentwork

Date: 1.9.2008 **Number of pages:** 29 p. + 6 appendixes

Department: Mechanical engineering

Study Programme: Maschine automation

Instructor: T.Sc. Jari Savolainen

Supervisor: T.Sc. Jari Savolainen

The goal of this thesis was to make repairs and improve the original MTS2 assembly line's state machine program and to create a visual presentation of the program logic. The line already had an existing base program. The base program was used to control the line via program commands. This program was, however, inadequate and as such it could not be used during lectures. The devices and sensors were already connected and working.

At the beginning the program and installations were explored with the help of the supervisor. Test runs had to be made to identify faulty situations and imperfections in the program. The line's program was divided in parts, which were tested and modified both separately and linked. After successfully identifying the faulty situations the necessary changes were made to the program structure.

For the program's visual representation Metaedit+-program was explored and introduced. A separate representation was made for every state machine with the help of the Meatedit+-program. The phases of the state machines' actions were explained by making verbal comments. The idea behind the created state machine model was to provide a clear description of the different phases of the program.

This study introduces the line's operating principles and different devices as well as the field bus of the line. The work also describes the program to which the changes were made. The main functions, used for controlling the line, are discussed as well as how the line's operation can be changed by adding and modifying these functions.

Keywords:MTS2, C++, state maschine, Metaedit, control system

SISÄLLYS

LIITTEET

1 JOHDANTO

Työn tarkoituksena on saada Metropolia Ammattikorkeakoulun koneautomaatiolaboratoriossa sijaitseva MTS2-kokoonpanolinjasto opetuskäyttöön soveltuvaksi. MTS2-kokoonpanolinjasto (kuva 1) on peräisin saksalaisen Boschin messukäytöstä, ja se on hankittu ammattikorkeakouluun vuonna 1995.

Ohjelman pohjana käytetään Juha-Matti Äyvärin linjastosta vuonna 2007 tekemää perusohjelmaa, johon muutokset tehtiin. Ohjelma oli toteutettu C++-ohjelmointikielellä käyttämällä tilakonerakenteita. Ongelmana olivat kokoonpanolinjan käynnistyksen alussa linjan hihnoilla liikkuvat paletit, jotka jäivät linjaston eri osiin linjaston edelliseltä käyttökerralta. Jäämäpaletit jäivät haittaamaan linjaston seuraavaa käyttökertaa jumiutumalla linjaston eri osiin.

Tavoitteena oli kehittää ohjelman ajoituksia estämään palettien törmäykset risteyksissä. Laitteiston käynnistyttyä paletit oli saatava kiertoon eri linjaston kohdista sekä linjaston toiminta tuli saada toimivaksi normaaliajossa. Lisäksi antureiden toimintaa tutkittiin, sillä anturit eivät aina havainneet palettia. Nämä ongelmat saatiin poistetuksi tekemällä muutoksia ja lisäyksiä tilakoneiden toimintaan erilaisin toteutuksin.

Lisäksi tilakoneiden toiminnasta tuli tehdä kirjallinen selostus sekä piirtää tilakoneista toimintakaaviot, joiden avulla voidaan linjaston toimintaa selittää opiskelijoille.

2 MTS2-KOKOONPANOLINJAN LAITTEISTO

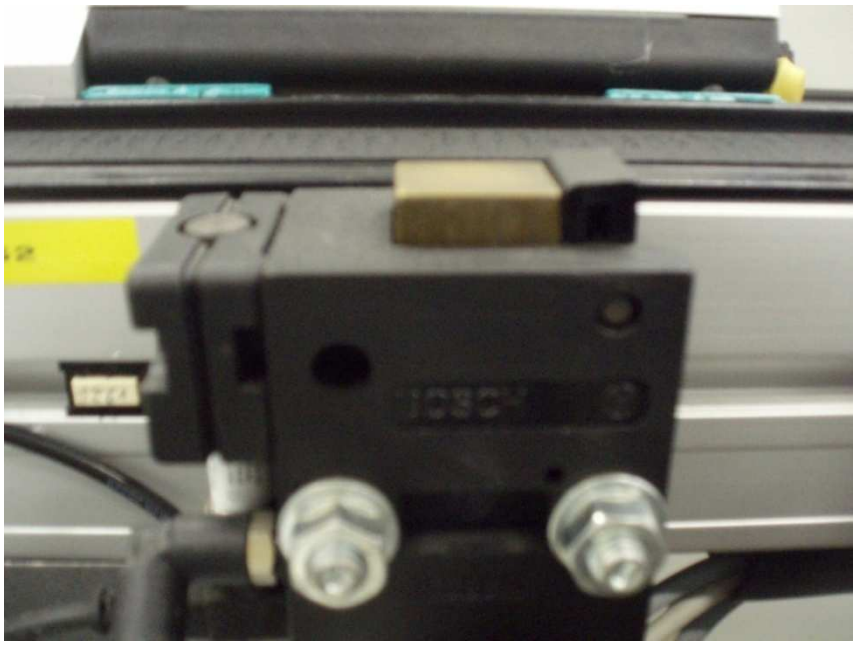
Kokoonpanolinjasto sijaitsee Metropolia Ammattikorkeakoulun koneautomaatiolaboratoriossa. Kokoonpanolinjasto on Boschin MTS2-kokoonpanolinja, joka on ollut ennen Boschin messukäytössä. Linjastossa on kolme työpistettä, joista yksi on manuaalityöpiste ja kaksi robottityöpisteitä. Robotit ovat Boschin Scara (Selective Compliance Assembly Robot Arm) SR6 ja SR60.

Kuva 1. Kokoonpanolinjasto kokonaisuutena

Linjasto siirtää paletteja (kuva 2) nauhakuljettimilla vakionopeudella sähkömoottoreilla. Paletit ovat antistaattisesta muovista valmistettuja, ja ne voivat kantaa 6 kg:n kuorman. Palettien pohjan reunoissa on upotettuina metallinapit, jotta paletit voidaan tunnistaa induktiivisten anturien avulla. Palettien pysäytys tapahtuu paineilmatoimisten pysäyttimien avulla.

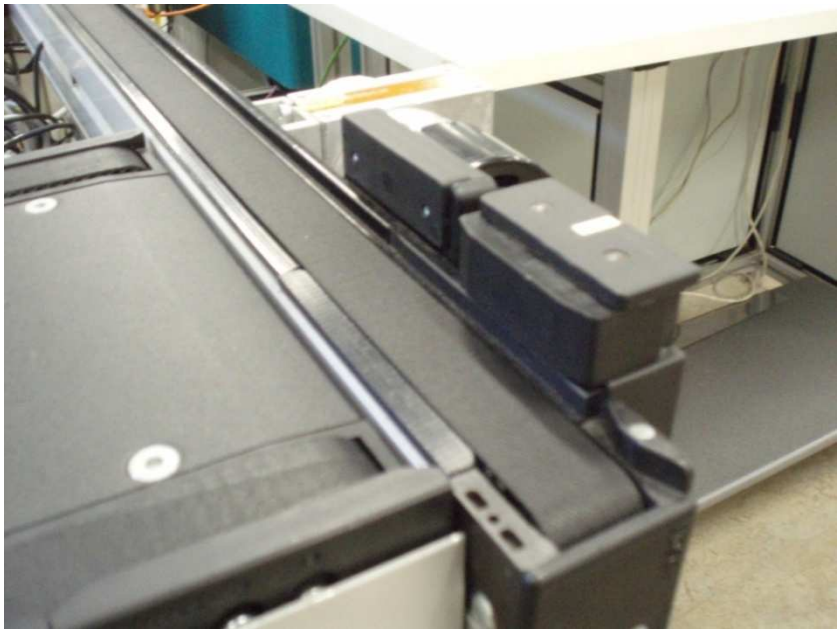
Kuva 2. Kokoonapanolinjaston paletti

Linjaston pysäyttimet (kuva 3) ovat jousipalautteisia. Pysäyttimen aktivoituessa paineilma laskee pysäyttimen metallipalan ala-asentoon. Tällöin paletti pääsee kulkemaan pysäyttimen ylitse. Kun pysäytin asetetaan poispäältä, palauttaa jousi pysäyttimen metallipalan yläasentoon, jolloin paletin kulku linjalla estyy.



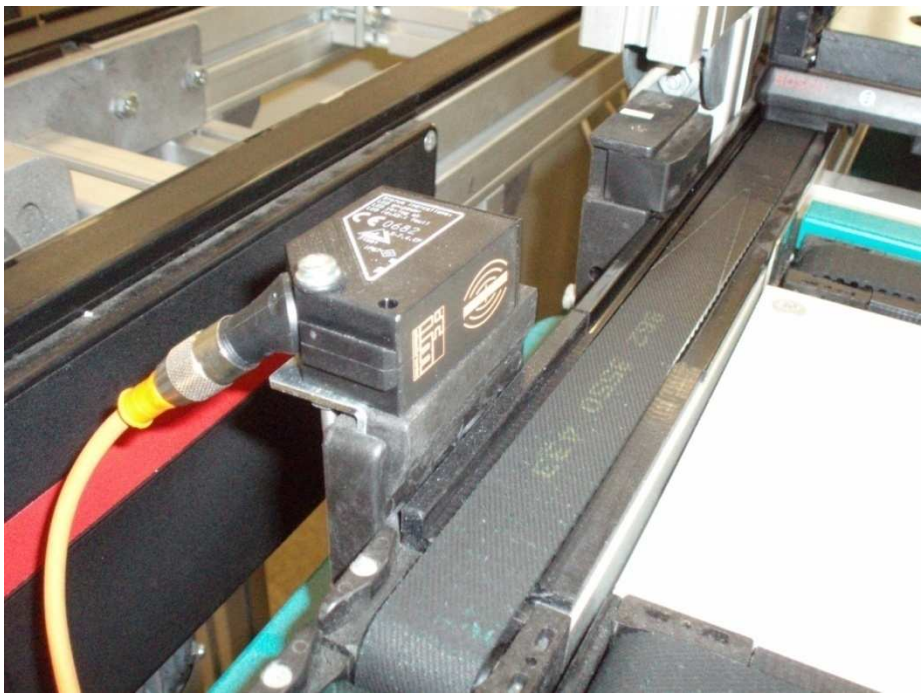
Kuva 3. Paineilmatoiminen pysäytin ja induktiivinen anturi

Palettien kulkua linjalla havainnoidaan induktiivisten antureiden (kuva 3) ja mekaanisten kytkinten (kuva 4) avulla.



Kuva 4. Mekaaninen kytkin

Jokaiselle kolmelle työpisteelle johtavassa risteyksessä on IFM:n RFID-lukija (kuva 5), joka lukee paletille yksilöllisen RFID-tagin (kuva 6) tunnisteluvun.



Kuva 5. RFID-lukija

Kuva 6. RFID-tagi paletissa

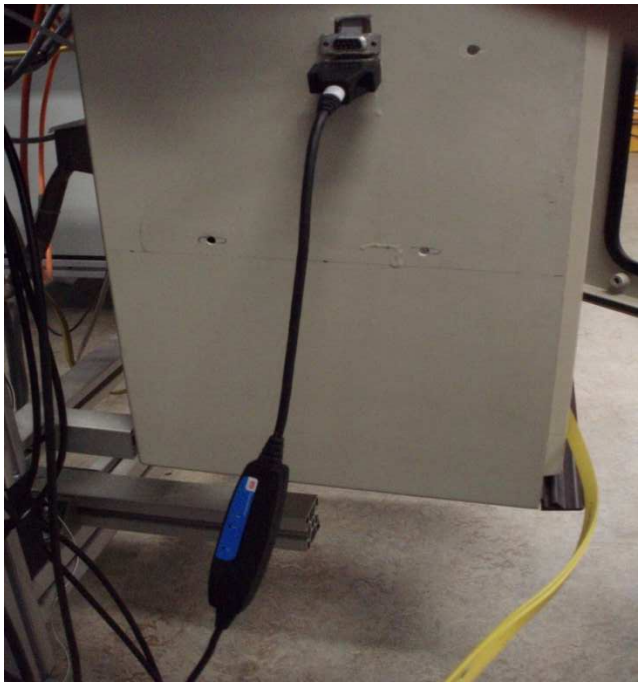
Laitteisto on liitetty ASi-väylään orjien kautta. Paineilmatoimisten laitteiden kanssa käytetään venttiilein varustettuja orjia (Air Box) (kuva 7).

Kuva 7. ASi-väylän orjia

Masterina väylälle on asennettu IFM:n Asi controller (kuva 8). Controllerissa on CANopen-liitäntä, jonka avulla väylää voi hallita PC:n kautta.

Kuva 8. ASi-väylän master

PC on liitetty CANopen-väylään Kvaserin USBcan II -laitteella (kuva 9). Kvaserin T-connector-palikkaa käytettäessä liittimenä ei tarvita erillistä 120 ohmin päätevastusta.



Kuva 9. USBcan II –laite

3 AS-INTERFACE

AS-Interface on standardisoitu kaapelointijärjestelmä kenttätason antureille ja toimilaitteille. AS-Inteface-järjestelmä pienentää kuluja verrattuna perinteisellä kaapelointitavalla tehtyihin järjestelmiin. AS-Interface-järjestelmää käytettäessä perinteinen moninapainen rinnakkaiskaapelointi vaihtuu yhteen parikaapeliin (kuva 10). [1.]

Kuva 10. ASi-väylän kaapeli [2]

3.1 Järjestelmän toiminto

AS-Interface on standardisoitu (AS-i-standard EN50295) kenttäväylä, siihen voidaan liittää antureita sekä toimilaitteita. AS-Interface-väylän kokonaisuuden (kuva 11) muodostavat väylämaster, virtalähde, keltainen tiedonsiirtokaapeli, musta lisätehonsyöttökaapeli ja vähintään yksi kenttälaite. Väylämaster tekee kiertokyselyn orjayksiköille (slaves) ja kerää tulojen tilatiedot ja päivittää lähtöjen tilan. Lähtöjen tilojen päivittäminen saattaa kestää kaksi tai useampia päivityskierroksia.

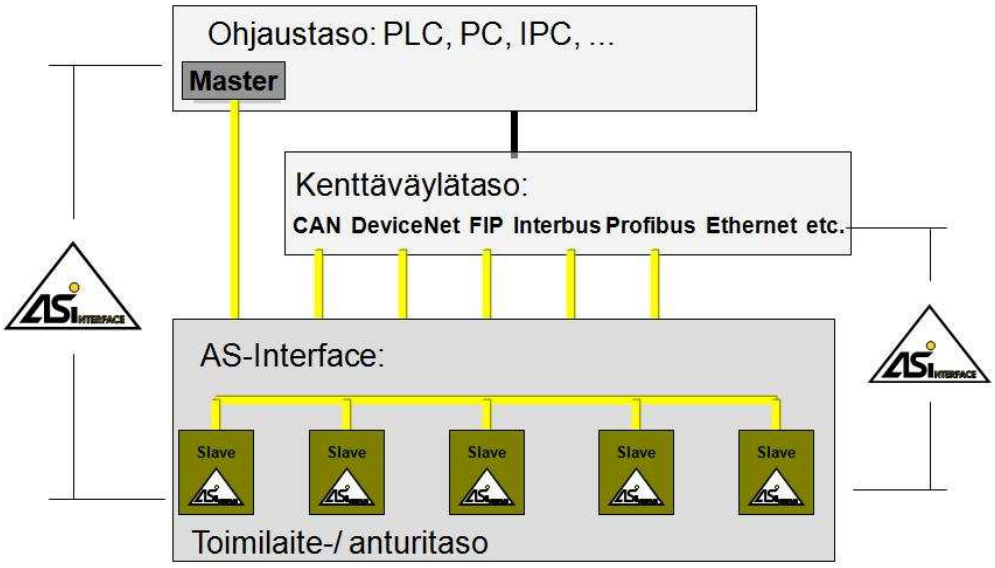
Informaatio ja energia kulkevat samassa kaapelissa. Väylämaster ylläpitää liikennöintiä AS-Interface-väylässä. Väylämaster ja kenttäkomponentit kommunikoivat keskenään keltaisen kaapelin välityksellä, joka välittää myös kenttälaitteiden elektroniikan tarvitseman tehon. Monesti kenttälaitteet kuten binäärilähdöt tarvitsevat lisätehoa, jonka jännitetaso on 24 VDC ja kaapeli on väriltään musta. [1.]

3.2 Vapaasti valittava väylätopologia

AS-Interface-kaapeloinnin rakenteita ovat tähti-, puu- tai linjamuotoinen tai niiden yhdistelmä. Rajoituksena on ainoastaan se, että ei ylitetä väylän ja sen haarojen suurinta sallittua yhteenlaskettua pituutta. Yhden segmentin pituus on 100 m, ja segmenttejä voi olla kaikkiaan kolme. Väylän pituutta voidaan lisätä 300 m:n pituudesta jopa 600 m:iin käyttämällä päätevastusta. [1.]

3.3 AS-Interface -järjestelmän rakenne

AS-Interface-masteriin voidaan liittää 31 tai 62 osoitetta eli orjaa riippuen järjestelmän versiosta. Versioita on käytössä V2.0, V2.1 ja uusimpana V3.0. Versiossa V2.0 kyselyn kierrosaika on 5 ms, suurin osoitemäärä 31, suurin binääritulomäärä on 124 ja suurin binäärilähtömäärä 124. Versioissa V2.1 ja V3.0 puhutaan A- ja B-osoitteista. Versiossa V2.1 kyselyn kierrosaika A/B-osoitteiston ollessa käytössä on 10 ms, suurin osoitemäärä on 61, suurin binääritulomäärä 248 ja binäärilähtömäärä on 186. Versiossa V3.0 kyselyn kierrosaika A/B-osoitteiston ollessa käytössä on binäärituloille 10 ms ja neljälle binäärilähdöille 20 ms. Jos versiossa V3.0 käytetään standardin sallimaa 8I/8O-moduulia, tulee kyselyn kierrosajaksi 50 ms. Lisäksi versio V3.0 sallii tarkan analogiaviestin ja nopeudeltaan 100 baudin sarjaliikenneviestin siirtämisen. Suurin sallittu binääritulojen määrä on 496 ja binäärilähtöjen määrä on 496, kun käytetään 8I/8O-moduulia. [1.]



Kuva 11. ASi-väylän käyttötasot [3]

4 RFID-TUNNISTEET

RFID, joka on lyhenne sanoista Radio frequency identification eli radiotaajuinen etätunnistus. Se on menetelmä tiedon etälukuun ja -tallentamiseen käyttäen RFID-tunnisteita. RFID-tunniste, joka on suomenkieliseltä nimeltään saattomuisti, on pieni laite, joka voidaan sisällyttää tuotteeseen valmistusvaiheessa tai liimata jälkikäteen tarralla. RFID-tunnisteet sisältävät antennin voidakseen lähettää ja vastaanottaa radiotaajuisia kyselyitä RFID-lähetin-vastaanottimelta. [4.]

4.1 RFID-järjestelmän laitteet

RFID-järjestelmiin kuuluu RFID-tunnite, RFID-lukija ja taustajärjestelmä. RFID-tunniste on haluttuun tunnistettavaan kohteeseen kiinnitettävä tarra, kortti, lappu, nappi, implantti tai muu näiden kaltainen. RFID-tunniste sisältää antennin ja sirun, jossa tieto säilytetään. Tunnisteissa on kiinteä sarjanumero ja standardista riippuva määrä vapaata kirjoitustilaa. RFID-lukija on laite jolla tunnisteen sisältöä voidaan lukea ja laitteesta riippuen myös kirjoittaa, ilman että kohteeseen on kontakti tai edes näköyhteys. Lukuetäisyydet riippuvat taajuusalueesta ja standardista. Taustajärjestelmässä tunnistustietoa hyödynnetään prosessitasolla. [4.]

4.2 RFID-tunnisteiden tyypit

RFID-tunnisteet voivat olla joko aktiivisia, passiivisia tai puoli-passiivisia. Passiivisilla RFID-tunnisteilla ei ole omaa virtalähdettä. Laitteen käyttöön vaadittava erittäin pieni sähkövirta induktoituu antenniin saapuvasta radiotaajuisesta skannauksesta, jonka avulla tunniste pystyy lähettämään vastauksen. Virta- ja hintavaatimuksista johtuen passiivisen RFID-tunnisteen vastaus on lyhyt, tyypillisesti ID-numero. Oman virtalähteen puuttuminen tekee laitteesta varsin pienen. [4.]

Puolipassiivinen RFID-tunniste sisältää virtalähteen, mutta ei omaa lähetintä. Omalla virtalähteellä saavutetaan kuitenkin passiivista tunnistetta suurempi toimintasäde ja mahdollistetaan laajennettu toiminnallisuus mukaan lukien tietojen säilyttäminen tunnisteen omassa muistissa.

Aktiiviset RFID-tunnisteet puolestaan sisältävät virtalähteen, ja niillä voi olla pidempi kantomatka, sekä suurempi muisti kuin passiivisilla tunnisteilla. Ne voivat myös tallentaa lähetin-vastaanottimen lähettämiä lisätietoja. [4.]

5 CAN-VÄYLÄ

CAN-väylä (kuva 12), joka on englanniksi Controller Area Network, on automaatioväylä, jota käytetään ajoneuvoissa, koneissa ja teollisuuslaitteissa. CAN-väylässä kaikki liikenne välitetään kaikille moduuleille. Jokaiselle viestille on sanomatunniste. Vastaanottava moduuli päättää tunnisteen perusteella kuuluuko viesti myös sille. Toimintatapa mahdollistaa sen, että sama sanoma, esim. mittausdata ajoneuvon nopeudesta voidaan välittää useammalle laitteelle, kuten nopeusmittarille ja vakionopeuden säädölle yhdellä samalla sanomalla. [5.]

5.1 CAN-väylän johdotus ja liittimet

CAN-väylän johdostus on 2-napainen kierretty parikaapeli. Kaapelia käytetään tiedon siirtoon. ISO 11898 määrittelee CAN-väylällä käytettäväksi kaapelityypiksi impedanssiltaan 120 ohmin suojattua tai suojaamatonta kaapelia. Parikaapelinjohtimista käytetään nimityksiä high ja low. [5.]

CAN-väylän johtojen jännitteet

- Kun CAN-verkossa lähetetään arvo nolla, on high-johdon jännite 3.5 volttia ja low-johdon jännite 1,5 volttia.
- Kun CAN-verkossa lähetetään arvo yksi, on molempien johtojen jännite 2,5 volttia.

[5.]

Käyttäjäorganisaation CiA, joka on lyhenne sanoista CAN in Automation, määrittelemä CAN-väylän liitin on yhdeksännapainen D-liitin (DIN 41652), CiA:n standardi DS-102. [5.]

Väylän ohjaamisessa tarvitaan siis CAN-ohjain, joita on saatavissa nykyään monilta eri valmistajilta kohtuulliseen hintaan. Tämän lisäksi tarvitaan mikrokontrolleri, joka ohjaa koko väylän toimintaa. Ohjainkortti toimii mikrokontrollerin liitäntänä väylälle. Ohjainkortti saa kontrollerilta vertailujännitteen, johon väylän jännitetasoa verrataan tulkittaessa bittien arvoja. [5.]

CAN-väylän toiminta perustuu kahden johtimen, toisin sanoen kelluvan jännitteen periaatteeseen, joka hyödyntää kahden johtimen välistä jännite-eroa. Kaapelointiin indusoituvat häiriöt nostavat tai laskevat kummankin johtimen jännitettä suunnilleen saman verran, jolloin jännite-ero ei juurikaan muutu. Tästä syystä CAN-väylän häiriönsietokyky on kohtuullisen hyvä. [5.]

Väylän päissä käytetään johtimien väliin kytkettyjä ns. terminointivastuksia, joiden tarkoituksena on estää signaalin heijastuminen väylän päädystä takaisin. Tällaisen terminointivastuksen yleinen koko on noin 120 ohmia. [5.]

5.2 CAN-väylän laajennettavuus

CAN-väylällä voisi periaatteessa olla melkein ääretön määrä asemia, mutta käytännössä ohjaimen virransyöttökyky rajoittaa tätä määrää. Yleisimmät ohjaimien tukemat asemamäärät ovat 32 ja 64, mutta eräällä valmistajalla on olemassa myös 110:tä asemaa tukeva ohjain. Mikäli jo olemassa olevaan väylään halutaan liittää uusi asema, se usein vaatii väylän ohjaimen uudelleenohjelmointia. Kuitenkin jos väylälle liitettävä laite on pelkästään vastaanottavaa tyyppiä ja ainoastaan käyttää hyväkseen väylällä jo entuudestaan olevia tietoja, se ei aiheuta mitään muutoksia väylän muuhun osuuteen. [6.]

