Kari Saalasti

Hakepolttimen ohjauskeskus

Opinnäytetyö
Sähkötekniikan koulutusohjelma

Lokakuu 2016

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** **28.11.2016**
Tekijä(t) Kari Saalasti	**Koulutusohjelma ja suuntautuminen** **Sähkötekniikka**
Nimeke Hakepolttimen ohjauskeskus	

Tiivistelmä

Tässä opinnäytetyössä rakensin ohjauskeskuksen pienelle, noin 20kW hakepolttimelle. Yritin tehdä polttimen toiminnan mahdollisimman automaattiseksi ja vähentää tarvetta käsisäädölle. Samalla polttimen taloudellisuutta saatiin parannettua.

Jäännöshapen määrää mitattiin autokäyttöön tarkoitetulla halvalla, kapeakaistaisella happianturilla, joka osoittautui sopivaksi tähän käyttötarkoitukseen. Jäännöshapen mittauksen avulla poltin osasi mukautua polttoaineen laadun vaihteluihin, sitä voitiin käyttää pienemmällä ilmaylimäärällä ja polttoaineen syötön käsisäätö korvautui syöttöruuvin PID-säätimelle annettavalla ohjearvolla.

Taajuusmuuttajia ohjattiin Modbus kenttäväylällä, jolloin ohjauskaapelointi yksinkertaistui huomattavasti ja sen avulla niiden toiminnasta saatiin paljon tietoa, jota voi hyödyntää myöhemmin esimerkiksi polttimen toimintavarmuuden parantamisessa. Logiikka ohjelmoitiin Codesys 2 ohjelmointityökaluilla. Tietoja polttimen toiminnasta tallennettiin SQL-tietokantaan.

Asiasanat (avainsanat)

Automaatio lämmöntuotanto metsäenergia lämmitysvoimalat

Sivumäärä	**Kieli**
73	Suomi

Huomautus (huomautukset liitteistä)

Kaksi liitettä

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Teemu Manninen	

DESCRIPTION

MAMK
University of Applied Sciences

Date of the bachelor's thesis

28.11.2016

Author(s)

Kari Saalasti

Degree programme and option

Electrical engineering

Name of the bachelor's thesis

Control unit of wood chip burner

Abstract

In this thesis I built control unit for small, about 20kW wood chip burner. Aim was making burner operate automatically and reduce need for manual adjustments, while reducing fuel consumption.

Amount of residual oxygen was measured with narrow band automotive oxygen sensor, which proved to be suitable for this purpose. By measuring residual oxygen, burner was able to adapt in variations of fuel quality and it operated with less excess air. Manual adjustments were replaced with setpoint adjustment of PID-controller, which needed only small corrections by user.

Motor drives were controlled with Modbus RTU fieldbus, which made control cabling simpler. It also allowed logic controller to read various kind of information from drive, which could be later used for increasing reliability of burner. Programmable logic controller was programmed with Codesys 2 tools and information from burner control was stored in SQL-database.

Subject headings, (keywords)

Pages

73

Language

Finnish

Remarks, notes on appendices

Two appendices

Tutor

Teemu Manninen

Bachelor's thesis assigned by

SISÄLLYSLUETTELO

1 JOHDANTO

Puulla lämmittäminen on edullista, mutta usein se on myös työlästä. Hakelämmityksessä puuta voidaan polttaa enemmän tai vähemmän automatisoidusti ja hakkeen tekeminen on paljon nopeampaa, helpompaa ja usein myös edullisempaa, kuin polttopuiden tekeminen. Tässä opinnäytetyössä tutustutaan vanhaan, 90-luvulla rakennettuun lämpökeskukseen ja sen ongelmiin. Sen pohjalta yritetään kehittää nykyaikaisempi ohjauskeskus, joka on sopiva pienille polttimille ja toimii hyvin myös lauhalla säällä tarvittavan tehon ollessa pieni.

2 HAKELÄMMITYS

Puusta saadaan haketta, kun se silputaan hakkurilla (kuva 1). Se on karkeampaa, kuin sahajauho, palakoko voi olla muutamasta millimetristä ylöspäin useisiin kymmeniin millimetreihin. Toisin, kuin polttopuita, sitä voidaan siirtää ja syöttää polttimeen automaattisesti siirto- ja syöttöruuveilla.