5.3 CAN-väylän viestien rakenne

CAN-väylä on tehty tiedon sarjamuotoiseen siirtämiseen. Viesteissä on käytössä NRZ-johtokoodi, mikä tarkoittaa sitä että väylän tila on koko bitin esitysajan sama. Näin voidaan käyttää suurempaa taajuutta ja siten suurempaa siirtonopeutta kuin toisella esitystavalla, jossa bitin arvo esitetään keskellä esitettävän bitin aikajaksoa olevilla nousevilla tai laskevilla reunoilla. Kuitenkin tilanteessa, jossa monen peräkkäisen bitin arvo on sama, kellotahdin löytäminen signaalista voi olla vaikeata. Tämän takia johtokoodiin tehdään rike ylimääräisellä välibitillä, mikäli 5 peräkkäistä bittiä ovat arvoltaan samoja. Näillä väliin laitettavilla biteillä kuljetetaan muuta informaatiota, kuten kontrollikenttä, CRC-tarkistussumma ja datakenttä. [6.]

CAN-väylän viestit eivät sisällä lähettäjän eivätkä vastaanottajan osoitetta tai muutakaan tunnistusinformaatiota. Tästä syystä viestien koko on pystytty minimoimaan ja saadaan aikaan suurempi tiedonsiirtokyky. Sen sijaan viestin sisältö, kuten moottorin kierrosluku tai pumpun tilavuusvirta, on otsikoitu koko verkossa yksikäsitteisellä tunnisteella. CAN-väylän ominaisuuksiin kuuluu se, että kaikki väylällä olevat laitteet vastaanottavat viestin ja tekevät sille hyväksyttävyystestin. Mikäli testi osoittaa että viestillä ei ole merkitystä kyseiselle laitteelle, se hylkää sen. Muutoin viesti hyväksytään. [6.]

Myös lähettävä laite tarkkailee itse omaa viestiänsä väylällä. Mikäli se havaitsee bittivirheen tai jonkin muun virheen, se aloittaa viestin lähettämisen uudelleen. Tällä lähettäjän omalla tarkkailulla pystytään havaitsemaan tilanteet, joissa monta asemaa yrittää lähettää väylälle viestiä samanaikaisesti. Näissä tilanteissa ratkaisee niin sanottu priorisointiluku, joka asetetaan kaikille lähettäville asemille. Mitä pienempi priorisointiluku, sitä tärkeämpi viesti on. Törmäystilanteissa pienemmän luvun omaava laite saa lähettää viestinsä ensin ja muut odottavat, että väylä on jälleen vapaasti käytettävissä. [6.]

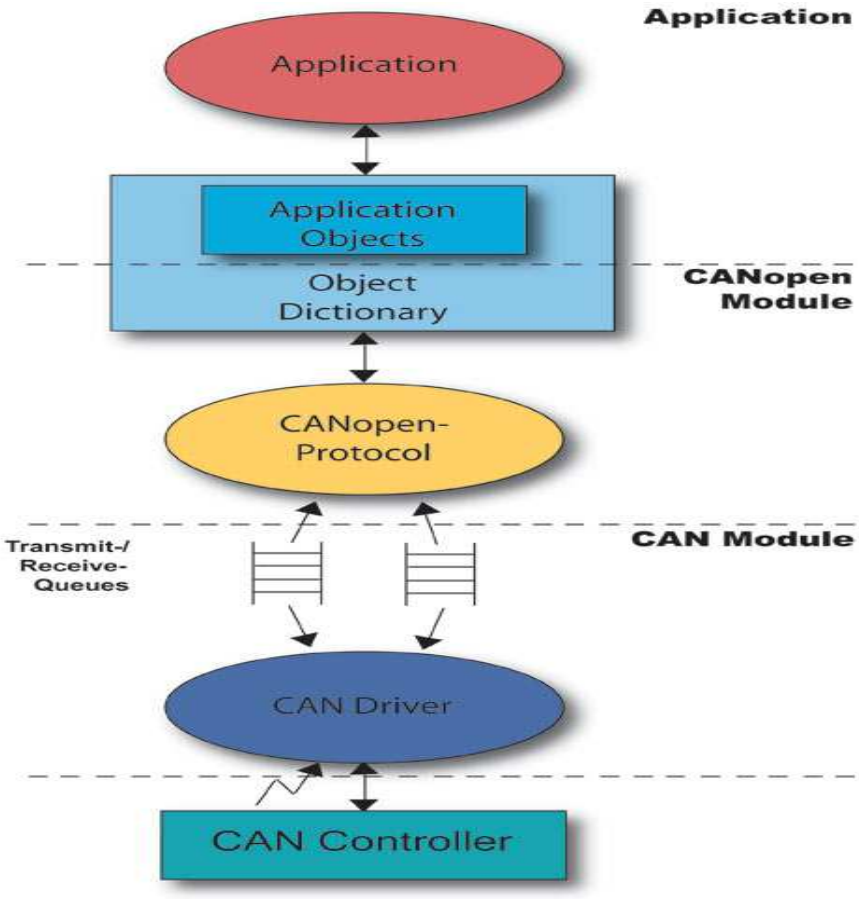
5.4 CAN-väylän ylemmän tason protokollat

CAN-järjestelmissä (kuva 12) on käytössä monia eri protokollia. Tämä johtuu siitä, että eri valmistajat eivät ole päässeet yhteisymmärrykseen siitä mitä protokollalta halutaan, ja siten aihetta koskeva standardointikin on jäänyt tekemättä. Käytössä on kymmeniä erilaisia protokollia. [6.]

Can-järjestelmissä käytettävät yleisimmät protokollat ovat

- CAN Open
- CAN Kingdom
- DeviceNet
  SDS (Smart Distributed System). [4.]

Eri protokollien ominaisuuksissa on suuria eroja, ja niiden soveltuvuus eri käyttökohteisiin vaihtelee suuresti. Vertailtaessa eri protokollia on havaittavissa selkeitä eroja esimerkiksi laajennettavuudessa ja uudelleenkonfiguroinnissa. Mikäli olemassa olevista pitäisi valita jokin yksittäinen standardisoinnin pohjaksi, se saattaisi olla CAN Open. [6.]

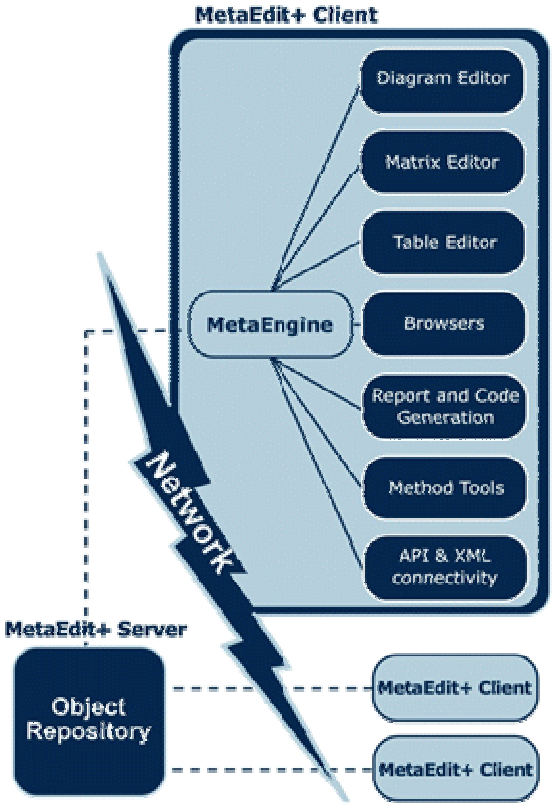


Kuva 12. CAN-väylän rakenne [7]

6 METAEDIT+-OHJELMA

MetaEdit+-ohjelma tarjoaa valikoiman työkaluja (kuva 13) dokumenttien generointiin. Ohjelmassa dataa voi tarkastella graafisena diagrammina, matriisina tai taulukkona. Suunnittelun informaatiota voi selata filttereillä, komponentteja voi hakea, malleja voi linkittää toisiin malleihin ja malleja voi tarkastaa erilaisilla ennalta määrätyillä tai käyttäjän määräämillä raporteilla. Mallinnuksen tuloksen voi julkaista internetiin tai Word-suorittimiin sekä tuottaa koodiksi tuotteelle. [8.]

Kaikki työkalut ovat integroituja objekti varaston kautta, joka ylläpitää ja ajaa työkalujen yhteensopivuutta. MetaEdit+-ohjelma käy joko yhden käyttäjän työaseman käyttöön, tai samanaikaisesti monen työaseman toimeksiannolla serveriin kytkettynä. MetaEdit+-ohjelmaa voi siis käyttää tiimityöskentelyssä. [8.]



Kuva 13. Metaedit+-ohjelman esittely [9]

6.1 Ennalta määrätty mallinnuskielen tuki Metaedit+-ohjelmassa

MetaEdit+-ohjelmaa on sovellettu satoihin aluekohtaisiin mallinnus kieliin, koodigeneraattoreihin ja dokumenttigeneraattoreihin. Monet tavalliset mallinnuskielet tarjotaan MetaEdit+-ohjelman mukana. Ohjelmassa on kirjasto uudelleen käytettäviä mallinnuskielien komponentteja. [8.]

Eri mallinnuskielien käyttäminen MetaEdit+-ohjelmalla on yksinkertaista. Koska kaikki ominaisuudet, jotka eivät ole riippuvaisia tietystä kielestä, pysyvät samoina jatkuvasti. Ei ole tarvetta hankkia monia eri mallinnustyökaluja eri kielille. Tämä säästää kuluja ja resursseja sekä hankinnassa että koulutuksessa. MetaEdit+-ohjelma tarjoaa siirtymistien eri kielten välillä ja kaavoja. Ohjelma voi tukea moninkertaisia mallinnuskieliä samanaikaisesti. Samalla tavoin monikieliset ominaisuudet mahdollistavat linkittämisen ja uudelleen käytön yli eri mallinnuskielien, ylläpitäen tiedonkulun niiden välillä. [8.]

MetaEdit+-ohjelma tukee automaattista koodin generointia ennalta määrätyille ja käyttäjän määräämille ohjelmointikielille. Mahdollisuudet generoida koodi automaattisesti riippuu käytettävästä mallinnuskielestä sekä kohteena olevasta ohjelmointikielestä. [8.]

Ennalta määrättyjä koodigeneraattoreita on saatavilla seuraaville ohjelmointikielille:

- Smalltalk
- C++
- Java
- Delphi (Object Pascal)
- SQL
- COBRA IDL. [8.]

6.2 Automaattinen dokumenttien generointi

MetaEdit+-ohjelman mukana tulee joukko dokumentaatioraportteja, joilla voi luoda internetdokumentteja ja Microsoft Word -prosessointiformaatteja helposti. MetaEdit+-ohjelma tarjoaa ennalta rakennettuja raportteja mallien analysoimiseksi ja tarkistamiseksi. [8.]

MetaEdit+-ohjelma tarjoaa dokumentaatioita seuraaville alustoille:

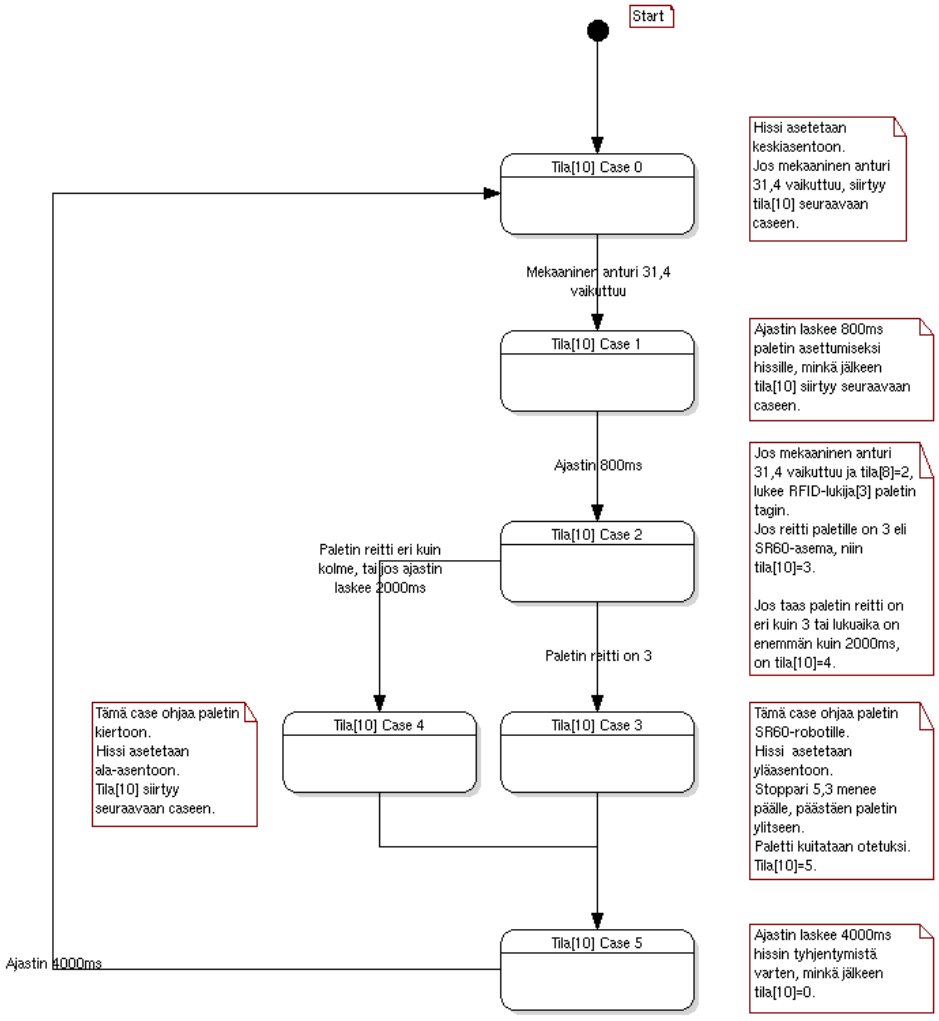
- Word
- RTF
- HTML
- XML

XMI. [8.]

MetaEdit+-ohjelmassa ei tarvitse tehdä manuaalista dokumentointia eikä päivittämätön dokumentointi ole ongelmana. Projektin dokumentointi sekä graafinen dokumentointi raporteiksi sisältävät HTML-dokumentoinnin, joka muuttaa koko projektin HTML-tiedostoiksi, joissa kuvat ovat GIF-muodossa. Ohjelma omaa myös Word-dokumentoinnin, joka muuttaa koko projektin doc-tiedostoksi vektorigraafisina kuvina. [8.]

6.3 Tilakoneiden graafisen esityksen mallit

Tilakoneiden graafiseen esitykseen (kuva 14) käytettiin MetaEdit+-ohjelman tilakoneiden esitystä varten suunniteltua työkalua. Tilakoneet tuli esittää graafisesti, jotta tilakoneiden rakennetta voidaan esittää opiskelijoille yksinkertaisessa muodossa. Tilakoneet esitetään rakennepuuna, jossa jokaisesta tilakoneen casea kuvaa oma laatikkonsa. Casejen väliset siirtymät on kuvattu nuolilla, ja siirtymän aktivoiva toiminto on kirjoitettuna nuolen kohdalle. Jokaiselle caselle on luotu kommentti casen viereen, jossa kerrotaan sen tekemät toiminnot. Start alussa kuvaa casen aloitusta eli ohjelman käynnistystä. Joissain tilakoneissa ei palata tilakoneen lopusta tilakoneen alkuun, vaan mahdolliset jäämäpalettien poiston caset tilakoneen alussa jäävät pois käytöstä normaaliajon aikana. Tällöin palataan tilakoneen lopusta tilakoneen ensimmäiseen toiminta-ajon caseen.



Kuva 14. Esimerkki tilakoneen esityksestä

7 TILAKONEOHJELMAN TOIMINTAPERIAATE

Alkuperäinen ohjelma oli Juha-Matti Äyvärin tekemästä opinnäytetyöstä vuodelta 2007. [10.] Ohjelmassa oli puutteita, jotka estivät linjaston käytön opetuksessa. Ongelmana olivat paletit, jotka olivat edellisen ajon päätyttyä jääneet sattumanvaraisesti linjaston eri kohtiin. Paletit jäivät ajon alussa linjaston eri kohtiin jumiin, koska linjasto ei havainnut niitä. Ohjelma ei myöskään ottanut huomioon paletteja tietyissä linjaston risteyksissä tai muissa linjaston eri paikoissa.

Perustoiminnaltaan ohjelma perustuu C++-ohjelmointikielen tilakonerakenteeseen. Toimilaitteita eli linjan moottoreita, risteyksien hissejä ja pysäyttimiä ohjataan nodeon ja nodeoff -komennoilla (kuva 15). Induktiivisten antureiden sekä mekaanisten kytkimien tilaa havainnoidaan nodestate-komennolla. Ohjelmassa käytetään timedelay-funktiota ajastimena.

```
switch(tila[9]) {
    case 0:
        nodeoff(1,14,2);    //Hissi 14,2 asetetaan ala-asentoon.
    break;

    case 1:
        strcpy(tilan_viesti[9],"13.2, tila[15]");
        if(nodestate(1,13,2) && tila[15]==0)
        //Jos mekaaninen anturi 13,2 vaikuttuu ja tila[15]=0, siirtyy tila[9] seuraavaan caseen.
            tila[9]++;
    break;

    case 2:
        strcpy(tilan_viesti[9],"ajastin 1500ms");
        nodeoff(1,14,2);    //Hissi 14,2 asetetaan ala-asentoon.
        if(timedelay(9,2000000)){ //Ajastin laskee 2000ms, jonka jälkeen tila[9]=0.
            tila[9]=0;
        }
    break;
}
```

Kuva 15. Esimerkki ohjelman tilakoneesta

Palettien reitit voidaan asettaa ohjelman lopussa reitinasetus-funktioon neljällä eri arvolla: 0, 1, 2 ja 3. Näistä arvo 0 on kierto linjaston keskellä, arvo 1 on manuaalityöasema, arvo 2 SR6-työasema ja arvo 3 SR60-työasema (taulukko 1). Lisäksi voidaan määrittää kierrosten määrä tietylle reitin arvolle, jolloin reitti kierretään halutun määrän verran kierroksia.

Taulukko 1. Työasemien numerointi

Numero	Työasema
1	Manuaaliasema
2	SR6-asema
3	SR60-asema

7.1 Kehittäminen

Linjaston laitteiden ohjaus, oli jaettu 19 eri tilakoneeseen liitteen 1 mukaisesti. Tein sekä muutoksia että lisäyksiä alkuperäiseen tilakoneohjelmaan, jotta linjasto lähtisi käynnistettäessä sulavasti tekemään työkiertoa. Edelliseltä ajokerralta eri linjaston pisteisiin jääneet paletit, oli saatava kiertoon ajon alussa. Tein ajon alussa toimivia erillisiä jäämäpalettien poistoa varten toteutettuja tilakoneita alkuperäisten tilakoneiden sisään sekä uusia caseja poistamaan jäämäpaletteja. Nämä tilakoneet ja caset ovat käynnissä alle minuutin ajan ajon alussa, minä aikana paletit ehtivät kiertoon eri pisteistä ja normaali työkierto voi lähteä käyntiin.

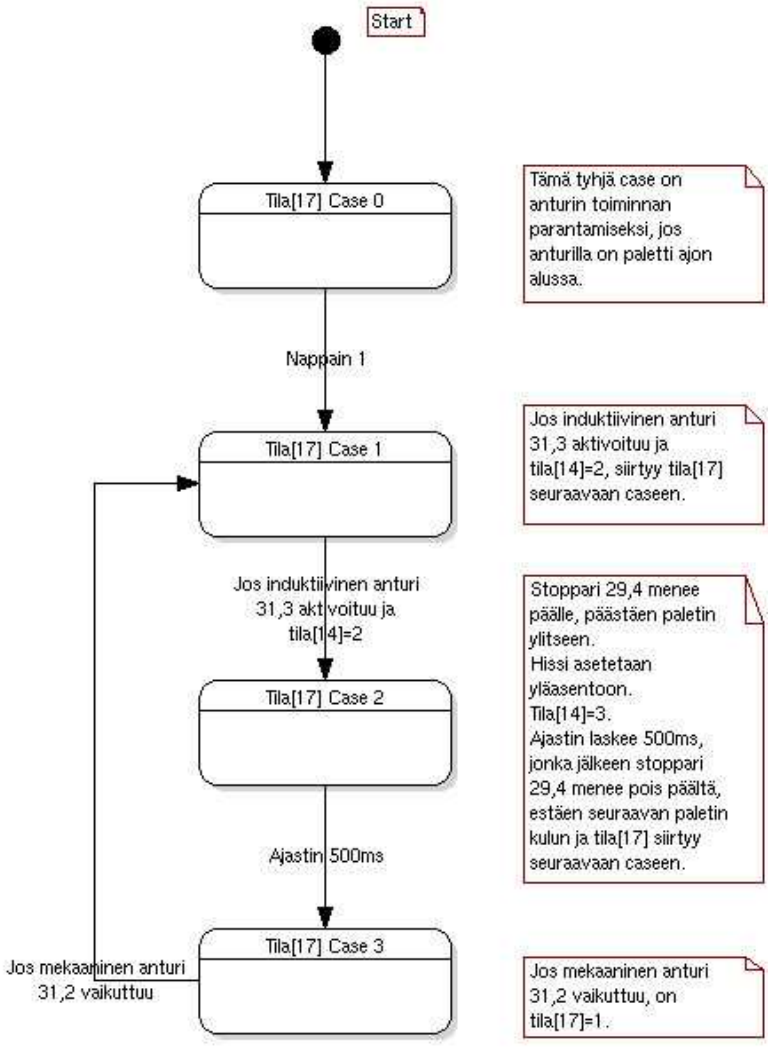
Nämä tilakoneiden alkuun lisätyt caset sekä sisäkkäiset tilakoneet, joita lisättiin neljään eri tilakoneeseen numeroltaan: 7, 8, 16 ja 18, siis ajavat tyhjäksi linjaston paleteista. Tyhjennyksen aikana paletit poistuvat asemilta ja siirtyvät kiertämään linjaston keskellä. Linjaston keskeltä paletit siirtyvät työpisteille ohjelman jatkuessa, ohjelmassa annetun aseman arvon mukaisesti. Monessa tilakoneessa on aluksi tyhjä Case 0, jonka tehtävänä on parantaa tilakoneessa olevien anturien toimintaa. Tämä johtuu havainnostani, että linjaston käynnistyttyä paletin ollessa anturilla ennen käynnistystä, ei anturi havaitse palettia. Kokeilemalla havaitsin anturin huomaavan paletin pienellä muutoksella. Muutoksessa tilakoneessa on tyhjä case ensimmäisenä case-

na, josta siirrytään seuraavaan caseen ohjelman alustuksessa. Näin anturi havaitsee paletin normaalisti eikä paletti jää jumiin. Nämä tyhjät caset jäävät pois normaaliajon aika käytöstä, kuten jäävät kaikki jäämäpalettien poistoa varten luodut caset ja sisäkkäiset tilakoneet.

7.2 Muutostyöt

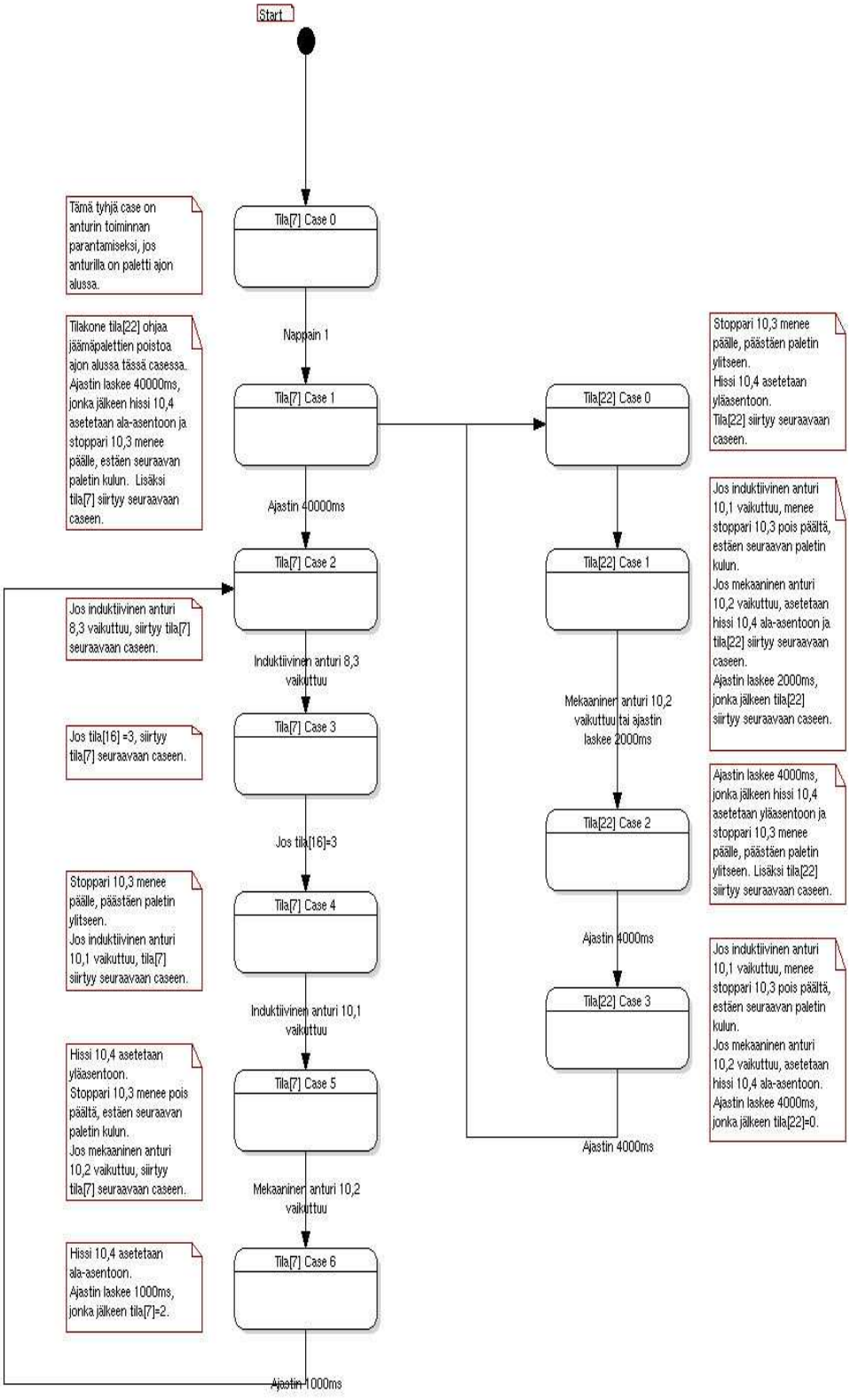
Tilakoneiden muutostyöt tapahtuivat testaamalla ja etsimällä puutteellisia ohjelman kohtia. Testaamisessa oli otettava huomioon kaikki mahdolliset tilanteet, joissa paletit voivat sijaita ajon alussa. Varsinkin risteyskohtiin oli paneuduttava, sillä niissä palettien saapumisjärjestyksen kanssa oli ongelmia. Myös linjaston kohdat, joissa ei ollut anturia havaitsemassa mahdollista jäämäpalettia, piti kiinnittää erityistä huomiota. Ratkaisin tällaisia tilanteita muun muassa ajastimella, joka tyhjentää ajon alussa linjan tyhjäksi kaikista paleteista, jolloin anturittomat linjan kohdat tulevat myös tyhjiksi. Joillekin hisseille oli luotava tällaisessa tyhjennystilassa oma tilakoneensa ohjaamaan pelkästään jäämäpalettien tyhjennystä ajon alussa. Loin nämä tilakoneet normaaliajon tilakoneiden jäämäpalettien poiston casejen sisään. Nämä jäämäpalettien poiston caset toimivat ajon alussa jäämäpalettien poistoon tarvittavan ajan, jonka jälkeen tilakone siirtyy normaaliajoon, eikä enää käytä kyseisiä caseja.

Risteyksissä oli ongelmana paletin jääminen risteyksen hissiin jumiin, koska stoppari ja hissi eivät olleet ajoitettu keskenään riittävän hyvin. Muuttamalla hissin ja stopparin aktivoitumisessa olevia viiveitä, sain hissien ja stoppareiden keskinäisen toiminnan sulavaksi. Myös risteyskohtiin tuli tehdä muutoksia. Niissä paletit helposti törmäävät eri suunnista tullessaan, jos risteykseen tulevien linjojen tilakoneet eivät toimi hyvin keskenään. Esimerkiksi kuvassa 15 on kuvattu Tilakone 17:n rakenne Metaedit+-ohjelmalla. Kyseisen tilakone ohjaa risteystä edeltävää linjan osaa. Sen Case 1:ssä on kommentissa merkintä, että odotetaan risteystä ohjaavan Tilakone 14:n olevan Case 2:ssa ennen kuin tilakone jatkaa eteenpäin. Näin tilakoneet linkittyvät toisiinsa, jotta linjastossa ei tapahtuisi törmäyksiä.



Kuva 16. Tilakone 17

Kuvassa 16 on esitetty Tilakoneen 7 rakenne. Case 0 on tyhjä case, josta siirrytään ohjelman alustuksessa seuraavan caseen. Case 1 on jäämäpalettien poistoa varten tehty case. Tässä Case 1:ssä sijaitsee sisäkkäinen tilakone, Tilakone 22, joka toimii vain jäämäpalettien poiston ajan. Tämä tilakone tyhjentää useamman kuin yhden jäämäpaletin, mikäli se on tarpeen. Case 1:lle on annettu toiminta-aika, missä ajassa kaikki tämän tilakoneen vaikutusalueella mahdollisesti olevat paletit ehditään tyhjentää linjalta. Kun ajastin on kulunut loppuun, siirrytään Case 2:een. Case 2 kuuluu ohjelman normaaliajoon, johon palataan aina tilakoneen suoritettua vaiheensa loppuun. Näin Case 0 ja Case 1 ovat poissa käytöstä normaaliajon aikana. Linjastossa molemmat robottiasemat eli asemat 2 ja 3 muistuttavat hyvin paljon toisiaan, kuitenkin aseman 3 ollessa hieman pidempi fyysisesti. Tällöin käytin samankaltaisia ratkaisuita kummassakin asemassa, mikä osaltaan helpotti työtä, koska ei ollut tarvetta keksiä erilaista ratkaisua molemmille asemille.



Kuva 17. Tilakone 14

Manuaaliaseman toteutus poikkesi kahden robottiaseman toteutuksesta täysin. Manuaaliaseman rakenne oli erilainen kuin robottiasemat, sillä siinä linjan pituus oli muihin asemiin verrattuna lyhyempi. Mutta toisaalta manuaaliasemassa oli enemmän tilakoneita, antureita ja toimilaitteita suhteessa linjan pituuteen kuin muualla linjassa. Löytämällä oikeanlainen tilakoneiden, hissien, antureiden, kytkinten ja stoppareiden yhteistoiminta saatiin jäämäpaletit tyhjennettyä asemalta. Toteutuksessa suljettiin tie saapuvilta uusilta paleteilta. Eli kierrossa olevilta paleteilta estettiin tulo manuaaliaseman alueelle siksi aikaa, kunnes manuaaliaseman tyhjentämiseen kuluva aika on mennyt umpeen, minkä jälkeen manuaaliaseman laitteet aloittavat normaalin työkierron.

Manuaaliaseman toiminnan ohjauksen kehittäminen vähemmillä tilakoneilla toimivaksi oli harkinnassa. Kuitenkin voidaan todeta, että usean lukumäärältään pieniä laitteita ohjaavien tilakoneiden rypäs ei käytännössä ole monimutkaisempi, kuin jos tilakoneita liitettäisiin toisiinsa suuremmaksi kokonaisuudeksi. Lisäksi aseman oli toimittava hyvin erilaisissa palettien sijaintien tilanteissa. Näin voidaan todeta, että yksittäiset melko pienten alueiden toimintaa hallitsevat tilakoneet ovat parempi vaihtoehto, kuin ohjaus yhdellä laajemmalla ja monimutkaisemmalla tilakoneella, sillä tällöin aseman muuttuvan käytön vuoksi tarpeelliset muutokset ovat helpompia tehdä.

8 YHTEENVETO

Opinnäytetyössä oli tarkoituksena kehittää alkuperäistä MTS2-kokoonpanolinjastolle tehtyä tilakoneohjelmaa. Tämän alkuperäisen tilakoneohjelman rakenteessa oli puutteita ja virheitä, joiden vuoksi linjastoa ei voitu käyttää opiskelijakäytössä. Linjaston toiminnan kannalta ongelmana olivat paletit, jota olivat jääneet edelliseltä ajokerralta linjaston eri kohtiin sekä tilanteet, joissa paletit jäivät jumiin linjastossa. Lisäksi opinnäytetyön kirjallisessa osiossa esitellään linjaston toimintaan perustuva tekniikka kuten toimilaitteet, anturit, tunnisteet ja kenttäväylät.

Ohjelman puutteiden aiheuttamia ongelmia lähdettiin ratkaisemaan tutustumalla ohjelmaan ja sen tilakoneiden toimintaan ja toteutukseen. Tämä suoritettiin testaamalla ja kokeilemalla erilaisia vaihtoehtoisia toteutusmalleja. Jäämäpalettien ongelman ratkaisemiseksi alkuperäiseen tilakonerakenteeseen tehtiin caseja, sisäkkäisiä tilakoneita ja muutoksia alkuperäiseen ohjelmaan. Lisäysten ja muutosten jälkeen linjasto ajaa jäämäpaletit pois linjaston eri kohdista kiertoon ajon alussa. Kun tietty aika on kulunut ja kaikki jäämäpaletit on saatu kiertoon, aloittaa ohjelma normaalin työkierron, jonka jälkeen jäämäpalettien poistoa varten luodut caset ja tilakoneet jäävät pois käytöstä.

Toinen insinöörityön tehtävä oli luoda ohjelmasta graafisesti esitettävä malli. Tässä käytettiin Metaedit+-ohjelmaa, joka oli asennettuna koulun automaatiolaboratorion tietokoneille. Metaedit+-ohjelmalla luotiin jokaisesta tilakoneesta graafinen rakennepuu, jossa jokaisesta casesta on oma casea kuvaava laatikko. Caseja liittävät niitä aktivoivat toiminnot, joita kuvataan nuolilla. Lisäksi kaikille caseille on selittävä laatikko, jossa kerrotaan mitä casessa tapahtuu.

Opinnäytetyön tulos täytti asetetut vaatimukset. Linjaston toiminta muuttui toimivammaksi, ja linjasto kykenee itsenäisesti käynnistymään, vaikka linjastolle olisi jäänyt paletteja aiemmalta ajokerralta. Lisäksi ongelmat linjaston normaaliajon toiminnassa poistuivat työn ansiosta. Graafinen esitys tilakoneiden toiminnasta mahdollistaa linjaston toiminnan esityksen opiskelijoille, ja se on helposti omaksuttavassa muodossa.

VIITELUETTELO

[1] Siemens Osakeyhtiö, AS Interface, [verkkodokumentti, viitattu 22.2.2008]. Saatavissa: http://www.siemens.fi/CMSTeollisuus.nsf/all/637C2A301F79A7BCC22570A400437306?opendocument&expand=1

[2] Fujielectric, AS-Interface [verkkodokumentti, viitattu 14.4.2008]. Saatavissa: http://www-.fujielectric.co.jp/fcs/jpn/as-interface/seihin/cable/img/photo_1.jpg

[3] AS-Iterface (PowerPoint esitys 0,9mb), [verkkodokumentti, viitattu 14.4.2008]. Saatavissa: http://opetus.stadia.fi/koneauto-maatiolaboratorio/kenttavaylat/Asi.htm

[4] RFIDLab Finland, RFID-tekniikan perusteet. 21.8.2007 [verkkodokumentti, viitattu 22.3.2008]. Saatavissa: http://rfidlab.fi > RFID-tekniikan perusteet

[5] CAN in Automation [verkkodokumenttii, viitattu 30.4.2008]. Saatavissa: http://www.can-cia.de/

[6] Siltanen, Vesa, CAN-väylä, [verkkodokumentti, viitattu 22.4.2008]. Saatavissa: http://www.fhpa.fi/fpf/arkisto/_seminaarit/index.php?p=seminaarit&l=20011114_can_vayla

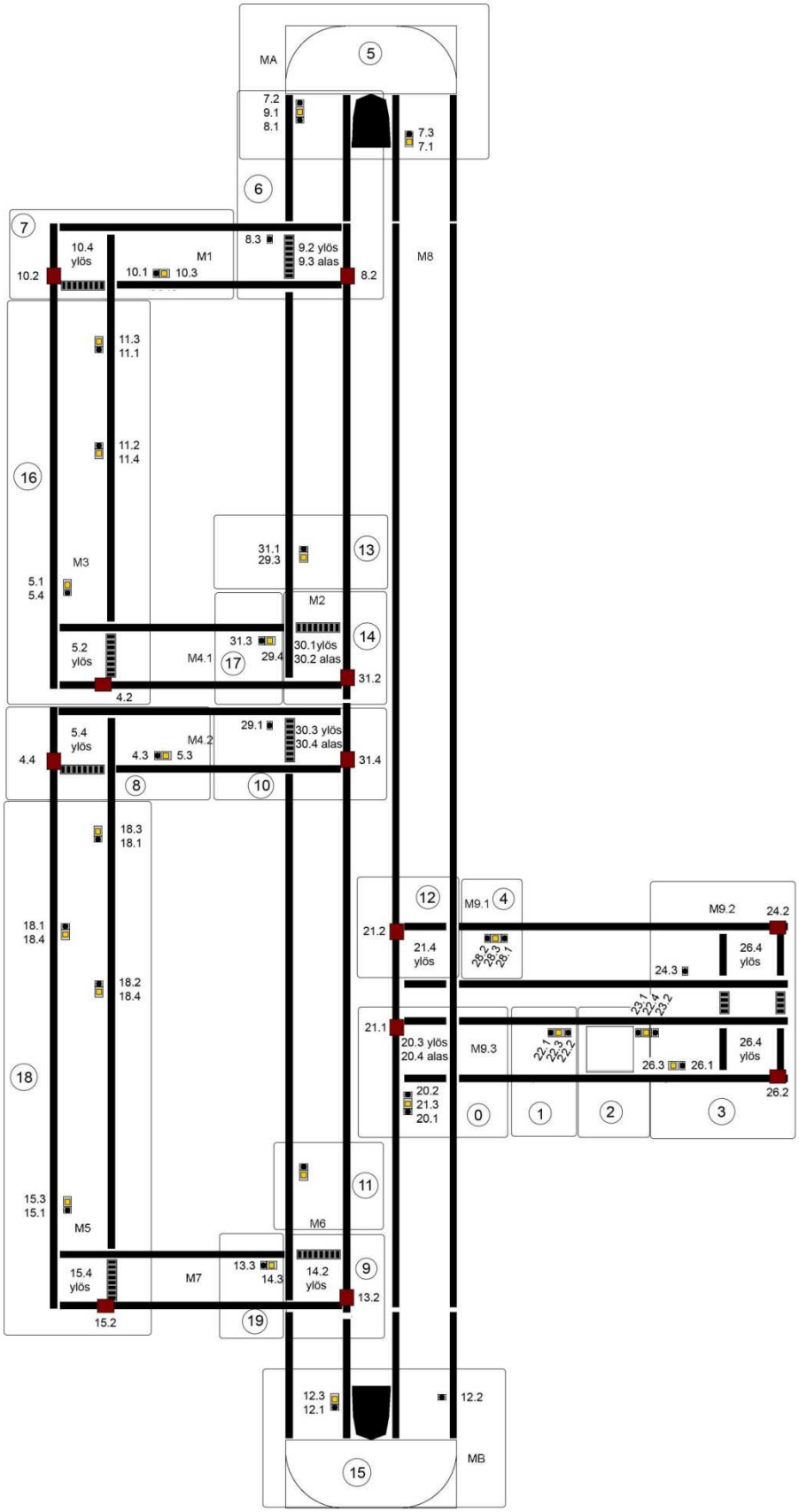
[7] Embedded Access Inc [verkkodokumentti, viitattu 14.4.2008]. Saatavissa: http://www.embedded-access.com/images/product_images/canopen.jpg

[8] Metacase, Metaedit+ [verkkodokumentti, viitattu 24.3.2008]. Saatavissa: http://www.metacase.com

[9] Metacase, Metaedit+ [verkkodokumentti, viitattu 14.4.2008]. Saatavissa: http://www.metacase.com

[10] Äyväri Juha-Matti, *MTS2-kokoonpanolinjan linux-pohjainen ohjausjärjestelmä*. Insinöörityö. Helsingin ammattikorkeakoulu. Kone- ja tuotantotekniikan koulutusohjelma. Helsinki. 2007.

Linjaston kartta



Manuaalilinjan ASi-väylä

Manuaalilinjan Asi-väylä

I/O	Slave numero	Nimi kentällä	Osoite
input	20	S201	20.1
input	20	S202	20.2
output	20	Y203	20.3
output	20	Y204	20.4
input	21	S211	21.1
input	21	S212	21.2
output	21	Y213	21.3
output	21	Y214	21.4
input	22	S221	22.1
input	22	S222	22.2
output	22	Y223	22.3
output	22	Y224	22.4
input	23	S231	23.1
input	23	S232	23.2
output	23	Y233	23.3
output	23	Y234	23.4
input	24	S241	24.1
input	24	S242	24.2
input	24	S243	24.3
input	26	S261	26.1
input	26	S262	26.2
output	26	Y262	26.3
output	26	Y264	26.4
input	28	S281	28.1
input	28	S282	28.2
output	28	Y283	28.3
PAINONAPPI			
input	25	schlecht	25,1
input	25	gut	25,2
input	25	auto/man	25,3
MOOTTORIT			
output	17	M9.1, M9.2, M9.3	17,1

SR6-linjan ASi-väylä

SR6-linjan Asi-väylä

I/O	Slave numero	Nimi kentällä	Osoite
input	7	S62	7,1
input	7	S63	7,2
output	7	Y61	7,3
input	8	S12	8,1
input	8	S23	8,2
input	8	S24	8,3
output	9	Y11	9,1
output	9	Y22	9,2
output	9	Y23	9,3
input	10	S42	10,1
input	10	S43	10,2
output	10	Y41	10,3
output	10	Y42	10,4
input	11	S52	11,1
input	11	S53	11,2
output	11	Y51	11,3
output	11	Y52	11,4
MOOTTORIT			
output	27	M1	27,1
output	27	MA, M8	27,2

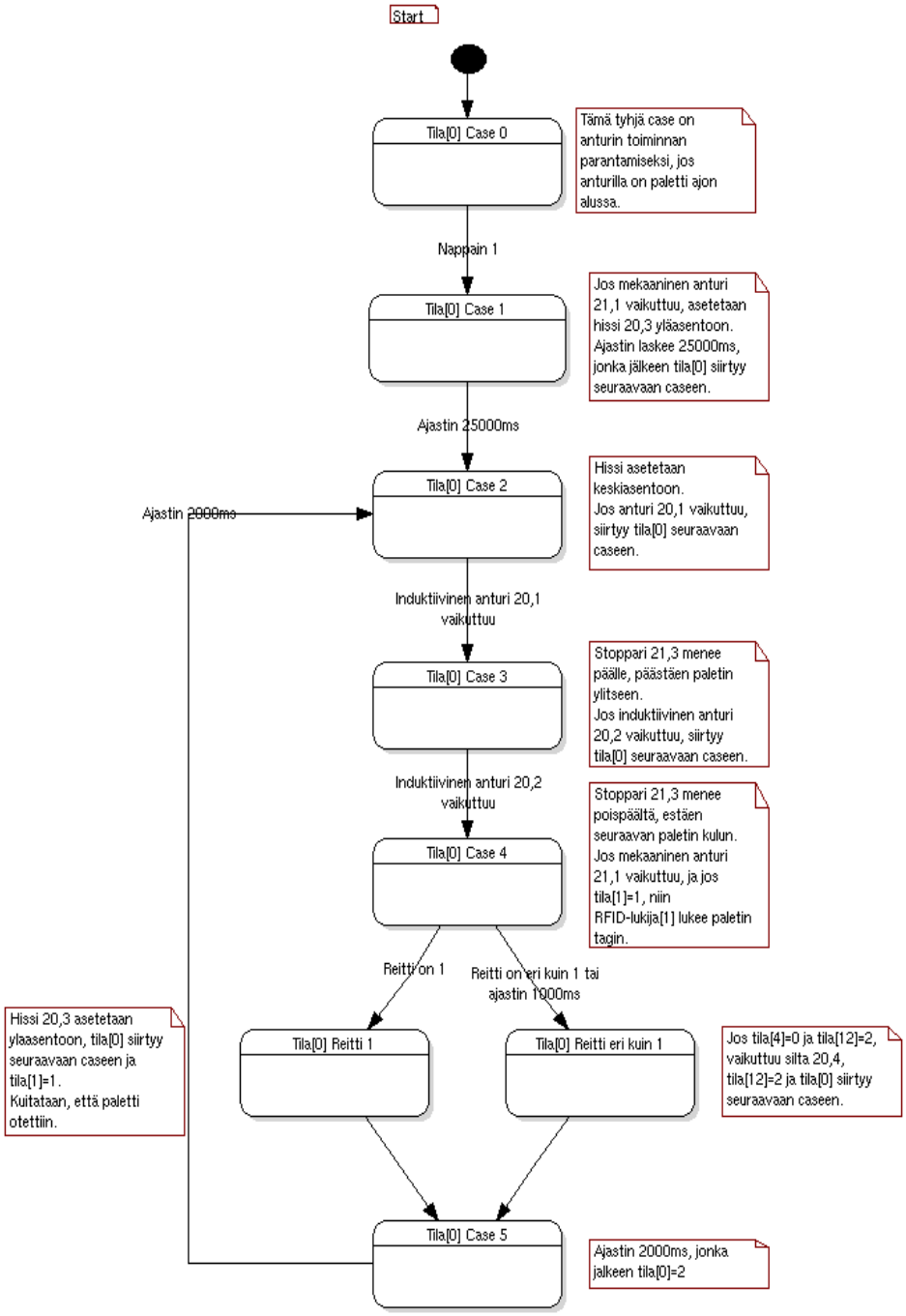
SR60-linjan ASi-väylä

SR60-linjan Asi-väylä

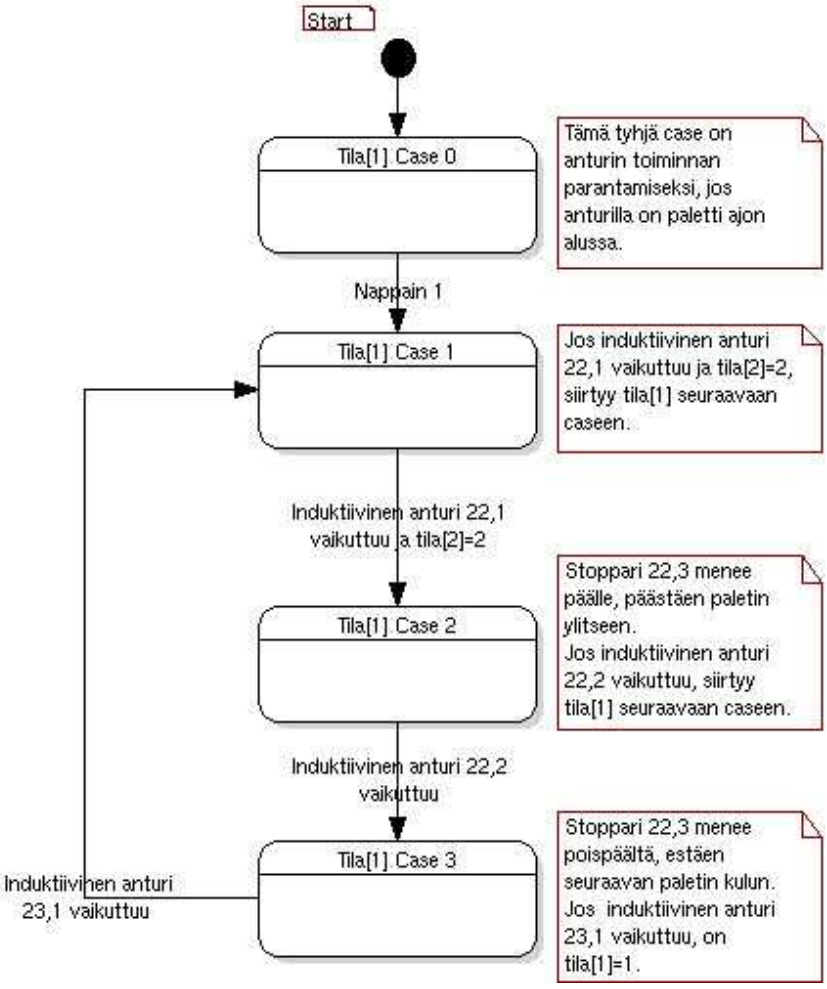
I/O	Slave numero	Nimi kentällä	Osoite
input	4	S32	4,1
input	4	S33	4,2
input	4	S42	4,3
input	4	S43	4,4
output	5	Y31	5,1
output	5	Y32	5,2
output	5	Y41	5,3
output	5	Y42	5,4
input	12	S62	12,1
input	12	S63	12,2
output	12	Y61	12,3
input	13	S11	13,1
input	13	S13	13,2
input	13	S15	13,3
output	14	Y11	14,1
output	14	Y12	14,2
output	14	Y15	14,3
input	15	S32	15,1
input	15	S33	15,2
output	15	Y31	15,3
output	15	Y32	15,4
input	18	S52	18,1
input	18	S53	18,2
output	18	Y51	18,3
output	18	Y52	18,4
input	2	S24	29,1
output	2	Y11	29,3
output	2	Y15	29,4
output	3	Y12	30,1
output	3	Y13	30,2
output	3	Y22	30,3
output	3	Y23	30,4
input	1	S11	31,1
input	1	S13	31,2
input	1	S15	31,3
input	1	S23	31,4
PAINONAPPI			
input	19	schlecht	19,1
input	19	gut	19,2
MOOTTORIT			
output	6	M3, M4.1	6,1
output	6	M4.2	6,2
output	6	M2	6,3
output	16	M5, M7	16,1
output	16	M6, MB	16,2

Tilakonekaavioiden kuvat

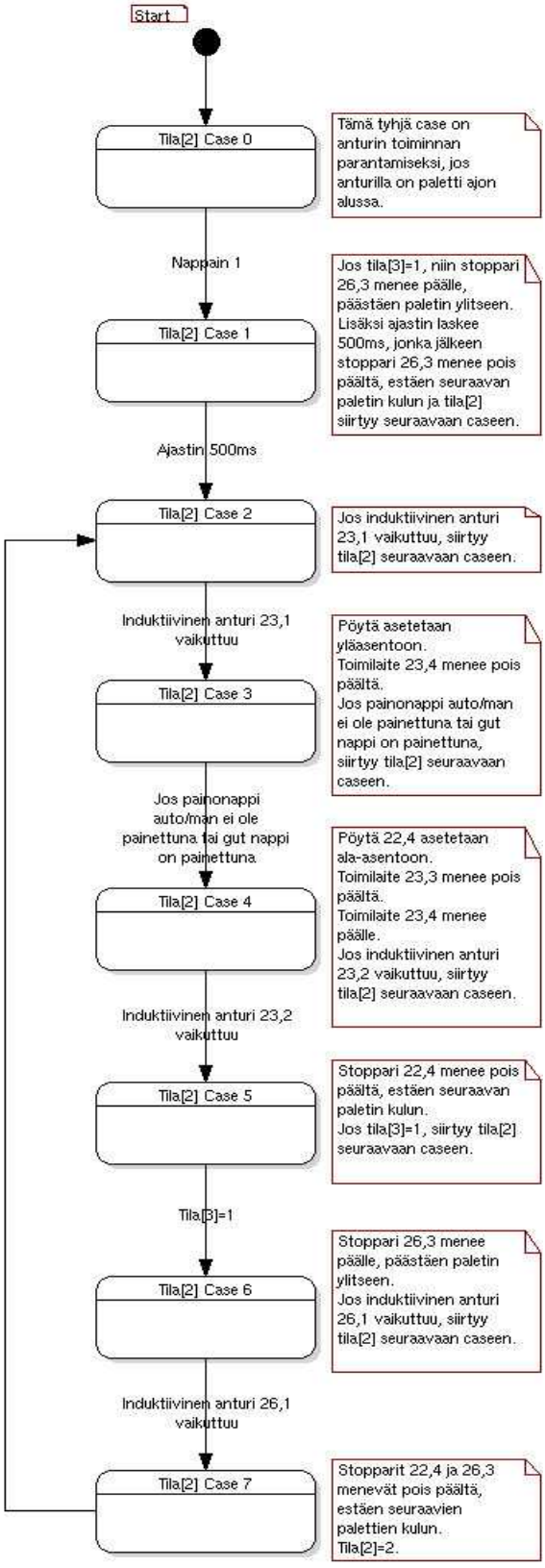
Tilakone 0 kaavio



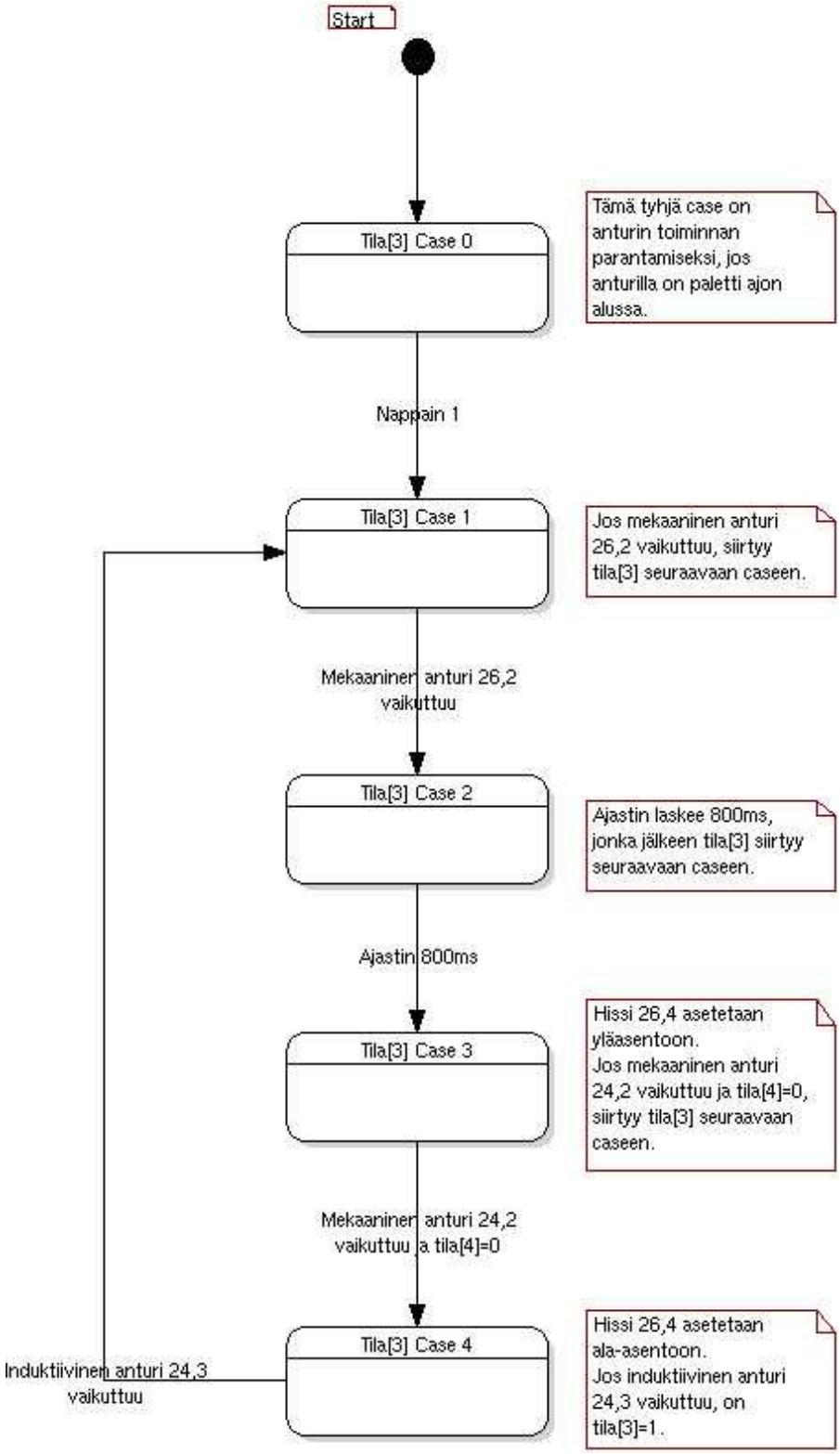
Tilakone 1 kaavio



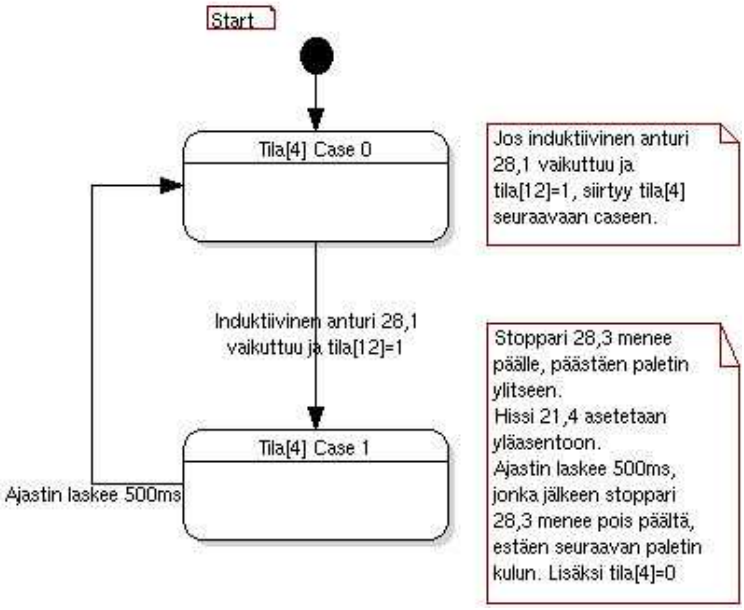
Tilakone 2 kaavio



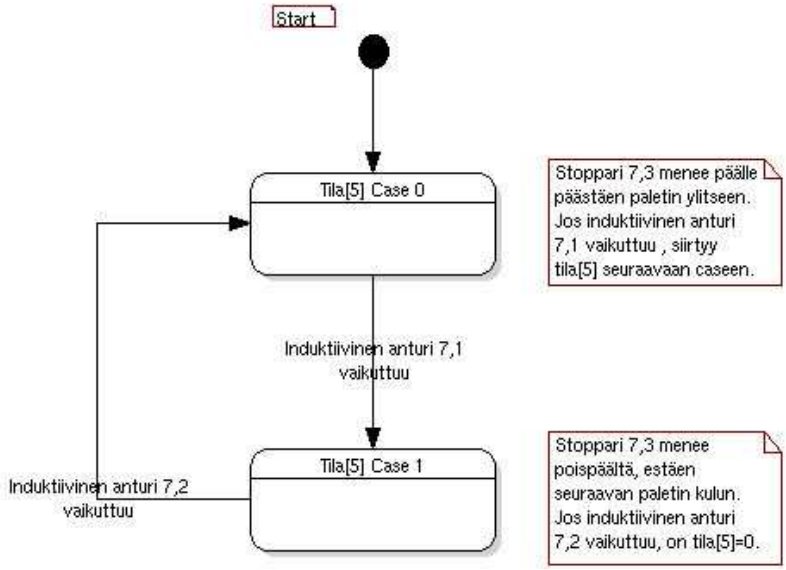
Tilakone 3 kaavio



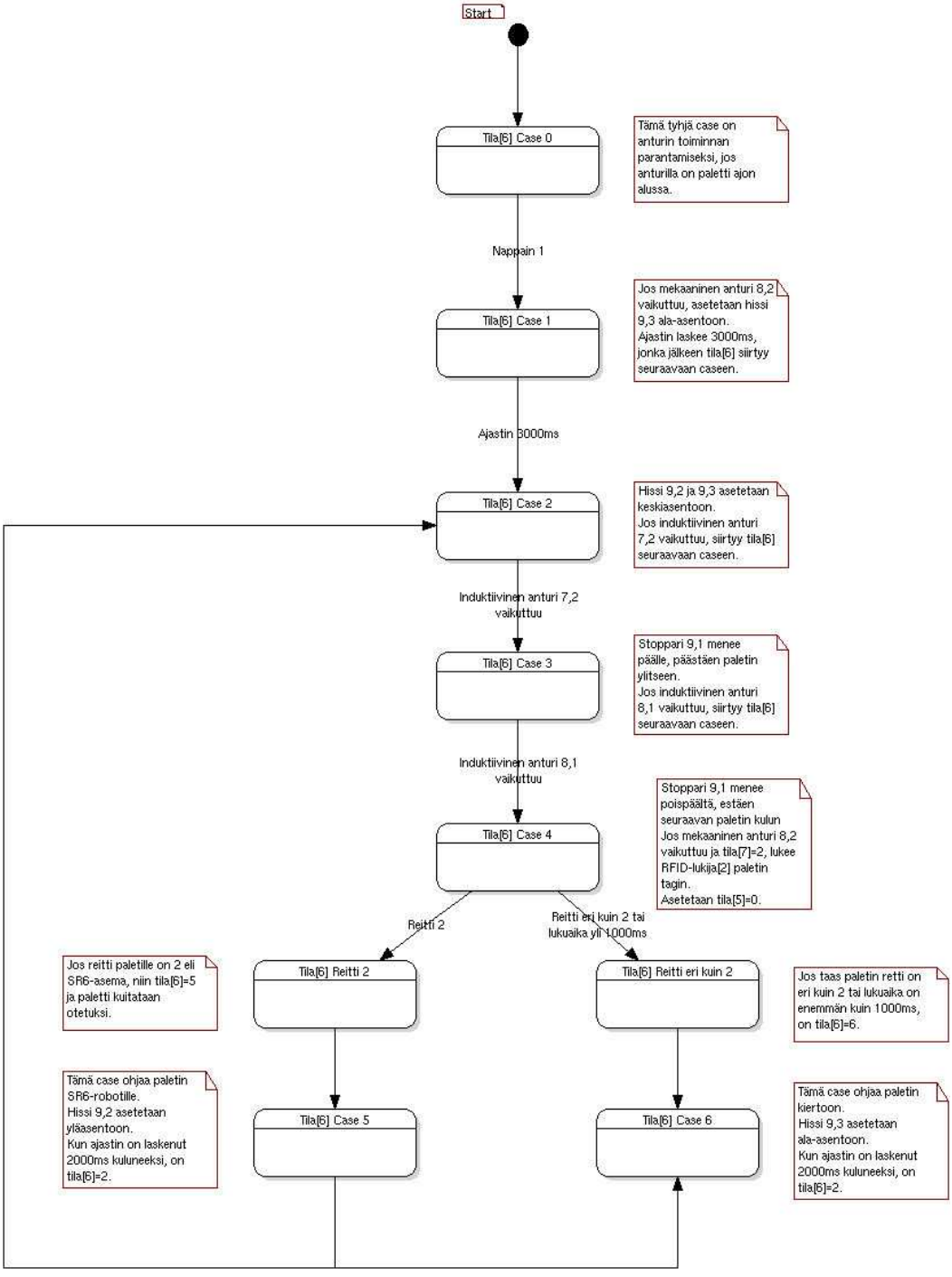
Tilakone 4 kaavio



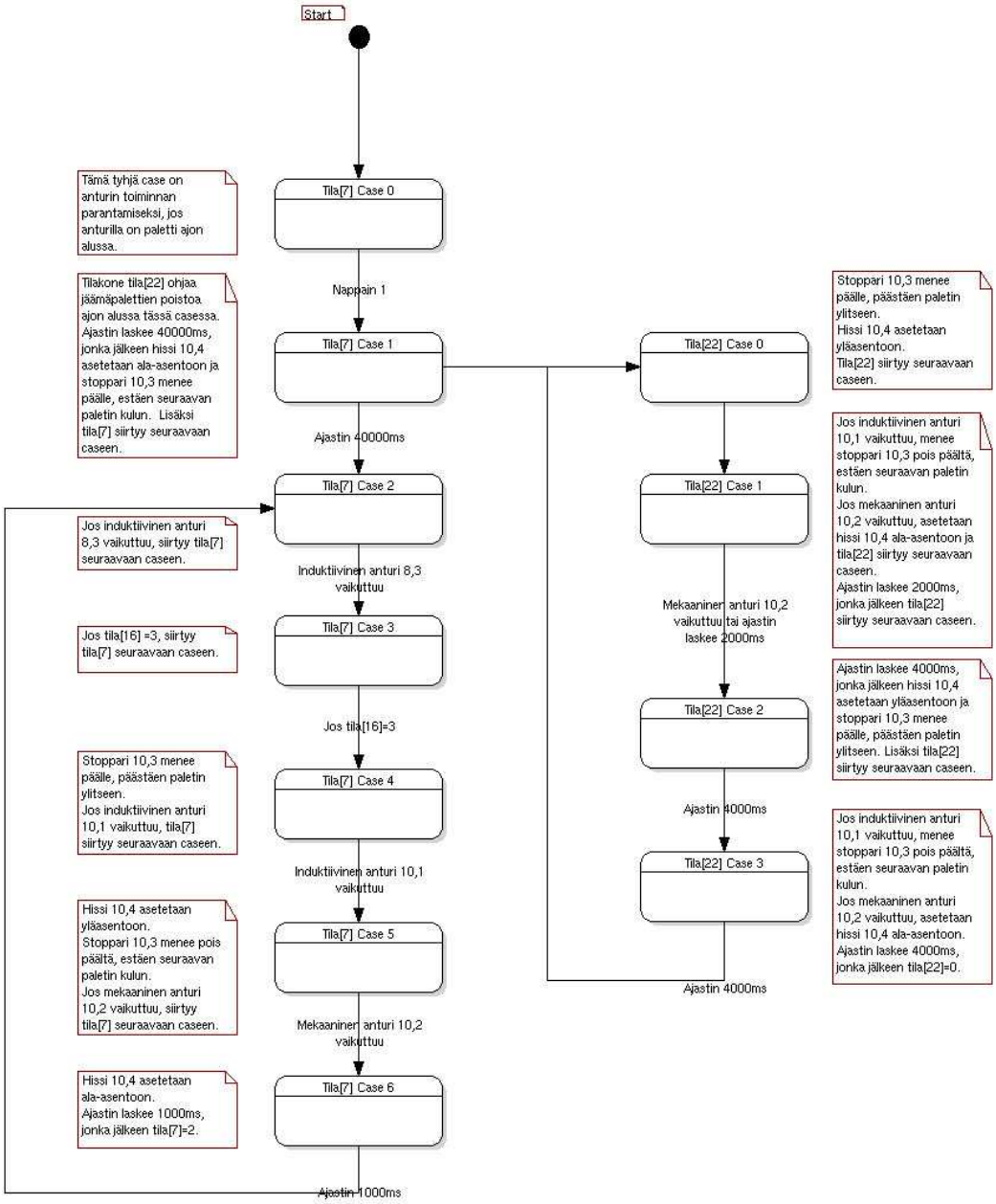
Tilakone 5 kaavio



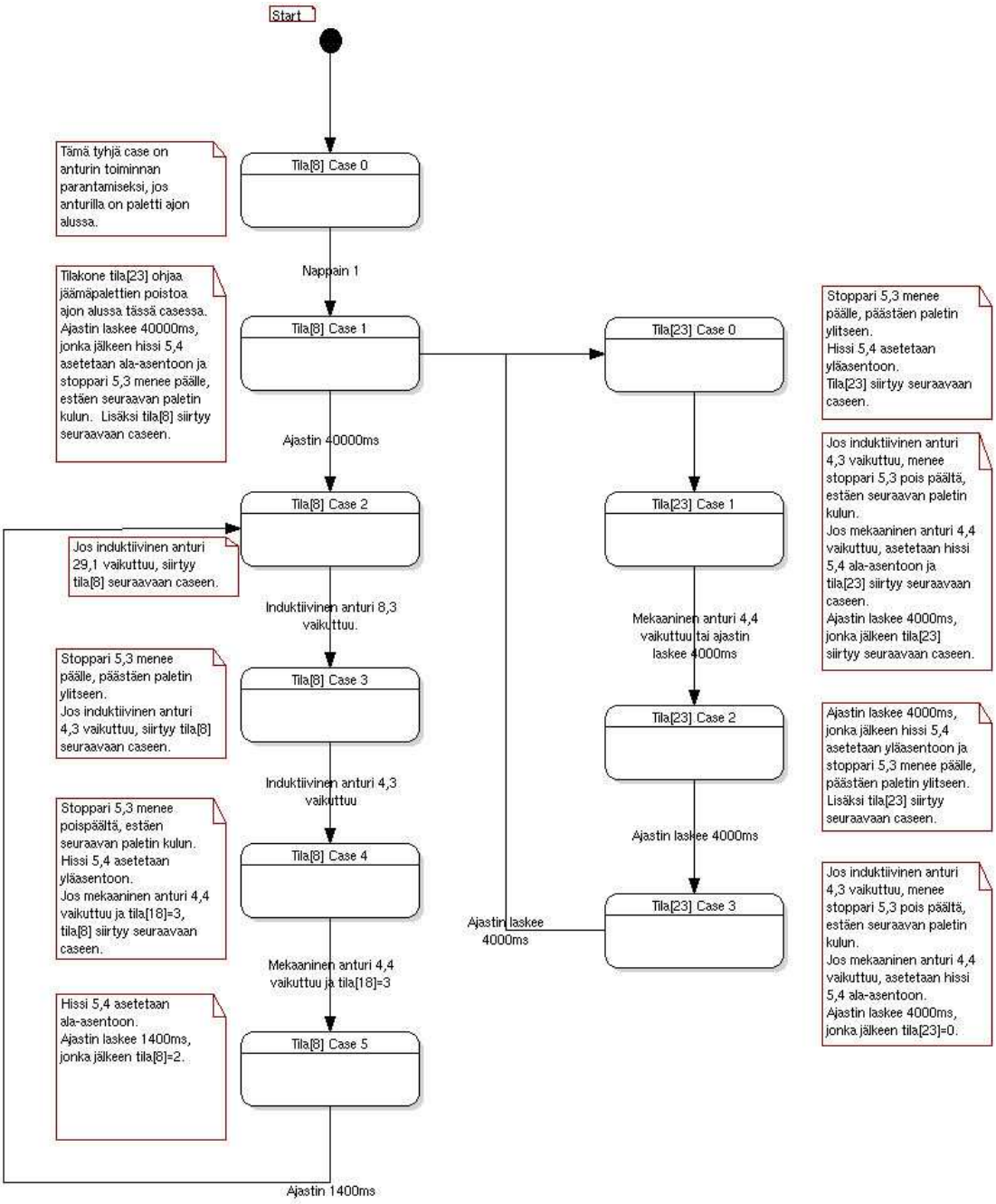
Tilakone 6 kaavio



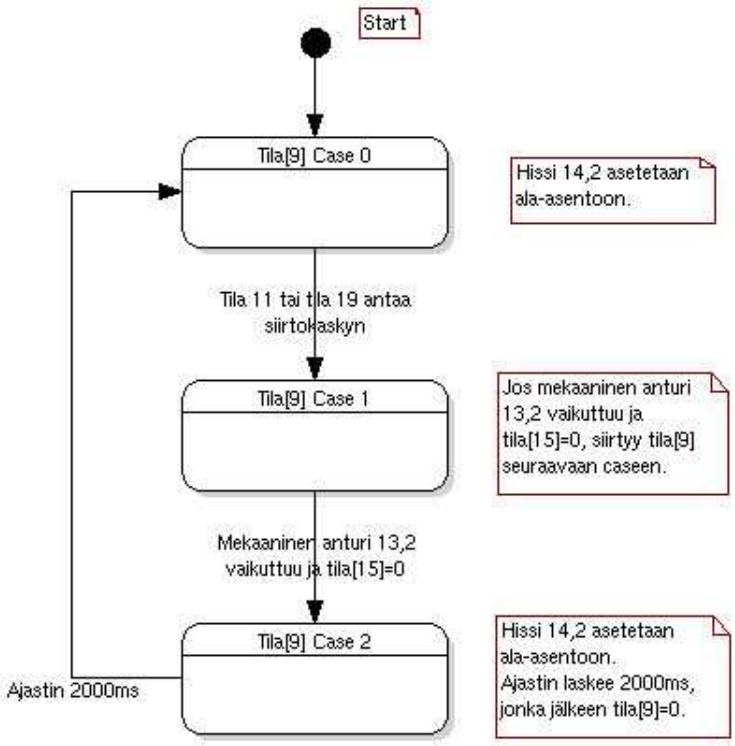
Tilakone 7 ja 22 kaavio



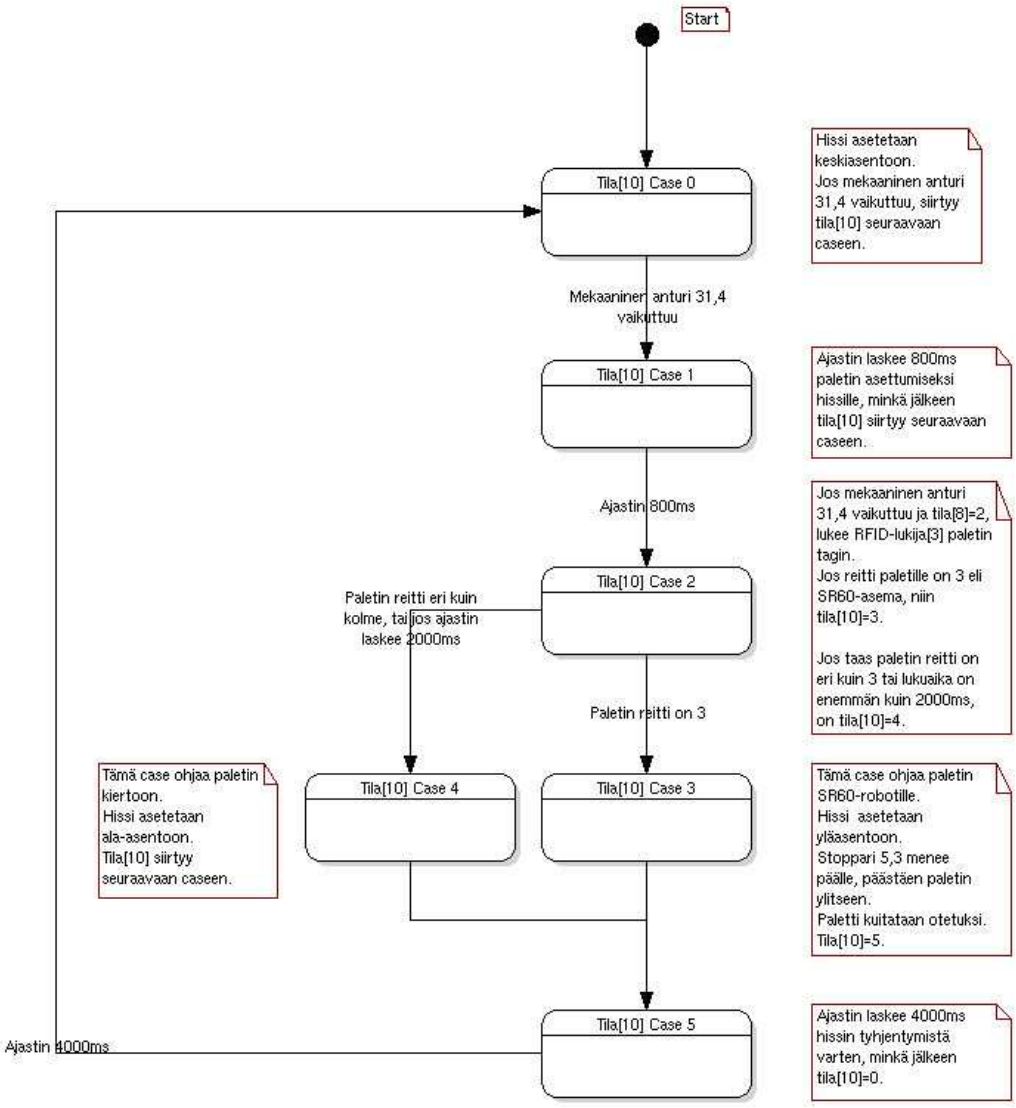
Tilakone 8 ja 23 kaavio



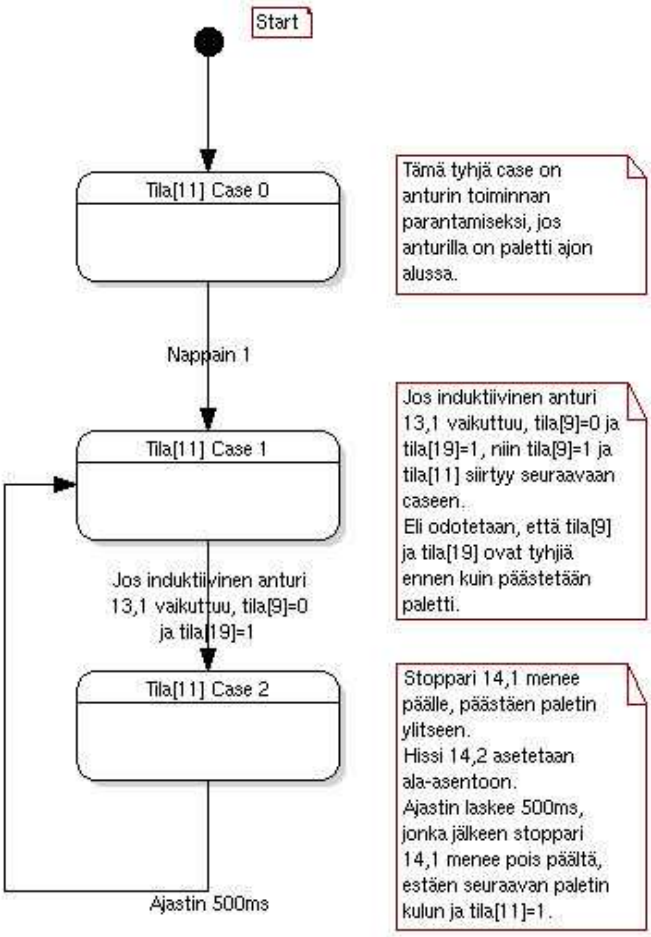
Tilakone 9 kaavio



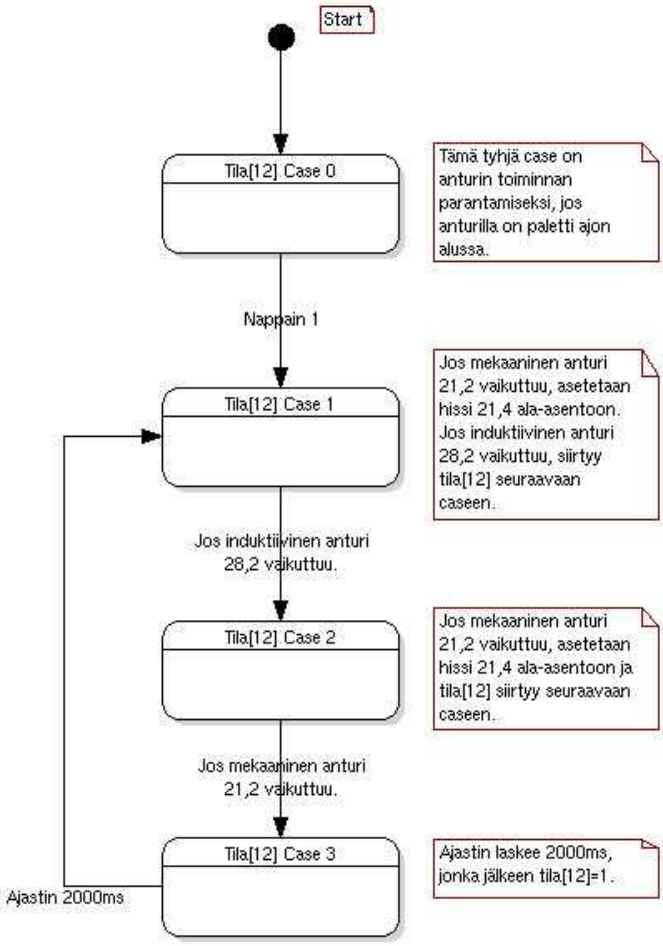
Tilakone 10 kaavio



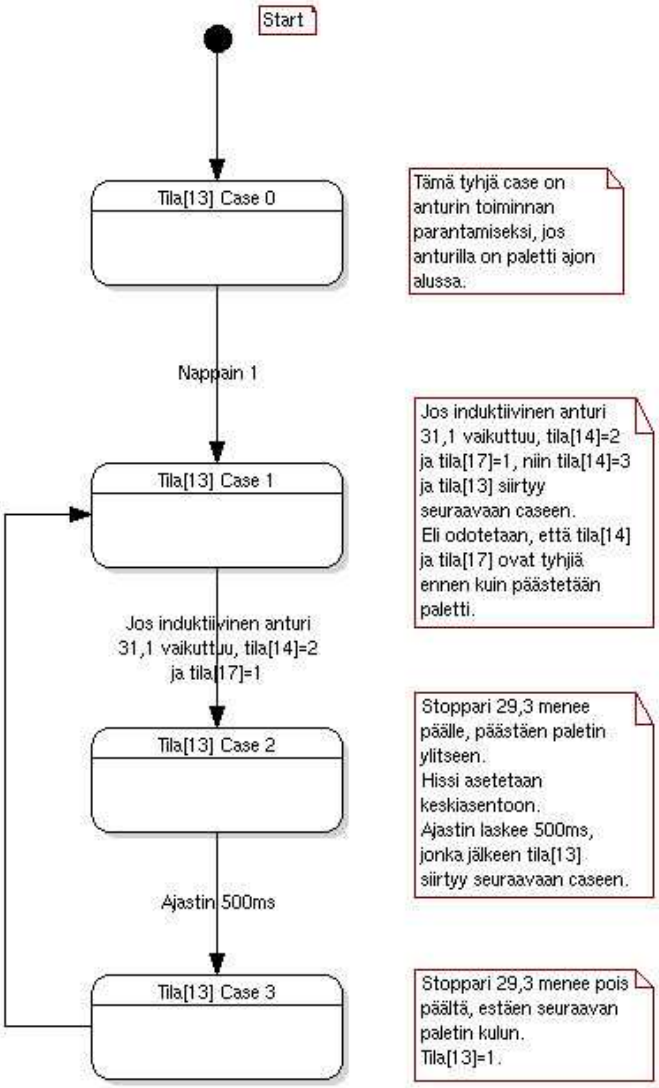
Tilakone 11 kaavio



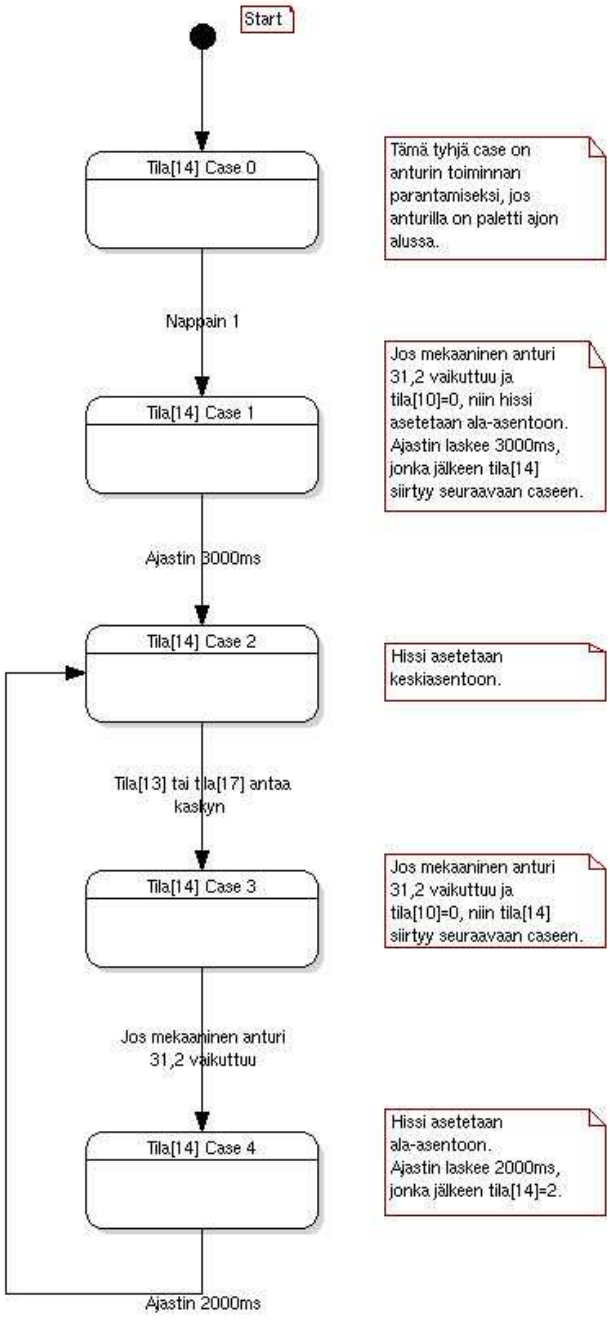
Tilakone 12 kaavio



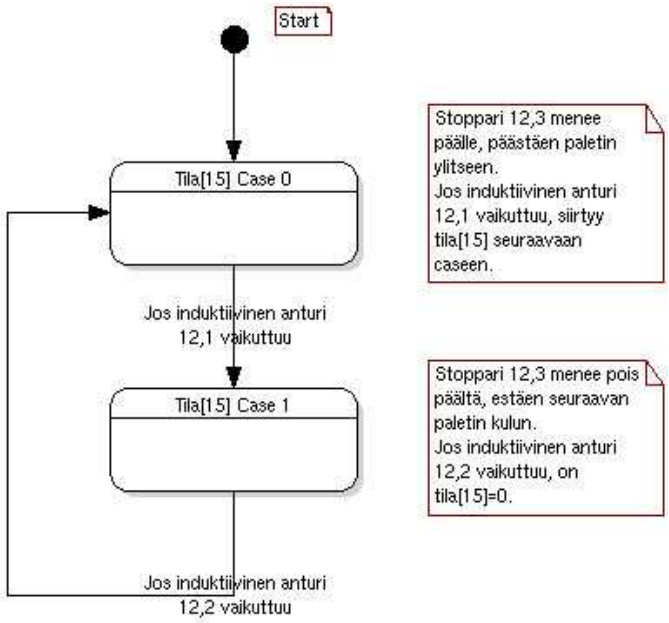
Tilakone 13 kaavio



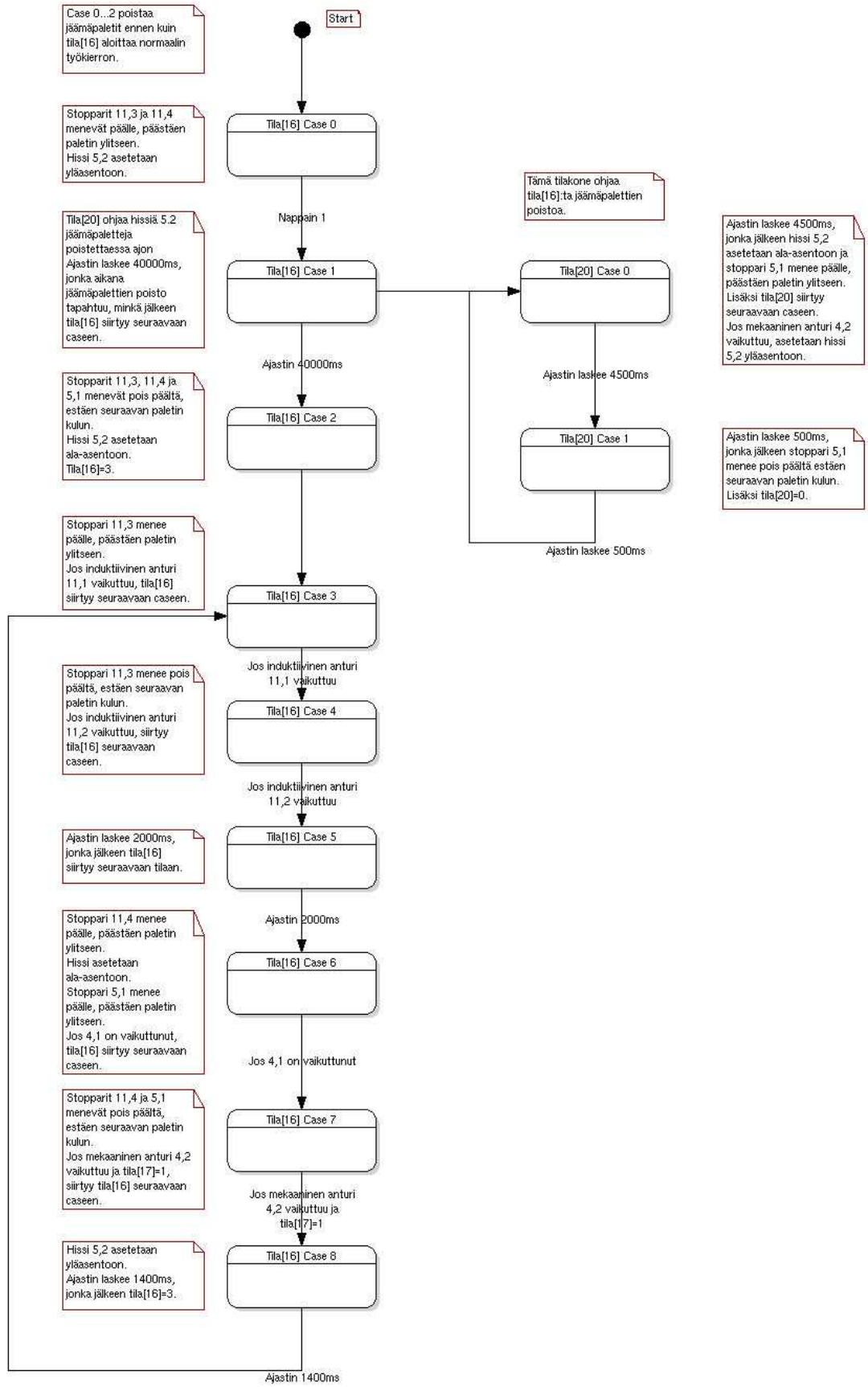
Tilakone 14 kaavio



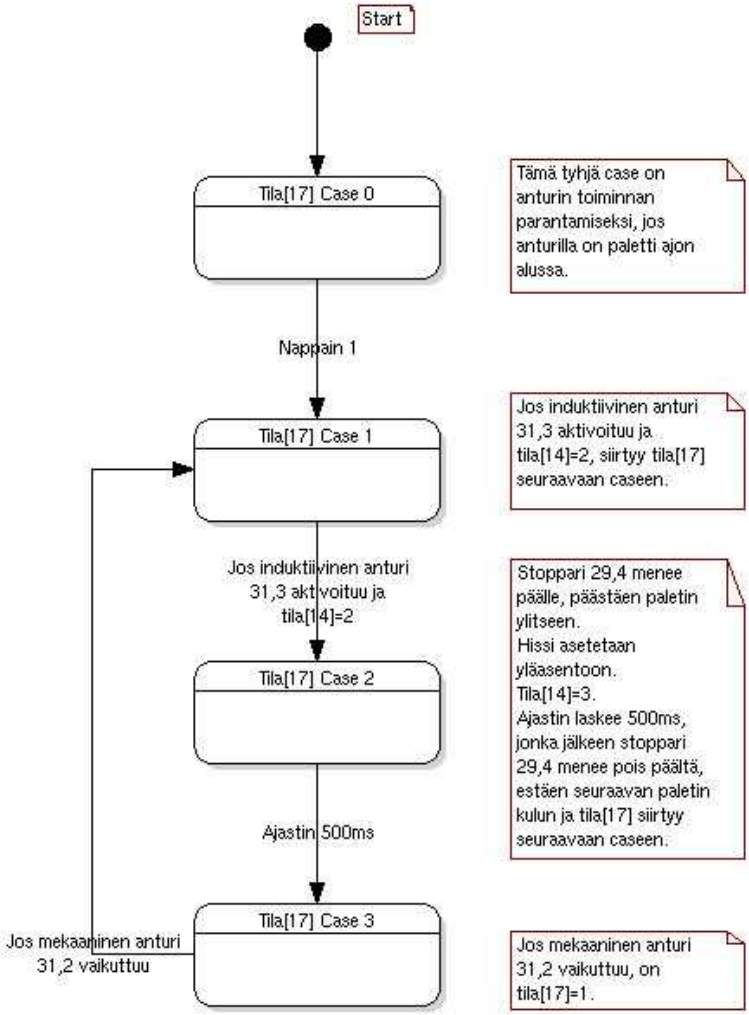
Tilakone 15 kaavio



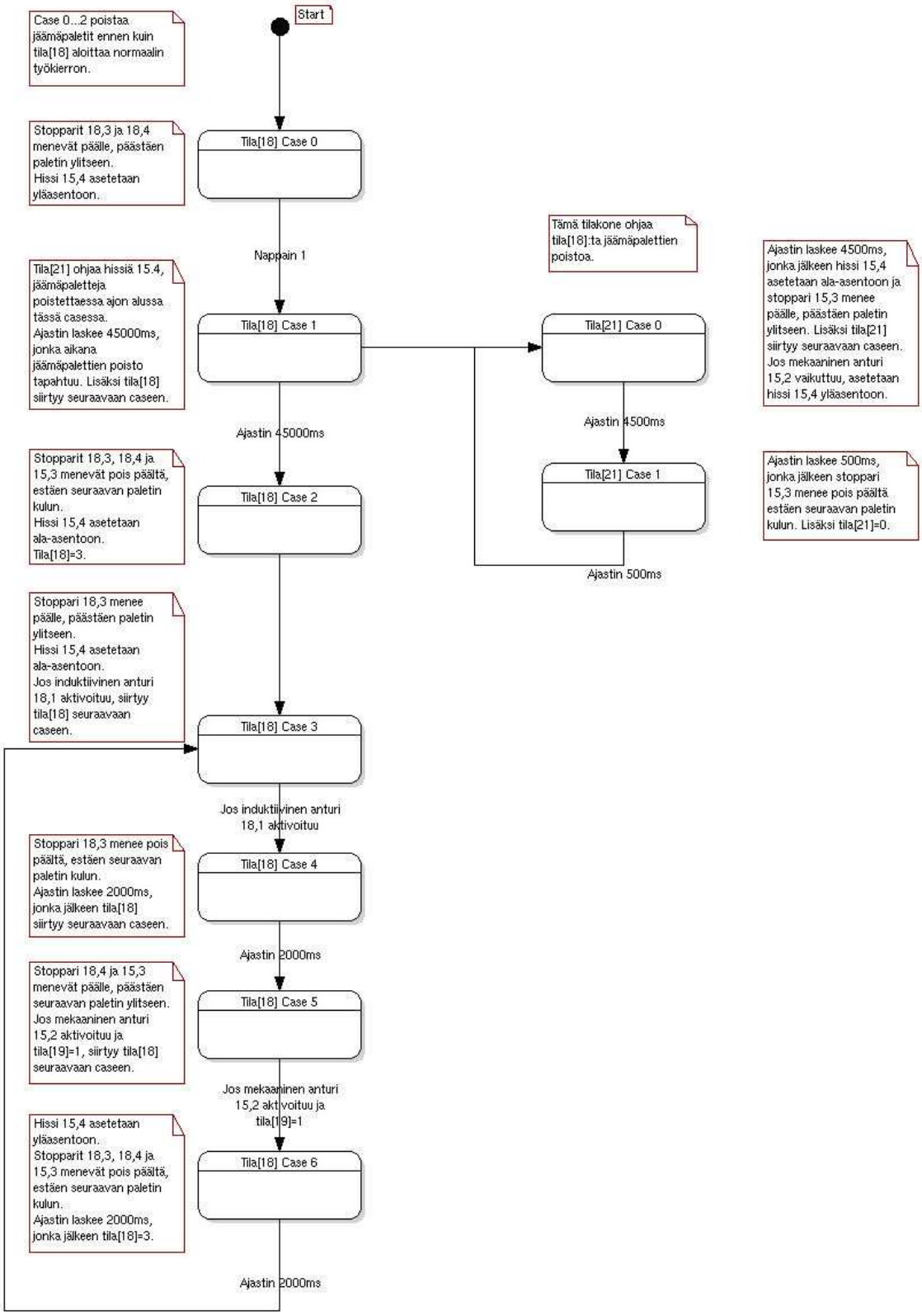
Tilakone 16 ja 20 kaavio



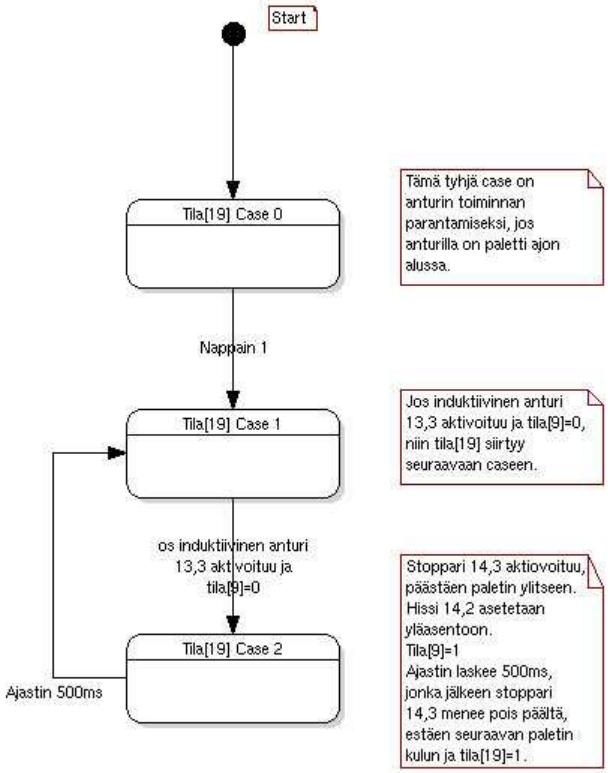
Tilakone 17 kaavio



Tilakone 18 ja 21 kaavio



Tilakone 19 kaavio



Ohjelman koodi

```
// miniCANopen.cpp : Defines the entry point for the console application.
//

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "can.h"
#include "kvaserdriver.h"
#include "inputevent.h"
#include "util/vt100.h"
#include "util/misc.h"

// Paletti luokan esittely
// Jokaista linjan palettia vastaan yksi tämän luokan instanssi
// Luokan metodeilla voidaan määrätä palettikohtaisesti palettien reitti
ja kierrosten lukumäärä

class paletti
{
public:
           //int sijainti;
        void otettu();
           // Kun paletti otettu jollekkin asemalle, funktio muuttaa
           paletin vaihetta

        int kohde();
// Lukijat tarkistavat minne paletin kuuluisi mennä
// Funktio palauttaa seuraavana vuorossa olevan työpisteen
numeron

        void aseta_reitti(int k0, int k1, int k2, int k3, int k4, int k5,
        int k6, int k7, int k8, int k9);
        // Parametreiksi annetaan työkierto asemien numeroina
        // 1=manuaaliasema
        // 2=sr6
        // 3=sr60
        // Viimeisen vaiheen on oltava 0, jolloin työkierto palaa alkuun

           void aseta_tyovaihe(int k0, int k1, int k2, int k3, int k4,
           int k5, int k6, int k7, int k8, int k9);
           //asetetaan työvaihe paleteille mahd. arvot 1-3

        void aseta_kierrosten_maara(int k);
           // Parametriksi annetaan kuinka monta kertaa työkierto
           halutaan        suorittaa
        // Arvolla 0 työkiertoa ajetaan ikuisesti

        int kierros();
// Palauttaa arvon joka kertoo kuinka monta kertaa työkierto on käyty
läpi

        void aktivoi();
        void deaktivoi();
// Paletti aktivoidaan tai deaktivoidaan. Deaktivoitu paletti kiertää
linjaston keskellä

        void reset();
// Kierrosten määrä ja työkierron vaihe nollataan
```

```
        bool aktiivisuus();
        // Palauttaa boolean arvon, joka kertoo onko paletti aktiivinen

        paletti();
        virtual ~paletti();

       int sijainti;  //1 = manuaali-asema, 2 = sr6, 3 = sr60
//pidetään lukua onko paletti otettu jollekkin roboteista, jotta sitten
robotilla
//voidaan tarkistaa, että mikä paletti on tulossa robolle (voidaan lukea
robon arvot). Arvo asetetaan rfid:n lukuhetkellä tila[0],[6],[10]:ssa

//Vai onko paletilla tieto siitä missä kohtaa linjalle paletti on
menossa? --JH

private:
        int reitti[10];
        int tyovaihe[10];       //työvaiheen numerot eli mitä tehdään
                                kyseisellä robolla.
        int vaihe;
        int kierrosten_maara;
        int kierroslaskuri;
        bool aktiivinen;
        int robotinOhjelma;
        int kuittausRobotilta;   //0 = susi, 1 = ok, 2 = aika ylitetty
//asetetaan kuittaus-tyyppi. paletti odottaa tietyn ajan ja jos
kuittausta ei siinä
//ajassa ole tullut niin asetetaan tila '3' jolloin paletti siirtyy
yleiseen kiertoon

};


// ************************
//         Palettiluokan toteutus
// ************************

// Konstruktori
paletti::paletti()
{
}

// Destruktori
paletti::~paletti()
{
}

void paletti::otettu() {
        vaihe++;
        if(reitti[vaihe]==0) {
                vaihe=0;
                kierroslaskuri++;
        }
        if(kierroslaskuri>=kierrosten_maara && kierrosten_maara!=0) {
                deaktivoi();
        }

}

int paletti::kohde() {
```

```
        if(aktiivinen)
                return reitti[vaihe];
        else return 0;
}

void paletti::aseta_reitti(int k0, int k1, int k2, int k3, int k4, int
k5, int k6, int k7, int k8, int k9) {
        reitti[0]=k0;
        reitti[1]=k1;
        reitti[2]=k2;
        reitti[3]=k3;
        reitti[4]=k4;
        reitti[5]=k5;
        reitti[6]=k6;
        reitti[7]=k7;
        reitti[8]=k8;
        reitti[9]=k9;
}


void paletti::aseta_tyovaihe(int k0, int k1, int k2, int k3, int k4, int
k5, int k6, int k7, int k8, int k9) {
           tyovaihe[0] = k0;
           tyovaihe[1] = k0;
           tyovaihe[2] = k0;
           tyovaihe[3] = k0;
           tyovaihe[4] = k0;
           tyovaihe[5] = k0;
           tyovaihe[6] = k0;
           tyovaihe[7] = k0;
           tyovaihe[8] = k0;
           tyovaihe[9] = k0;
}

void paletti::aseta_kierrosten_maara(int k) {
        kierrosten_maara=k;
        kierroslaskuri=0;
        vaihe=0;
}

int paletti::kierros() {
        return kierroslaskuri;
}

void paletti::aktivoi() {
        aktiivinen=true;
}

void paletti::deaktivoi() {
        aktiivinen=false;
}

void paletti::reset() {
        kierroslaskuri=0;
        vaihe=0;
}

bool paletti::aktiivisuus() {
        return aktiivinen;
}
```

```
// **********************************
// Globaalien apufunktioiden esittely
// **********************************
void reittien_alustus();
void nodeon(int CANnode, int slave, int bitti);
void nodeoff(int CANnode, int slave, int bitti);
bool nodestate(int CANnode, int slave, int bitti);
//testiä: olivat ennen ..nodeon(int CAN_NodeID..)


// **************************************
// Ohjelman Globaalit muuttujat ja luokat
// **************************************

// Taulukko tilakoneille
int tila[24];
char tilan_viesti[20][50];

//Muuttujat rfid-lukijoille
unsigned char lukija[3];

//Luodaan 13 palettia
paletti p[13];

//Laskuri ohjelman nopeuden selvittämistä varten
int laskuri;

// Luodaan globaalit PDOt jotka sisältävät linjan inputtien tilat
PDO tpdo181(0x181); //Digitaali-inputit 1-15
PDO tpdo281(0x281); //Digitaali-inputit 16-31
PDO tpdo1c1(0x1c1); //RFID-lukija 1
PDO tpdo2c1(0x2c1); //RFID-lukija 2
PDO tpdo3c1(0x3c1); //RFID-lukija 3

//ASi node 2 transmit pdot
PDO tpdo381(0x381);   //Digitaali-inputit 1-15 ??? -JH

// Linjan outputit ovat näissä kahdessa RDPOssa
PDO rpdo201(0x201); //Digitaalioutputit 1-15
PDO rpdo301(0x301); //Digitaalioutputit 16-31

PDO rpdo401(0x401); //Digitaalioutputit 1-15 ASi Master 2

//testiä:
//uuden konfiguraation vaatimat muuttujat@14.4 -JH
//node 2:
PDO tpdo182(0x182);   //node 2, input 1-15
PDO tpdo282(0x282);   //input 1-15 //onko tätä? -JH
PDO rpdo202(0x202);   //node 2, output 1-15
PDO rpdo302(0x301);   //output 16-31        //onko tätä? -JH

int Run=1;
int LinjaAsiNode=1;
int RoboAsiNode=2;


// ***********
// Pääohjelma
// ***********

int main()
```

```
{
//Asetetaan paleteille reitit
reittien_alustus();

// Alustetaan näppäimistön käyttö
eventinit();

KvaserDriver driver;

// Avataan megabitin nopaudella
driver.open(1000000);
driver.set_debug(6);
decoder_openlog("\\temp\\canlog.txt");

// Luodaan CAN olio
CAN can;
can.set_debug(0); // Tulostetaan errorit ja ohjelma pysähtyy CAN-pinon
virhetilanteisiin

// Ladataan CAN driver
can.load_driver(driver);
// Pysäytetään kaikien noodien toiminta

can.stop_node(0);
u_sleep(100000);

// Määritellään PDO:den pituudet
tpdo181.set_len(8);
tpdo281.set_len(8);
tpdo1c1.set_len(8);
tpdo2c1.set_len(8);
tpdo3c1.set_len(8);
//node 2
tpdo381.set_len(1);

rpdo201.set_len(8);
rpdo201.set_value(0); // Nollataan RDPOn kaikkien kahdeksan tavun sisältö
rpdo301.set_len(8);
rpdo301.set_value(0); // Nollataan RDPOn kaikkien kahdeksan tavun sisältö
//node 2
rpdo401.set_len(1);
rpdo401.set_value(0);

//testiä:
//asetetaan pituudet.
tpdo182.set_len(1);
rpdo202.set_len(8);   // pitäisikö olla ..(1)..?      --JH
rpdo201.set_value(0);

// Mapataan rfid-lukijoiden muuttujat oikeisii tpdo:hin.
tpdo1c1.mapcopy(0,lukija[1]);
tpdo2c1.mapcopy(0,lukija[2]);
tpdo3c1.mapcopy(0,lukija[3]);

// Lisätään TPDOt listaan
can.add_tpdo(tpdo181);
can.add_tpdo(tpdo281);
can.add_tpdo(tpdo1c1);
can.add_tpdo(tpdo2c1);
can.add_tpdo(tpdo3c1);
```

```
can.add_tpdo(tpdo381);

//testiä:
//lisätään luotujen tpdo182 + rpdo202 listaan. -- JH
can.add_tpdo(tpdo182);
can.add_rpdo(rpdo202);

// Lisätään RPDOt listaan
can.add_rpdo(rpdo201);
can.add_rpdo(rpdo301);

can.add_rpdo(rpdo401);

//SDO alustukset

// Asetetaan CAN-noodi preoperational-tilaan
can.set_preoperational(0);
u_sleep(100000);


// IFM:n boksin alustukset
// Tarvitaan 5 TPDO:ta ja 2 RPDO:ta
// Määrätään PDO-viesteille COBID:t
// TDPO cobid konfigurointi
can.sdo_write(1, 0x1800, 1, 4, 384+1);
can.sdo_write(1, 0x1801, 1, 4, 640+1);
can.sdo_write(1, 0x1802, 1, 4, 448+1);
can.sdo_write(1, 0x1803, 1, 4, 704+1);
can.sdo_write(1, 0x1804, 1, 4, 960+1);

can.sdo_write(1, 0x1805, 1, 4, 896+1);

// RDPO cobid konfigurointi
can.sdo_write(1, 0x1400, 1, 4, 512+1);
can.sdo_write(1, 0x1401, 1, 4, 768+1);

can.sdo_write(1, 0x1402, 1, 4, 1024+1);


u_sleep(100000);
can.start_node(0);
u_sleep(100000);

VT100::clearscreen();

//luodaan globaali-tekstitiedosto, johon voi kirjoittaa halutussa kohtaa
ohjelmaa tekstiä ---@7.4 JH
int iApu;
//apumuuttuja (voidaan tallentaa tiloja esim. iApu = nodeon(x,x,x)
FILE *pFile;
pFile = fopen("debug_tekstia.txt", "w");
//"w" = kirjoita tiedostoon, ylikirjoita kaikki, luo tarvittavissa
//"a" = jatka jo olemassa olevan perään
if (pFile == NULL)
          perror("error opening file");
fprintf(pFile, "debug tekstiä\n*****************************\n");
//käytetään kirjoittamalla: "fprintf(pFile, "_teksti_tähän_, %d, %s",
int_muuttuja, char_muuttuja);"

// Pääohjelma alkaa tästä ja suorittaa ikuista looppia
```

```
//tuleva valikko-ohjelma:
int valinta = 0;
//do{
          printf("This is the program for the production line!!\n\n");
          printf("\nWhen the line program is running you should stretch
your terminal window downwards so that all information can be shown at
the screen\n\n");
          printf("The following keys are valid:\n\n");
          printf("When the production line program is _running_ pressing
'1' will activate the line. Pressing '0' will shutdown the line. Pressing
'ESC' will shutdown the line and exit\n\n");
          printf("What do you want me to do MASTER?\n");
          printf("Pressing number '1' will start the production-line
program\n");
          printf("Any other button will end quit the program\n\n\n\n");
          printf("your selection: ");
          scanf("%d", &valinta);
          VT100::clearscreen();
          if (valinta == 1) {

do{

          // Linjan moottorit päälle kun painetaan näppäintä '1'
          if (keydown('1'))
          {
                    //moottorit käynnistetään yksi kerrallaan
                    virtapiikin estämiseksi
                    for(int i=1; i<=23; i++) {
                              if(i==1) nodeon(1,17,1);
                              //manuaalilinjan moottorit
                              if(i==2) nodeon(1,27,1);
                              if(i==3) nodeon(1,27,2);
                              if(i==4) nodeon(1,6,1);
                              if(i==5) nodeon(1,6,2);
                              if(i==6) nodeon(1,6,3);
                              if(i==7) nodeon(1,16,1);
                              if(i==8) nodeon(1,16,2);
// tehdään alustus, jossa '1' napin painalluksella jäämäpalettien poiston
caset lähtevät käyntiin
                              if(i==9) tila[0]=1;
                              if(i==10) tila[1]=1;
                              if(i==11) tila[2]=1;
                              if(i==12) tila[3]=1;
                              if(i==13) tila[6]=1;
                              if(i==14) tila[7]=1;
                              if(i==15) tila[8]=1;
                              if(i==16) tila[11]=1;
                              if(i==17) tila[12]=1;
                              if(i==18) tila[13]=1;
                              if(i==19) tila[14]=1;
                              if(i==20) tila[16]=1;
                              if(i==21) tila[17]=1;
                              if(i==22) tila[18]=1;
                              if(i==23) tila[19]=1;
                              printf("Moottori :%d\n",i);
                              u_sleep(500000); //500ms
                              can.send_only_changed_rpdos();

                              }
          }
```

```
            // Linjan moottorit pois päältä kun painetaan näppäintä '0'
            if (keydown('0'))
            {
//moottorit pysäytetään yksi kerrallaan virtapiikin estämiseksi
                        for(int i=1; i<=8; i++) {
                                    if(i==1) nodeoff(1,17,1);
//manuaalilinjan moottorit
                                    if(i==2) nodeoff(1,27,1);
                                    if(i==3) nodeoff(1,27,2);
                                    if(i==4) nodeoff(1,6,1);
                                    if(i==5) nodeoff(1,6,2);
                                    if(i==6) nodeoff(1,6,3);
                                    if(i==7) nodeoff(1,16,1);
                                    if(i==8) nodeoff(1,16,2);
                                    printf("Moottori :%d\n",i);
                                    u_sleep(500000); //500ms
                                    can.send_only_changed_rpdos();
                        }
            }


if(Run==1)
{

// Varsinainen ohjelman logikka alkaa
// 20 tilakonetta seuraavat linjaston tapahtumia, ja ohjaavaat
toimilaitteita
// lisäksi neljä tilakonetta ohjaavat jäämäpalettien poistoa
switch(tila[0]) {

            // case 0-1 ovat jäämäpalettien poistamiseksi ajon alussa
            case 0:
            //tyhjä case anturien toiminnan parantamiseksiajon alussa
            break;

            case 1:
                        if(nodestate(1,21,1))   //Jos mekaaninen anturi
21,1 vaikuttuu, asetetaan hissi 20,3 yläasentoon.
                                    nodeon(1,20,3);
                        if(timedelay(0,25000000)) //Ajastin laskee 25000ms,
jonka jälkeen tila[0] siirtyy seuraavaan caseen.
                                    tila[0]++;
            break;

            case 2:
                        strcpy(tilan_viesti[0],"20.1");
                        nodeoff(1,20,4);                //Hissi asetetaan
keskiasentoon.
                        nodeoff(1,20,3);
                        if(nodestate(1,20,1))           //Jos anturi 20,1
vaikuttuu, siirtyy tila[0] seuraavaan caseen.
                                    tila[0]++;
            break;

            case 3:
                        strcpy(tilan_viesti[0],"20.2");
                        nodeon(1,21,3);
            //Stoppari 21,3 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,20,2)) //Jos induktiivinen anturi
20,2 vaikuttuu, siirtyy tila[0] seuraavaan caseen.
                                    tila[0]++;
```

```
            break;

            case 4:
                        strcpy(tilan_viesti[0],"21.1");
                        nodeoff(1,21,3);      //Stoppari 21,3 menee
                        poispäältä, estäen seuraavan paletin kulun.
                        if(nodestate(1,21,1)) {
//Jos mekaaninen anturi 21,1 vaikuttuu, ja jos tila[1]=1, niin RFID-
lukija[1] lukee paletin tagin.
                        strcpy(tilan_viesti[0],"tila[4], tila[12],
                        tila[1]");
                        if(tila[1]==1 && p[ lukija[1] ].kohde()==1 ) {

//Jos reitti paletille on 1 eli manuaaliasema, niin silta 20,3 vaikuttuu,
tila[0] siirtyy seuraavaan caseen ja tila[1]=1.
                                    resettimedelay(0);
                                    nodeon(1,20,3);
                                    tila[0]++;
                                    tila[1]=1;
                                    p[lukija[1]].otettu(); //kuitataan että
paletti otettiin
                                    p[lukija[1]].sijainti = 1;
            //astetaan paletin sijainniksi 1, eli manuaaliasema
                                    }
//Jos kohde ei ole manuaaliasema tai tunnistuksessa kestää yli sekunti
//(paletti ilman tagia?), ajetaan suoraan
                                    else if((lukija[1]!=0 && p[ lukija[1]
].kohde()!=1) || timedelay(0,1000000)) {
                                                if(tila[4]==0 && tila[12]==1)
{                       //Jos tila[4]=0 ja tila[12]=1, vaikuttuu silta
20,4, tila[12]=2 ja tila[0] siirtyy seuraavaan caseen.
                                                            resettimedelay(0);
                                                            nodeon(1,20,4);
                                                            tila[12]=2;
                                                            tila[0]++;
                                                }
                                    }
                        }
            break;

            case 5: //viive että paletti ehtii pois hissistä, sitten
            nollaus
                        strcpy(tilan_viesti[0],"Ajastin 2s");
                        if(timedelay(0,2000000))
                        //Kun ajastin on laskenut 2000ms, on tila[0]=2.
                        tila[0]=2;
            break;


}

switch(tila[1]) {

            case 0:
            //tyhjä case anturien toiminnan parantamiseksi ajon alussa
            break;

            case 1:
                        strcpy(tilan_viesti[1],"22.1");
```

```
                    if(nodestate(1,22,1) && tila[2]==2)
          //Jos induktiivinen anturi 22,1 vaikuttuu ja tila[2]=2,
siirtyy tila[1] seuraavaan caseen.
                              tila[1]++;
          break;

          case 2:
                    strcpy(tilan_viesti[1],"22.2");
                    nodeon(1,22,3);
          //Stoppari 22,3 menee päälle, päästäen paletin ylitseen.

                    if(nodestate(1,22,2)) //Jos induktiivinen anturi
22,2 vaikuttuu, siirtyy tila[1] seuraavaan caseen.
                              tila[1]++;
          break;

          case 3:
                    strcpy(tilan_viesti[1],"23.1");
                    nodeoff(1,22,3);
          //Stoppari 22,3 menee poispäältä, estäen seuraavan paletin
kulun.
                    if(nodestate(1,23,1)) {                    //Jos
induktiivinen anturi 23,1 vaikuttuu, on tila[1]=1.
                              tila[1]=1;
                    }
          break;
}

switch(tila[2]) {
          //Case 0…1 poistaa jäämäpaletit ennen kuin tila[2] aloittaa
normaalin työkierron.
          case 0:
          //tyhjä case anturien toiminnan parantamiseksiajon alussa
          break;

          case 1:
          //if-lause yhden stopparin toiminnan parantamiseksi
jäämäpaletteja poistettaessa ajon alussa
                    if(tila[3]==1) {                //Jos tila[3]=1,
niin stoppari 26,3 menee päälle, päästäen paletin ylitseen.
                              nodeon(1,26,3);
                              if(timedelay(2,500000)){
          //Lisäksi ajastin laskee 500ms, jonka jälkeen stoppari 26,3
menee pois päältä, estäen seuraavan paletin kulun ja tila[2] siirtyy
seuraavaan caseen.
                                                  nodeoff(1,26,3);
                                                  tila[2]++;
                                                  }
                                                  }
          break;

          case 2:

                    strcpy(tilan_viesti[2],"23.1");
                    if(nodestate(1,23,1))
//Jos induktiivinen anturi 23,1 vaikuttuu, siirtyy tila[2] seuraavaan
caseen.
                              tila[2]++;
          break;

          case 3:
```

```
                        strcpy(tilan_viesti[2],"gut tai auto");
                        nodeon(1,23,3);
//Pöytä asetetaan yläasentoon.
                        nodeoff(1,23,4);
//Toimilaite 23,4 menee pois päältä.
//kappale päästetään eteenpäin gut-napilla tai auto-valinnalla
                        if(!nodestate(1,25,3) || nodestate(1,25,2))
//Jos painonappi auto/man ei ole painettuna tai gut nappi on painettuna,
siirtyy tila[2] seuraavaan caseen.
                                    tila[2]++;
            break;

            case 4:
                        strcpy(tilan_viesti[2],"23.2");
                        nodeon(1,22,4);
//Pöytä 22,4 asetetaan ala-asentoon.
                        nodeoff(1,23,3);
//Toimilaite 23,3 menee pois päältä.
                        nodeon(1,23,4);
//Toimilaite 23,4 menee päälle.
                        if(nodestate(1,23,2))
//Jos induktiivinen anturi 23,2 vaikuttuu, siirtyy tila[2] seuraavaan
caseen.
                                    tila[2]++;
            break;

            case 5:
                        strcpy(tilan_viesti[2],"tila[3]");
                        nodeoff(1,22,4);
//Stoppari 22,4 menee pois päältä, estäen seuraavan paletin kulun.
                        if(tila[3]==1)
//Jos tila[3]=1, siirtyy tila[2] seuraavaan caseen.
                                    tila[2]++;
            break;

            case 6:
                        strcpy(tilan_viesti[2],"26.1");
                        nodeon(1,26,3);
            //Stoppari 26,3 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,26,1))
//Jos induktiivinen anturi 26,1 vaikuttuu, siirtyy tila[2] seuraavaan
caseen.
                                    tila[2]++;
            break;

            case 7:
                        nodeoff(1,22,4);
//Stopparit 22,4 ja 26,3 menevät pois päältä, estäen seuraavien palettien
kulun.
                        nodeoff(1,26,3);
                        tila[2]=2;            //Tila[2]=2.
            break;
}

switch(tila[3]) {

            case 0:
                        //tyhjä case anturien toiminnan parantamiseksi ajon
alussa
            break;
```

```
            case 1:
                        strcpy(tilan_viesti[3],"26.2");
                        if(nodestate(1,26,2)) //Jos mekaaninen anturi 26,2
vaikuttuu, siirtyy tila[3] seuraavaan caseen.
                                    tila[3]++;
            break;

            case 2:
                        strcpy(tilan_viesti[3],"ajastin 800ms");
                        if(timedelay(3,800000)) //Ajastin laskee 800ms,
jonka jälkeen tila[3] siirtyy seuraavaan caseen.
                                    tila[3]++;
            break;

            case 3:
                        strcpy(tilan_viesti[3],"24.2, tila[4]");
                        nodeon(1,26,4);                                 //Hissi
26,4 asetetaan yläasentoon.
                        if(nodestate(1,24,2) && tila[4]==0)
            //Jos mekaaninen anturi 24,2 vaikuttuu ja tila[4]=0, siirtyy
tila[3] seuraavaan caseen.
                                    tila[3]++;
            break;

            case 4:
                        strcpy(tilan_viesti[3],"24.3");
                        nodeoff(1,26,4);                //Hissi 26,4
asetetaan ala-asentoon.
                        if(nodestate(1,24,3)) //Jos induktiivinen anturi
24,3 vaikuttuu, on tila[3]=1.
                                    tila[3]=1;
            break;
}

switch(tila[4]) {
            case 0:
                        strcpy(tilan_viesti[4],"28.1, tila[12]");
                        if(nodestate(1,28,1) && tila[12]==1)        //Jos
induktiivinen anturi 28,1 vaikuttuu ja tila[12]=1, siirtyy tila[4]
seuraavaan caseen.
                                    tila[4]++;
            break;

            case 1:
                        strcpy(tilan_viesti[4],"28.2");
                        nodeon(1,28,3);                 //Stoppari 28,3
menee päälle, päästäen paletin ylitseen.
                        nodeon(1,21,4);                 //Hissi 21,4
asetetaan yläasentoon.
                        if(timedelay(4,500000)){        //Ajastin laskee
500ms, jonka jälkeen stoppari 28,3 menee pois päältä, estäen seuraavan
paletin kulun. Lisäksi tila[4]=0.
                                    nodeoff(1,28,3);
                                    tila[4]=0;
                        }
            break;

}

switch(tila[5]){
            case 0:
```

```
                        strcpy(tilan_viesti[5],"7.1");
                        nodeon(1,7,3);
//Stoppari 7,3 menee päälle päästäen paletin ylitseen.
                        if(nodestate(1,7,1))
//Jos induktiivinen anturi 7,1 vaikuttuu , siirtyy tila[5] seuraavaan
caseen.
                                    tila[5]++;
            break;

            case 1:
                        strcpy(tilan_viesti[5],"7.2");
                        nodeoff(1,7,3);
//Stoppari 7,3 menee poispäältä, estäen seuraavan paletin kulun.
                        if(nodestate(1,7,2)) {
//Jos induktiivinen anturi 7,2 vaikuttuu, on tila[5]=0.
                                    tila[5]=0;
                        }
            break;
}

switch(tila[6]) {
            //Case 0…1 poistaa jäämäpaletit ennen kuin tila[6] aloittaa
normaalin työkierron.
            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
                        if(nodestate(1,8,2))
//Jos mekaaninen anturi 8,2 vaikuttuu, asetetaan hissi 9,3 ala-asentoon.
                                    nodeon(1,9,3);
                        if(timedelay(6,3000000))
//Ajastin laskee 3000ms, jonka jälkeen tila[6] siirtyy seuraavaan caseen.
                                    tila[6]++;
            break;

            case 2:
                        strcpy(tilan_viesti[6],"7.2");
                        nodeoff(1,9,2);
//Hissi 9,2 ja 9,3 asetetaan keskiasentoon.
                        nodeoff(1,9,3);
                        if(nodestate(1,7,2))
//Jos induktiivinen anturi 7,2 vaikuttuu, siirtyy tila[6] seuraavaan
caseen.
                                    tila[6]++;
            break;

            case 3:
                        strcpy(tilan_viesti[6],"8.1");
                        nodeon(1,9,1);
//Stoppari 9,1 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,8,1))  //Jos induktiivinen anturi
8,1 vaikuttuu, siirtyy tila[6] seuraavaan caseen.
                                    tila[6]++;
            break;

            case 4:
                        strcpy(tilan_viesti[6],"8.2, tila[7]");
                        nodeoff(1,9,1);                  //Stoppari 9,1
menee poispäältä, estäen seuraavan paletin kulun
```

```
                        if(nodestate(1,8,2)) {                      //Jos
mekaaninen anturi 8,2 vaikuttuu ja tila[7]=2, lukee RFID-lukija[2]
paletin tagin.
                                    if(tila[7]==2 &&
p[lukija[2]].kohde()==2) { //Jos reitti paletille on 2 eli SR6-asema,
niin tila[6]=5 ja paletti kuitataan otetuksi.
                                                resettimedelay(6);
                                                tila[6]=5;
                                                p[lukija[2]].otettu();
//testiä:
                                                p[lukija[2]].sijainti = 2;
                                                //asetetaan paletin-
sijainniksi rfid:n numero
                                                //jotta tiedetään mikä
paletti on tulossa kyseiselle robolle.
                                    }
                                    else if((lukija[2]!=0 && p[ lukija[1]
].kohde()!=2) || timedelay(6,1000000)) {    //Jos taas paletin retti on
eri kuin 2 tai lukuaika on enemmän kuin 1000ms, on tila[6]=6.
                                                resettimedelay(6);
                                                tila[6]=6;
                                    }
                                    tila[5]=0; //Asetetaan tila[5]=0.
                        }
            break;

            //Case 5 robotille, case 6 kiertoon
            case 5:
                        //Tämä case ohjaa paletin SR6-robotille.
                        strcpy(tilan_viesti[6],"ajastin 2s");
                        nodeon(1,9,2);        //Hissi 9,2 asetetaan
yläasentoon.
                        if(timedelay(6,2000000)) //Kun ajastin on laskenut
2000ms kuluneeksi, on tila[6]=2.
                                    tila[6]=2;
            break;

            case 6:
                        //Tämä case ohjaa paletin kiertoon.
                        strcpy(tilan_viesti[6],"ajastin 2s");
                        nodeon(1,9,3);        //Hissi 9,3 asetetaan ala-
asentoon.
                        if(timedelay(6,2000000))  //Kun ajastin on laskenut
2000ms kuluneeksi, on tila[6]=2.
                                    tila[6]=2;
            break;

}

switch(tila[7]) {
            //case 0-1 poistaa jäämäpaletit aluksi ennen kuin tila[7]
aloittaa normaalin työkierron
            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
                        //Tilakone tila[22] ohjaa jäämäpalettien poistoa
ajon alussa tässä casessa.
```

```
                    switch(tila[22]) {
//Tämä tilakone ohjaa tila[7]:n jäämäpalettien poistoa.

                                case 0:
                                        nodeon(1,10,3);
          //Stoppari 10,3 menee päälle, päästäen paletin ylitseen.
                                        nodeon(1,10,4);
//Hissi 10,4 asetetaan yläasentoon.
                                        tila[22]++;
          //Tila[22] siirtyy seuraavaan caseen.
                                break;

                                case 1:
                                        if(nodestate(1,10,1))
//Jos induktiivinen anturi 10,1 vaikuttuu, menee stoppari 10,3 pois
päältä, estäen seuraavan paletin kulun.
                                                nodeoff(1,10,3);
                                        if(nodestate(1,10,2)){
          //Jos mekaaninen anturi 10,2 vaikuttuu, asetetaan hissi 10,4
ala-asentoon ja tila[22] siirtyy seuraavaan caseen.
                                                nodeoff(1,10,4);
                                                tila[22]++;
                                        }
                                        if(timedelay(22,2000000))
//Ajastin laskee 2000ms, jonka jälkeen tila[22] siirtyy seuraavaan
caseen.
                                                tila[22]++;
                                break;

                                case 2:
                                        if(timedelay(22,4000000)){
//Ajastin laskee 4000ms, jonka jälkeen hissi 10,4 asetetaan yläasentoon
ja stoppari 10,3 menee päälle, päästäen paletin ylitseen. Lisäksi
tila[22] siirtyy seuraavaan caseen.
                                                nodeon(1,10,4);
                                                nodeon(1,10,3);
                                                tila[22]++;
                                                }
                                break;

                                case 3:
                                        if(nodestate(1,10,1))
//Jos induktiivinen anturi 10,1 vaikuttuu, menee stoppari 10,3 pois
päältä, estäen seuraavan paletin kulun.
                                                nodeoff(1,10,3);
                                        if(nodestate(1,10,2))
//Jos mekaaninen anturi 10,2 vaikuttuu, asetetaan hissi 10,4 ala-
asentoon.
                                                nodeoff(1,10,4);

                                        if(timedelay(22,4000000))
//Ajastin laskee 4000ms, jonka jälkeen tila[22]=0.
                                                tila[22]=0;
                                break;

                    }
                    if(timedelay(7,40000000)){ //Ajastin laskee
40000ms, jonka jälkeen jälkeen hissi 10,4 asetetaan ala-asentoon ja
stoppari 10,3 menee päälle, estäen seuraavan paletin kulun.  Lisäksi
tila[7] siirtyy seuraavaan caseen.
                                nodeoff(1,10,4);
```

```
                                    nodeoff(1,10,3);
                                    tila[7]++;
                                    }
            break;

            case 2:
                        strcpy(tilan_viesti[7],"8.3");
                        if(nodestate(1,8,3))  //Jos induktiivinen anturi
8,3 vaikuttuu, siirtyy tila[7] seuraavaan caseen.
                        tila[7]++;

            break;

            case 3:
                        strcpy(tilan_viesti[7],"tila[16]");
                        if(tila[16]==3)       //Jos tila[16]=3, siirtyy
tila[7] seuraavaan caseen.
                                    tila[7]++;
            break;

            case 4:
                        strcpy(tilan_viesti[7],"10.1");
                        nodeon(1,10,3);       //Stoppari 10,3 menee päälle,
päästäen paletin ylitseen.
                        if(nodestate(1,10,1)) //Jos induktiivinen anturi
10,1 vaikuttuu, tila[7] siirtyy seuraavaan caseen.
                                    tila[7]++;
            break;

            case 5:
                        strcpy(tilan_viesti[7],"10.2");
                        nodeon(1,10,4);       //Hissi 10,4 asetetaan
yläasentoon.
                        nodeoff(1,10,3);      //Stoppari 10,3 menee pois
päältä, estäen seuraavan paletin kulun.

                        if(nodestate(1,10,2)) //Jos mekaaninen anturi 10,2
vaikuttuu, siirtyy tila[7] seuraavaan caseen.
                                    tila[7]++;
            break;

            case 6:
                        strcpy(tilan_viesti[7],"ajastin 1s");
                        nodeoff(1,10,4);      //Hissi 10,4 asetetaan ala-
asentoon.
                        if(timedelay(7,1000000)) //Ajastin laskee 1000ms,
jonka jälkeen tila[7]=2.
                                    tila[7]=2;
            break;
}

switch(tila[8]){
            //case 0-1 poistaa jäämäpaletit aluksi ennen kuin tila[7]
aloittaa normaalin työkierron
            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
```

```
                    //Tilakone tila[23] ohjaa jäämäpalettien poistoa
ajon alussa tässä casessa.
                    switch(tila[23]) {    //Tämä tilakone ohjaa
tila[8]:n jäämäpalettien poistoa.

                                case 0:
                                        nodeon(1,5,3);
          //Stoppari 5,3 menee päälle, päästäen paletin ylitseen.
                                        nodeon(1,5,4);
          //Hissi 5,4 asetetaan yläasentoon.
                                        tila[23]++;
          //Tila[23] siirtyy seuraavaan caseen.
                                break;

                                case 1:
                                        if(nodestate(1,4,3))
//Jos induktiivinen anturi 4,3 vaikuttuu, menee stoppari 5,3 pois päältä,
estäen seuraavan paletin kulun.
                                                nodeoff(1,5,3);
                                        if(nodestate(1,4,4)){
//Jos mekaaninen anturi 4,4 vaikuttuu, asetetaan hissi 5,4 ala-asentoon
ja tila[23] siirtyy seuraavaan caseen.
                                                nodeoff(1,5,4);
                                                tila[23]++;
                                                }
                                        if(timedelay(23,4000000))
//Ajastin laskee 4000ms, jonka jälkeen tila[23] siirtyy seuraavaan
caseen.
                                                tila[23]++;
                                break;

                                case 2:
                                        if(timedelay(23,4000000)){
//Ajastin laskee 4000ms, jonka jälkeen hissi 5,4 asetetaan yläasentoon ja
stoppari 5,3 menee päälle, päästäen paletin ylitseen. Lisäksi tila[23]
siirtyy seuraavaan caseen.
                                                nodeon(1,5,4);
                                                nodeon(1,5,3);
                                                tila[23]++;
                                                }
                                break;

                                case 3:
                                        if(nodestate(1,4,3))
//Jos induktiivinen anturi 4,3 vaikuttuu, menee stoppari 5,3 pois päältä,
estäen seuraavan paletin kulun.
                                                nodeoff(1,5,3);
                                        if(nodestate(1,4,4))
//Jos mekaaninen anturi 4,4 vaikuttuu, asetetaan hissi 5,4 ala-asentoon.
                                                nodeoff(1,5,4);

                                        if(timedelay(23,4000000))
//Ajastin laskee 4000ms, jonka jälkeen tila[23]=0.
                                                tila[23]=0;
                                break;

                    }
                    if(timedelay(8,40000000)){ //Ajastin laskee
40000ms, jonka jälkeen hissi 5,4 asetetaan ala-asentoon ja stoppari 5,3
menee päälle, estäen seuraavan paletin kulun.  Lisäksi tila[8] siirtyy
seuraavaan caseen.
```

```
                                    nodeoff(1,5,4);
                                    nodeoff(1,5,3);
                                    tila[8]++;
                                    }
                        break;

            case 2:
                        strcpy(tilan_viesti[8],"29.1");
                        if(nodestate(1,29,1)) tila[8]++;
//Jos induktiivinen anturi 29,1 vaikuttuu, tila[8] siirtyy seuraavaan
caseen.
            break;

            case 3:
                        strcpy(tilan_viesti[8],"4.3");
                        nodeon(1,5,3);
//Stoppari 5,3 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,4,3))
//Jos induktiivinen anturi 4,3 vaikuttuu, siirtyy tila[8] seuraavaan
caseen.
                                    tila[8]++;
            break;

            case 4:
                        strcpy(tilan_viesti[8],"4.4");
                        nodeoff(1,5,3);
//Stoppari 5,3 menee poispäältä, estäen seuraavan paletin kulun.
                        nodeon(1,5,4);
//Hissi 5,4 asetetaan yläasentoon.
                        if(nodestate(1,4,4) && tila[18]==3)
            //Jos mekaaninen anturi 4,4 vaikuttuu ja tila[18]=3, tila[8]
siirtyy seuraavaan caseen.
                                    tila[8]++;
            break;

            case 5:
                        strcpy(tilan_viesti[8],"ajastin 1400ms");
                        nodeoff(1,5,4);
//Hissi 5,4 asetetaan ala-asentoon.
                        if(timedelay(8,1400000)) tila[8]=2;
//Ajastin laskee 1400ms, jonka jälkeen tila[8]=2.
            break;
}

switch(tila[9]) {
            case 0:
                        nodeoff(1,14,2);
//Hissi 14,2 asetetaan ala-asentoon.
            break;

            case 1:
                        strcpy(tilan_viesti[9],"13.2, tila[15]");
                        if(nodestate(1,13,2) && tila[15]==0)
//Jos mekaaninen anturi 13,2 vaikuttuu ja tila[15]=0, siirtyy tila[9]
seuraavaan caseen.
                                    tila[9]++;
            break;

            case 2:
                        strcpy(tilan_viesti[9],"ajastin 1500ms");
                        nodeoff(1,14,2);
```

```
//Hissi 14,2 asetetaan ala-asentoon.
                        if(timedelay(9,2000000)){
//Ajastin laskee 2000ms, jonka jälkeen tila[9]=0.
                                    tila[9]=0;
                        }
            break;
}

switch(tila[10]) {
            case 0:
                        strcpy(tilan_viesti[10],"31.4");
                        nodeoff(1,30,3);
//Hissi asetetaan keskiasentoon.
                        nodeoff(1,30,4);
                        if(nodestate(1,31,4))
//Jos mekaaninen anturi 31,4 vaikuttuu, siirtyy tila[10] seuraavaan
caseen.
                                    tila[10]++;
            break;

            case 1:
                        strcpy(tilan_viesti[7],"ajastin 400ms");

                        if(timedelay(10,800000)) tila[10]++;
//Ajastin laskee 800ms paletin asettumiseksi hissille, minkä jälkeen
tila[10] siirtyy seuraavaan caseen.

            break;

            case 2:
                        strcpy(tilan_viesti[10],"31.4, tila[8]");
                        if(nodestate(1,31,4)) { //Jos mekaaninen anturi
31,4 vaikuttuu ja tila[8]=2, lukee RFID-lukija[3] paletin tagin.
                                    if(tila[8]==2 &&
p[lukija[3]].kohde()==3)  {
                                                resettimedelay(10);
                                                tila[10]=3;
                                                //testiä:
                                                p[lukija[3]].sijainti = 3;
            //paletti otetaan robolle, joten sijainti asetetaan 3.
                                    }
                                    else if((lukija[3]!=0 && p[ lukija[3]
].kohde()!=3) || timedelay(10,2000000)) {   //Jos reitti paletille on 3
eli SR60-asema, niin tila[10]=3.
                                                resettimedelay(10);
                                                tila[10]=4;
                                    }
                        }
            break;

            case 3:
                        //Tämä case ohjaa paletin SR60-robotille.
                        nodeon(1,30,3);
//Hissi  asetetaan yläasentoon.
                        nodeoff(1,30,4);
                        nodeon(1,5,3);
//Stoppari 5,3 menee päälle, päästäen paletin ylitseen.
                        p[lukija[3]].otettu();
//Paletti kuitataan otetuksi.
                        tila[10]=5;
            //Tila[10]=5.
```

```
            break;

            case 4:
                        //Tämä case ohjaa paletin kiertoon.
                        nodeon(1,30,4);
//Hissi asetetaan ala-asentoon.
                        nodeoff(1,30,3);
                        tila[10]++;
            //Tila[10] siirtyy seuraavaan caseen.
            break;

            case 5:
                        strcpy(tilan_viesti[10],"ajastin 2s");
                        // odotetaan ajastimella että hissi tyhjenee
                        if(timedelay(10,4000000)) tila[10]=0;
//Ajastin laskee 4000ms hissin tyhjentymistä varten, minkä jälkeen
tila[10]=0.
            break;


}

switch(tila[11]) {

            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
                        strcpy(tilan_viesti[11],"13.1, tila[9]");
            //odotetaan että tila[9] ja tila[19] ovat tyhjiä ennekuin
päästetään paletti
                        if(nodestate(1,13,1) && tila[9]==0 && tila[19]==1)
{           //Jos induktiivinen anturi 13,1 vaikuttuu, tila[9]=0 ja
tila[19]=1, niin tila[9]=1 ja tila[11] siirtyy seuraavaan caseen.
                                    tila[9]=1;
                                    tila[11]++;
                        }
            break;

            case 2:
                        strcpy(tilan_viesti[11],"ajastin 500ms");
                        nodeon(1,14,1);
//Stoppari 14,1 menee päälle, päästäen paletin ylitseen.
                        nodeoff(1,14,2);
//Hissi 14,2 asetetaan ala-asentoon.
                        if(timedelay(11,500000)){
//Ajastin laskee 500ms, jonka jälkeen stoppari 14,1 menee pois päältä,
estäen seuraavan paletin kulun ja tila[11]=1.
                                    nodeoff(1,14,1);
                                    tila[11]=1;
                        }
            break;
}

switch(tila[12]) {
            //Hissin 21,4 lasku case 1:ssa on jäämäpaletteja varten
            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
```

```
            break;

            case 1:
                        strcpy(tilan_viesti[12],"28.2");
                        if(nodestate(1,21,2))
//Jos mekaaninen anturi 21,2 vaikuttuu, asetetaan hissi 21,4 ala-
asentoon.
                                    nodeoff(1,21,4);
                        if(nodestate(1,28,2))
//Jos induktiivinen anturi 28,2 vaikuttuu, siirtyy tila[12] seuraavaan
caseen.
                                    tila[12]++;
            break;

            case 2:
                        strcpy(tilan_viesti[12],"21.2");
                        if(nodestate(1,21,2)) {
//Jos mekaaninen anturi 21,2 vaikuttuu, asetetaan hissi 21,4 ala-asentoon
ja tila[12] siirtyy seuraavaan caseen.
                                    nodeoff(1,21,4);
                                    tila[12]++;
                        }
            break;

            case 3:

                        strcpy(tilan_viesti[12],"ajastin 2s");
                        if(timedelay(12,2000000))  tila[12]=1;
//Ajastin laskee 2000ms, jonka jälkeen tila[12]=1.
            break;
}

switch(tila[13]) {


            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
            //odotetaan että tila[14] sekä tila[17] ovat tyhjiä ennen kuin
päästetään paletti
                        strcpy(tilan_viesti[13],"31.1, tila[14]");
                        if(nodestate(1,31,1) && tila[14]==2 && tila[17]==1)
{ //Jos induktiivinen anturi 31,1 vaikuttuu, tila[14]=2 ja tila[17]=1,
niin tila[14]=3 ja tila[13] siirtyy seuraavaan caseen.
                                    tila[14]=3;
                                    tila[13]++;
                        }
            break;

            case 2:
                        strcpy(tilan_viesti[13],"ajastin 500ms");
                        nodeon(1,29,3);
//Stoppari 29,3 menee päälle, päästäen paletin ylitseen.
                        nodeoff(1,30,1);
//Hissi asetetaan keskiasentoon.
                        nodeoff(1,30,2);
                        if(timedelay(13,500000)) tila[13]++;
//Ajastin laskee 500ms, jonka jälkeen tila[13] siirtyy seuraavaan caseen.
```

```
            break;

            case 3:
                        nodeoff(1,29,3);
//Stoppari 29,3 menee pois päältä, estäen seuraavan paletin kulun.
                        tila[13]=1;
            //Tila[13]=1.
            break;

}

switch(tila[14]) {
            //caset 0-1 ovat jäämäpalettien poistoa varten ajon alussa
            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
                        if(nodestate(1,31,2) && tila[10]==0){
//Jos mekaaninen anturi 31,2 vaikuttuu ja tila[10]=0, niin hissi
asetetaan ala-asentoon.
                                    nodeoff(1,30,1);
                                    nodeon(1,30,2);
                                    }
                        if(timedelay(14,3000000))//Ajastin laskee 3000ms,
jonka jälkeen tila[14] siirtyy seuraavaan caseen.
                                    tila[14]++;
            break;

            case 2:
                        nodeoff(1,30,1);
//Hissi asetetaan keskiasentoon.
                        nodeoff(1,30,2);
            break;

            case 3:
                        strcpy(tilan_viesti[14],"31.2, tila[10]");
                        if(nodestate(1,31,2) && tila[10]==0)
//Jos mekaaninen anturi 31,2 vaikuttuu ja tila[10]=0, niin tila[14]
siirtyy seuraavaan caseen.
                                    tila[14]++;
            break;

            case 4:
                        strcpy(tilan_viesti[14],"ajastin 2s");
                        nodeoff(1,30,1);
//Hissi asetetaan ala-asentoon.
                        nodeon(1,30,2);
                        if(timedelay(14,2000000))
//Ajastin laskee 2000ms, jonka jälkeen tila[14]=2.
                                    tila[14]=2;
            break;
}

switch(tila[15]) {
            case 0:
                        strcpy(tilan_viesti[15],"12.1");
                        nodeon(1,12,3);
//Stoppari 12,3 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,12,1))
```

```
//Jos induktiivinen anturi 12,1 vaikuttuu, siirtyy tila[15] seuraavaan
caseen.
                                    tila[15]++;
            break;

            case 1:
                        strcpy(tilan_viesti[15],"12.2");
                        nodeoff(1,12,3);
//Stoppari 12,3 menee pois päältä, estäen seuraavan paletin kulun.
                        if(nodestate(1,12,2))
//Jos induktiivinen anturi 12,2 vaikuttuu, on tila[15]=0.
                                    tila[15]=0;
            break;
}

switch(tila[16]) {
            //Case 0…2 poistaa jäämäpaletit ennenkuin tila[16] aloittaa
normaalin työkierron.
            case 0:
                                    nodeon(1,11,3);
//Stopparit 11,3 ja 11,4 menevät päälle, päästäen paletin ylitseen.
                                    nodeon(1,11,4);
                                    nodeon(1,5,2);
//Hissi 5,2 asetetaan yläasentoon.

            break;

            case 1:
            //Tila[20] ohjaa hissiä 5.2 jäämäpaletteja poistettaessa ajon
alussa.
                                    switch(tila[20]) {
//Tämä tilakone ohjaa tila[16]:ta jäämäpalettien poistoa.

                                    case 0:

                                                if(timedelay(20,4500000)){
//Ajastin laskee 4500ms, jonka jälkeen hissi 5,2 asetetaan ala-asentoon
ja stoppari 5,1 menee päälle, päästäen paletin ylitseen. Lisäksi tila[20]
siirtyy seuraavaan caseen.

            resettimedelay(20);
                                                            nodeoff(1,5,2);
                                                            nodeon(1,5,1);
                                                            tila[20]++;
                                                            }
                                                if(nodestate(1,4,2))
//Jos mekaaninen anturi 4,2 vaikuttuu, asetetaan hissi 5,2 yläasentoon.
                                                            nodeon(1,5,2);


                                    break;

                                    case 1:
            //suljetaan tie seuraavalta paletilta, jotta hissillä oleva
paletti ei puristuisi
                                                if(timedelay(25,500000)){
            //Ajastin laskee 500ms, jonka jälkeen stoppari 5,1 menee pois
päältä estäen seuraavan paletin kulun. Lisäksi tila[20]=0.

            resettimedelay(25);
                                                            nodeoff(1,5,1);
```

```
                                                tila[20]=0;
                                                }


                                    break;
                        }

                        if(timedelay(16,40000000)) //Ajastin laskee
40000ms, jonka aikana jäämäpalettien poisto tapahtuu, minkä jälkeen
tila[16] siirtyy seuraavaan caseen.
                                    tila[16]++;
            break;

            case 2:
                                    nodeoff(1,11,3);
//Stopparit 11,3, 11,4 ja 5,1 menevät pois päältä, estäen seuraavan
paletin kulun.
                                    nodeoff(1,11,4);
                                    nodeoff(1,5,1);
                                    nodeoff(1,5,2);
//Hissi 5,2 asetetaan ala-asentoon.
                                    tila[16]=3;
            //Tila[16]=3.
            break;

            case 3:
                        strcpy(tilan_viesti[16],"11.1");
                        nodeon(1,11,3);
            //Stoppari 11,3 menee päälle, päästäen paletin ylitseen.
                        if(nodestate(1,11,1))
//Jos induktiivinen anturi 11,1 vaikuttuu, tila[16] siirtyy seuraavaan
caseen.
                                    tila[16]++;
            break;

            case 4:
                        strcpy(tilan_viesti[16],"11.2");
                        nodeoff(1,11,3);
//Stoppari 11,3 menee pois päältä, estäen seuraavan paletin kulun.
                        if(nodestate(1,11,2))
//Jos induktiivinen anturi 11,2 vaikuttuu, siirtyy tila[16] seuraavaan
caseen.
                                    tila[16]++;
            break;
//robotin toiminta:
            case 5:
                        //Viive että robotti ehtii toimia
                        strcpy(tilan_viesti[16],"ajastin 2s");
//testiä:
//onko tila 11.2 aktiivinen, ts. onko paletti tullut noodille
                        if(nodestate(1,11,2)) {
//koska robotti ei tiedä mikä paletti tulee pisteelle, niin katsotaan
//millä paletilla sijainti = 2, sijainti asetetaan tila[6]:ssa

                                    for (int i = 0; i <= 9; i++) {
                                                if (p[i].sijainti == 2) {
//paletti on nyt robolla.

//kommentoi if-lause pois jos et halua että paletti pysähtyy robolle:
```

```
                                                  if (nodestate(2,
2, 1)) {

//kun sr6:n työvaihe on valmis niin se kuittaa

//noodin 2,2,1 päälle. Jolloin stopperi pois päältä ja kasvatetaan tilaa
eteenpäin.
                                                                  nodeoff
(1, 11, 2);

           tila[16]++;
                                                  }
                                        }
                              }

                              /* vanha timerillä toimiva setti:
                              for (int i = 0; i <= 9; i++) {
                                        if (p[i].sijainti == 2) {
                                                  if (timedelay(16,
33000000)) {
                                                  //nodeoff(1, 11,
2);
                                                  tila[16]++;
                                                  }
                                        }
                              }*/

                    }

/*                  if(timedelay(16,2000000)) //2000ms
                              tila[16]++;           */
           break;

           case 6:
                    strcpy(tilan_viesti[16],"4.1");
                    nodeon(1,11,4);
//Stoppari 11,4 menee päälle, päästäen paletin ylitseen.
                    nodeoff(1,5,2);
//Hissi asetetaan ala-asentoon.
                    nodeon(1,5,1);
//Stoppari 5,1 menee päälle, päästäen paletin ylitseen.
                    if(nodestate(1,4,1))
//Jos 4,1 on vaikuttunut, tila[16] siirtyy seuraavaan caseen.
                              tila[16]++;
           break;

           case 7:
                    strcpy(tilan_viesti[16],"4.2, tila[17]");
                    nodeoff(1,11,4);
//Stopparit 11,4 ja 5,1 menevät pois päältä, estäen seuraavan paletin
kulun.
                    nodeoff(1,5,1);
                    if(nodestate(1,4,2) && tila[17]==1)
           //Jos mekaaninen anturi 4,2 vaikuttuu ja tila[17]=1, siirtyy
tila[16] seuraavaan caseen.
                              tila[16]++;
           break;

           case 8:
                    strcpy(tilan_viesti[16],"ajastin 1400ms");
                    nodeon(1,5,2);
```

```
//Hissi 5,2 asetetaan yläasentoon.
                    if(timedelay(16,1400000)) tila[16]=3;
//Ajastin laskee 1400ms, jonka jälkeen tila[16]=3.
          break;
}

switch(tila[17]) {

          case 0:
          //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
          break;

          case 1:
                    strcpy(tilan_viesti[17],"31.1, tila[14]");
                    if(nodestate(1,31,3) && tila[14]==2)
//Jos induktiivinen anturi 31,3 aktivoituu ja tila[14]=2, siirtyy
tila[17] seuraavaan caseen.
                              tila[17]++;
          break;

          case 2:
                    strcpy(tilan_viesti[17],"31.2");
                    nodeon(1,29,4);
//Stoppari 29,4 menee päälle, päästäen paletin ylitseen.
                    nodeon(1,30,1);
//Hissi asetetaan yläasentoon.
                    nodeoff(1,30,2);
                    tila[14]=3;
          //Tila[14]=3.
                    if(timedelay(17,500000)){
          //Ajastin laskee 500ms, jonka jälkeen stoppari 29,4 menee pois
päältä, estäen seuraavan paletin kulun ja tila[17] siirtyy seuraavaan
caseen.
                              nodeoff(1,29,4);
                              tila[17]++;
                              }
          break;

          case 3:

                    if(nodestate(1,31,2))
//Jos mekaaninen anturi 31,2 vaikuttuu, on tila[17]=1.
                              tila[17]=1;
          break;


}

switch(tila[18]) {

          //Case 0…2 poistaa jäämäpaletit ennen kuin tila[18] aloittaa
normaalin työkierron.
          case 0:
                              nodeon(1,18,3);
          //Stopparit 18,3 ja 18,4 menevät päälle, päästäen paletin
ylitseen.
                              nodeon(1,18,4);
                              nodeon(1,15,4);
//Hissi 15,4 asetetaan yläasentoon.
          break;
```

```
            case 1:
            //Tila[21] ohjaa hissiä 15.4, jäämäpaletteja poistettaessa
ajon alussa tässä casessa.
                                    switch(tila[21]) {
//Tämä tilakone ohjaa tila[18]:ta jäämäpalettien poistoa.

                                    case 0:
                                                        if(timedelay(21,4500000)){
            //Ajastin laskee 4500ms, jonka jälkeen hissi 15,4 asetetaan
ala-asentoon ja stoppari 15,3 menee päälle, päästäen paletin ylitseen.
Lisäksi tila[21] siirtyy seuraavaan caseen.

            resettimedelay(21);
                                                                    nodeoff(1,15,4);
                                                                    nodeon(1,15,3);
                                                                    tila[21]++;
                                                                    }
                                                        if(nodestate(1,15,2))
//Jos mekaaninen anturi 15,2 vaikuttuu, asetetaan hissi 15,4 yläasentoon.
                                                                    nodeon(1,15,4);


                                    break;

                                    case 1:
                                    //suljetaan tie seuraavalta paletilta,
jotta hissillä oleva paletti ei puristu.
                                                        if(timedelay(21,500000)){
//Ajastin laskee 500ms, jonka jälkeen stoppari 15,3 menee pois päältä
estäen seuraavan paletin kulun. Lisäksi tila[21]=0.

            resettimedelay(21);
                                                                    nodeoff(1,15,3);
                                                                    tila[21]=0;
                                                                    }



                                    break;
                        }

                        if(timedelay(18,45000000))
//Ajastin laskee 45000ms, jonka aikana jäämäpalettien poisto tapahtuu.
Lisäksi tila[18] siirtyy seuraavaan caseen.
                                    tila[18]++;
                        break;


            case 2:
                        nodeoff(1,18,3);
//Stopparit 18,3, 18,4 ja 15,3 menevät pois päältä, estäen seuraavan
paletin kulun.
                        nodeoff(1,18,4);
                        nodeoff(1,15,3);
                        nodeoff(1,15,4);
//Hissi 15,4 asetetaan ala-asentoon.
                        tila[18]=3;
            //Tila[18]=3.
            break;
```

```
            case 3:
                        strcpy(tilan_viesti[18],"18.1");
                        nodeon(1,18,3);
//Stoppari 18,3 menee päälle, päästäen paletin ylitseen.
                        nodeoff(1,15,4);
//Hissi 15,4 asetetaan ala-asentoon.
                        if(nodestate(1,18,1))
//Jos induktiivinen anturi 18,1 aktivoituu, siirtyy tila[18] seuraavaan
caseen.
                                    tila[18]++;
            break;

            case 4:
                        //Viive robotin toimintaa varten
                        strcpy(tilan_viesti[18],"ajastin 2s");
                        nodeoff(1,18,3);
//Stoppari 18,3 menee pois päältä, estäen seuraavan paletin kulun.
//testiä:
                        //paletti robolla n+1 s.
                        if(nodestate(1, 18, 2)) {
                                    for (int i = 0; i <= 9; i++) {
            //tsekataan millä paletilla sijainti ==3
                                                if (p[i].sijainti == 3) {
            //sijainti asetetaan rfid:llä 3, tila[10].
                                                            if (timedelay(16,
33000000))

            tila[18]++;
                                                }
                                    }
                        }

                        /*if(timedelay(18,2000000)) //Ajastin laskee
2000ms, jonka jälkeen tila[18] siirtyy seuraavaan caseen.
                                    tila[18]++;
                        */
            break;

            case 5:
                        strcpy(tilan_viesti[18],"15.2");
                        nodeon(1,18,4);
//Stoppari 18,4 ja 15,3 menevät päälle, päästäen seuraavan paletin
ylitseen.
                        nodeon(1,15,3);
                        if(nodestate(1,15,2) && tila[19]==1)
//Jos mekaaninen anturi 15,2 aktivoituu ja tila[19]=1, siirtyy tila[18]
seuraavaan caseen.
                                    tila[18]++;
            break;

            case 6:
                        strcpy(tilan_viesti[18],"ajastin 2s");
                        nodeon(1,15,4);
//Hissi 15,4 asetetaan yläasentoon.
                        nodeoff(1,18,3);
//Stopparit 18,3, 18,4 ja 15,3 menevät pois päältä, estäen seuraavan
paletin kulun.
                        nodeoff(1,18,4);
                        nodeoff(1,15,3);

                        if(timedelay(18,2000000))
```

```
//Ajastin laskee 2000ms, jonka jälkeen tila[18]=3.
                                tila[18]=3;
            break;
}

switch(tila[19]) {

            case 0:
            //Tämä tyhjä case on anturin toiminnan parantamiseksi, jos
anturilla on paletti ajon alussa.
            break;

            case 1:
                        strcpy(tilan_viesti[19],"13.3, tila[9]");
                        if(nodestate(1,13,3) && tila[9]==0) {
//Jos induktiivinen anturi 13,3 aktivoituu ja tila[9]=0, niin tila[19]
siirtyy seuraavaan caseen.
                                tila[19]++;
                        }
            break;

            case 2:
                        strcpy(tilan_viesti[19],"ajastin 500ms");
                        nodeon(1,14,3);
//Stoppari 14,3 aktiovoituu, päästäen paletin ylitseen.
                        nodeon(1,14,2);
//Hissi 14,2 asetetaan yläasentoon.
                        tila[9]=1;                      //Tila[9]=1
                        if(timedelay(19,500000)){
//Ajastin laskee 500ms, jonka jälkeen stoppari 14,3 menee pois päältä,
estäen seuraavan paletin kulun ja tila[19]=1.
                                nodeoff(1,14,3);
                                tila[19]=1;
                        }
            break;

            case 3:
                        strcpy(tilan_viesti[19],"ajastin 1500ms");
                        if (nodestate(1,13,2))
//Jos mekaaninen anturi 13,2 vaikuttuu, niin tila[19]=1.
                                tila[19]=1;
            break;
}



// Printataan tilakoneiden ja palettien tilat
VT100::goxy(0,0);

for(int i=0;i<=19;i++)
            printf("Tila[%d]: %d  ,Viesti :%s\n", i,
tila[i],tilan_viesti[i]);

for(int i=1;i<=12;i++)
            printf("Paletti[%d]:  Kohde: %d  Kierrokset: %d Aktiivinen
:%i\n",i,p[i].kohde(),p[i].kierros(),p[i].aktiivisuus());

// Printataan RFID-lukijoiden tilat
for(int i=1;i<=3; i++) printf("Lukija[%d]: %d \n",i, lukija[i]);

// Printataan ohjelman nopeus
```

```
if(!timedelay(31,1000000)) laskuri++;
else { printf("%d kierrosta/s\n",laskuri); laskuri=0;}

}
// Hoidetaan CAN-kommunikaatio
// Lähetetään kaikki RPDOt joiden tilat ovat muuttuneet
can.send_only_changed_rpdos();
can.sync();

// Käydään lukemassa kaikki meille lähtetyt CAN viestit.
// Nukumme hetken kierrosten välissä jos ei useita samoja TDPO paketteja.
// Jos on samoja käsittelemme ne heti uudestaan jotta eivät jää puskuriin
ja käsittelemättä.
can.read_all_messages();
u_sleep(4000);

}
while(!key(VK_ESCAPE)); // ESC näppäimen painallus lopettaa ohjelman
suorituksen


VT100::clearscreen();
printf("\nSystem Shutdown !\n");
printf("Wait a few seconds for all devices to go down !\n");

// Ohjelma lopetaan.
// Asetetaan kaikki outputit nollatilaan.
// Nollataan kaikki outputit yksi kerrallaan pienin väliajoin jotta ei
tule sähköpiikkiä
for (int i=1 ; i<32 ; i++ )
          for (int j=1 ; j<5 ; j++ ) {
                    if (nodestate(1,i,j)) { // Jos nodeon päällä niin
                              nodeoff(1,i,j); // Node pois päältä
                              can.send_only_changed_rpdos();
                              u_sleep(100000);//ennen 100000
                    }
          }
u_sleep(10000);
can.stop_node(0); // Lähetetään CAN-väylälle stop-paketti kaikille
nodeille jolloin noodit menevät stopped tilaan

          } //if
          else
                    VT100::clearscreen();
                    printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\tYou ended the
program..");
                    u_sleep(1000000);


//} while (valinta != 9);
/*//ehkis maailman hienoin "efekti"
printf("\n\n You ended the production line program. The application will
now close");
for (int i = 0; i < 2; i++) {
          printf(".");
          u_sleep(1000000);
}*/
#ifdef LINUX
          VT100::terminalreset();
#endif // !LINUX
```

```
fclose(pFile);
//suljetaan avattu tiedosto

decoder_closelog();
}



// *********************
// Globaalit apufunkiot
// *********************

//testiä:
// Asetetaan palettien reitit
void reittien_alustus() {
        p[1].aktivoi();
        p[2].aktivoi();
        p[3].aktivoi();
        p[4].aktivoi();
        p[5].aktivoi();
        p[6].aktivoi();
        p[7].aktivoi();
        p[8].aktivoi();
        p[9].aktivoi();
        p[10].aktivoi();
        p[11].aktivoi();
        p[12].aktivoi();

        p[1].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[2].aseta_reitti(0,0,0,0,0,1,0,0,0,0);
        p[3].aseta_reitti(0,0,0,0,1,0,1,0,0,0);
        p[4].aseta_reitti(2,1,0,0,0,0,0,0,0,0);
        p[5].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[6].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[7].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[8].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[9].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[10].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[11].aseta_reitti(0,0,0,0,0,0,0,0,0,0);
        p[12].aseta_reitti(0,0,0,0,0,0,0,0,0,0);

//asetetaan robojen tyovaiheet:
           p[1].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[2].aseta_tyovaihe(0,0,0,0,1,0,0,1,0,0);
           p[3].aseta_tyovaihe(0,0,0,0,0,0,1,0,1,0);
           p[4].aseta_tyovaihe(1,2,3,2,0,0,0,0,0,0);
           p[5].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[6].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[7].aseta_tyovaihe(0,0,0,0,1,0,0,1,0,0);
           p[8].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[9].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[10].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[11].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);
           p[12].aseta_tyovaihe(0,0,0,0,0,0,0,0,0,0);


        p[1].aseta_kierrosten_maara(2);
        p[2].aseta_kierrosten_maara(2);
        p[3].aseta_kierrosten_maara(2);
        p[4].aseta_kierrosten_maara(1);
```

```
        p[5].aseta_kierrosten_maara(2);
        p[6].aseta_kierrosten_maara(2);
        p[7].aseta_kierrosten_maara(2);
        p[8].aseta_kierrosten_maara(1);
        p[9].aseta_kierrosten_maara(2);
        p[10].aseta_kierrosten_maara(2);
        p[11].aseta_kierrosten_maara(2);
        p[12].aseta_kierrosten_maara(2);

//laiskan miehen kierros lkm asetus, kommentoi jos et käytä
/*                  for (int i = 1; i <= 12; i++) {
                            p[i].aseta_kierrosten_maara(1);
                            //i++;
                    }*/
}

// ************************************************
// Funktiot Noodien tilan asettamiseen ja tutkimiseen
// ************************************************

//ottavat vastaan cannoodin (1 tai 2), slave, bitti

//asetetaan haluttu noodi päälle:
void nodeon(int CANnode, int slave, int bitti) {
          if (CANnode == 1) {
                    if ((slave >= 1 && slave <= 31) && (bitti >= 1 &&
bitti <= 4)) {
                              if (slave < 16) {
                                        if (slave &1)

          rpdo201.set_bit((slave - 1) * 4 + bitti - 1);
                                        else

          rpdo201.set_bit((slave + 1) * 4 + bitti - 1);
                              }
                              else {
                                        if (slave &1)

          rpdo301.set_bit((slave - 17) * 4 + bitti - 1);
                                        else

          rpdo301.set_bit((slave - 15) * 4 + bitti - 1);
                              }
                    }

          }
          else if (CANnode == 2) {
                    if ((slave >= 1 && slave <= 31) && (bitti >= 1 &&
bitti <= 4)) {
                              if (slave <16) {
                                        if (slave &1)

          rpdo202.set_bit((slave -1) * 4 + bitti - 1);
                                        else

          rpdo202.set_bit((slave + 1) * 4 + bitti - 1);
                              }
                              else {
                                        //outputteja 16-31 ei ole.
                              }
                    }
```

```
            }

}

//sammutetaan haluttu noodi:
void nodeoff(int CANnode, int slave, int bitti) {
            if (CANnode == 1) {
                        if((slave >= 1 && slave <= 31) && (bitti >= 1 &&
bitti <= 4)) {
                                    if (slave < 16) {
                                                if (slave &1)

            rpdo201.clear_bit((slave - 1) * 4 + bitti - 1);
                                                else

            rpdo201.clear_bit((slave + 1) * 4 + bitti - 1);
                                    }
                                    else {
                                                if (slave &1)

            rpdo301.clear_bit((slave - 17) * 4 + bitti - 1);
                                                else

            rpdo301.clear_bit((slave - 15) * 4 + bitti - 1);
                                    }
                        }
            }
            else if (CANnode == 2) {
                        if ((slave >= 1 && slave <= 31) && (bitti >= 1 &&
bitti <= 4)) {        //joku mättää jossain koska virhe tulee '>='
kohdasta..
                                    if (slave < 16) {
                                                if (slave &1)

            rpdo202.clear_bit((slave - 1) * 4 + bitti - 1);
                                                else

            rpdo202.clear_bit((slave + 1) * 4 + bitti - 1);
                                    }
                                    else {
                                                //outputteja 16-31 ei ole
                                    }
                        }
            }
}

//tarkistetaan halutun noodin inputin tila:
bool nodestate(int CANnode, int slave, int bitti) {
            bool retval;
            if (CANnode == 1) {
                        if (slave < 16) {
                                    if (slave &1)
                                                retval =
tpdo181.get_bit((slave - 1) * 4 + bitti - 1);
                                    else
                                                retval =
tpdo181.get_bit((slave + 1) * 4 + bitti - 1);
                        }
                        else {
                                    if (slave &1)
```

```
                                                retval =
tpdo281.get_bit((slave - 17) * 4 + bitti - 1);
                                        else
                                                retval =
tpdo281.get_bit((slave - 15) * 4 + bitti - 1);
                        }
                        return(retval);
            }
            else if (CANnode == 2) {
                        if (slave < 16) {
                                    if (slave &1)
                                                retval =
tpdo182.get_bit((slave - 1) * 4 + bitti - 1);
                                        else
                                                retval =
tpdo182.get_bit((slave + 1) * 4 + bitti - 1);
                        }
                        else {
                                    //inputteja 16-31 ei ole
                        }
                        return(retval);
            }
}


//!vanhat node-funktiot!!!! (ennen 15.4)

/*
// Asete3tetaan jokin noodi päälle
// Parametreina ASI slaven numero ja slaven outputin numero 1-4
void nodeon(int ASiMasterID,int slave, int bitti) {
            if(ASiMasterID==1)
            {
                        if((slave>=1 && slave<=31) && (bitti>=1 &&
bitti<=4)) {
                                    if(slave<16) {
                                                if(slave &1)

            rpdo201.set_bit((slave-1) * 4 + bitti -1);
                                                else

            rpdo201.set_bit((slave+1) * 4 + bitti - 1);
                                    }
                                    else {
                                                if(slave &1)

            rpdo301.set_bit((slave-17) * 4 + bitti - 1);
                                                else

            rpdo301.set_bit((slave-15) * 4 + bitti - 1);
                                    }
                        }
            }
            else if(ASiMasterID==2)
            {
                        if((slave>=1 && slave<=31) && (bitti>=1 &&
bitti<=4)) {
                                    if(slave<16) {
                                                if(slave &1)

            rpdo401.set_bit((slave-1) * 4 + bitti -1);
```

```
                                        else

            rpdo401.set_bit((slave+1) * 4 + bitti - 1);
                                }
                                else {      //kommentoi-->
                                        if(slave &1)

            rpdo302.set_bit((slave-17) * 4 + bitti - 1);
                                        else

            rpdo302.set_bit((slave-15) * 4 + bitti - 1);
                                        //<---kommentoi
                                }
                        }
            }
}
*/

/*
// Asetetetaan jokin noodi pois päältä
// Parametreina ASI slaven numero ja slaven outputin numero 1-4
void nodeoff(int ASiMasterID,int slave, int bitti) {
            if(ASiMasterID==1)
            {

                        if((slave>=1 && slave<=31) && (bitti>=1 &&
bitti<=4)) {
                                if(slave<16) {
                                        if(slave &1)

            rpdo201.clear_bit((slave-1) * 4 + bitti -1);
                                        else

            rpdo201.clear_bit((slave+1) * 4 + bitti - 1);
                                }
                                else {
                                        if(slave &1)

            rpdo301.clear_bit((slave-17) * 4 + bitti - 1);
                                        else

            rpdo301.clear_bit((slave-15) * 4 + bitti - 1);
                                }
                        }
            }
            else if(ASiMasterID==2)
            {

                        if((slave>=1 && slave<=31) && (bitti>=1 &&
bitti<=4)) {
                                if(slave<16) {
                                        if(slave &1)

            rpdo401.clear_bit((slave-1) * 4 + bitti -1);
                                        else

            rpdo401.clear_bit((slave+1) * 4 + bitti - 1);
                                }
                                else {      //kommentoi-->
                                        if(slave &1)
```

```
            rpdo302.clear_bit((slave-17) * 4 + bitti - 1);
                                        else

            rpdo302.clear_bit((slave-15) * 4 + bitti - 1);
                                                    //<---kommentoi
                                }
                        }
            }

}
*/
/*
// tarkistetaan halutun noodin inputin tila
// Parametreina ASI slaven numero ja slaven outputin numero 1-4
bool nodestate(int ASiMasterID,int slave, int bitti) {
            bool retval;
            if(ASiMasterID==1)
            {
                        if(slave<16) {
                                    if(slave &1)

            retval=tpdo181.get_bit((slave-1) * 4 + bitti -1);
                                    else

            retval=tpdo181.get_bit((slave+1) * 4 + bitti - 1);
                        }
                        else {
                                    if(slave &1)

            retval=tpdo281.get_bit((slave-17) * 4 + bitti - 1);
                                    else

            retval=tpdo281.get_bit((slave-15) * 4 + bitti - 1);
                        }
                        return(retval);
            }
 //vanha-->
            else if (ASiMasterID==2)
            {
                        if(slave<16)
                        {
                                    if(slave &1)

            retval=tpdo381.get_bit((slave-1) * 4 + bitti -1);
                                    else

            retval=tpdo381.get_bit((slave+1) * 4 + bitti - 1);
                        }
                        else { //kommentoi -->
                                    if(slave &1)

            retval=tpdo281.get_bit((slave-17) * 4 + bitti - 1);
                                    else

            retval=tpdo281.get_bit((slave-15) * 4 + bitti - 1); // <---
kommentoi
                        }
            return(retval);
            }//else if
}*/
```