KUVA 1. Lämmityskäyttöön sopivaa haketta

2.1 Taloudelliset edut ja haitat

Hakkeella lämmittäminen on edullista varsinkin, jos puita saadaan omasta metsästä. Vuonna 2014 energiapuun hinta oli 5€ kuutiometriltä pystykaupoissa, joissa ostaja huolehtii puun korjuusta ja kuljetuksesta. Hankintakaupassa hinta oli 22€ kuutiometriltä, jolloin metsänomistaja huolehtii puiden korjuusta ja toimittaa ne tien varteen. [1]

Poltettaessa kuutiometri kuivaa ja hyvälaatuista puuta, siitä saadaan noin 1000kWh lämpöä, joka vastaa 100 litraa kevyttä polttoöljyä tai hieman yli 100€:n edestä sähköä. Taloudellinen hyöty kuitenkin riippuu myös öljyn tai sähkön hinnasta, kuljetus- ja haketuskustannuksista, puun laadusta ja polttamisen hyötysuhteesta.

Hakelämmitys ei ole niin vaivaton, kuin öljylämmitys, sähkölämmitys tai vaikkapa maalämpöpumppu. Lämmittämiseen käytetty aika on pois vapaa-ajasta, sille jotkut laskevat suuremman arvon ja toiset taas pienemmän. Hakkeella lämmittämisen kannattavuus riippuu niin monista asioista, mukaan lukien ihmisten mieltymykset, ettei siitä voi tehdä tähän mitään laskelmaa.

2.2 Vanha lämpökeskus

Toisessa talossa on käytössä 90-luvulla rakennettu hakkeella toimiva lämmitysjärjestelmä. Tutustutaan tässä siihen ja samalla nähdään, minkälaisia koneita ja laitteita hakkeella lämmittämisessä voi tarvita. Tässä samassa luvussa on kerrottu myös hakkureista ja puiden kuljetuskalustosta, joita ei välttämättä tarvitse omistaa, mutta niillä saattaa säästää rahaa verrattuna niillä tehtävien töiden teettämiseen urakoitsijalla.

2.2.1 Puiden kuljetuskalusto

Puut on kuljetettava jollakin tavalla lämpökeskuksen lähelle ja se onnistuu järkevästi maatalous- tai metsätraktorilla. Kuvissa 2 ja 3 näkyvillä koneilla on aikoinaan korjattu metsistä puita ja tarvittaessa kuljetettu niitä. Nykyään puiden korjuu metsästä on jätetty metsäkoneurakoitsijoille ja näitä koneita on käytetty lähinnä puiden siirtämiseen paikasta toiseen tai nostureina. Kuljettamiseenkaan ei välttämättä tarvitse hankkia omia koneita, vaan senkin työn voi jättää urakoitsijan tehtäväksi.

KUVA 2. Maataloustraktori metsäperävaunulla ja kuormaajalla varustettuna

KUVA 3. Vanha Valmet 865AK metsätraktori 60-luvulta

2.2.2 Hakkuri

Kuvan 4 hakkurilla haketta tehty ohuemmista puista. Se ei pysty hakettamaan kovinkaan paksuja puun runkoja ja siitä puuttuu konesyöttö, jolloin puut pitää nostaa käsin

korkealle ja heittää hakkurin tuuttiin. Suurien hakemäärien tekeminen on sillä työlästä ja on se hieman vaarallinenkin. Sähkömoottorin käyttäminen traktorin sijaan olisi taloudellista ja kätevää, mutta hakkurin akselitehon tarve on useita kymmeniä kilowatteja, eikä sähköliittymien teho useimmiten riitä sen pyörittämiseen. Urakoitsijoilla on suuret ja tehokkaat kuormaajalla syötettävät hakkurit, jolloin vaivaa säästyy jättämällä puiden hakettaminen heidän tehtäväksi, eikä silloin hakkuria tarvitse lainkaan.

KUVA 4. Käsin syötettävä traktorikäyttöinen hakkuri

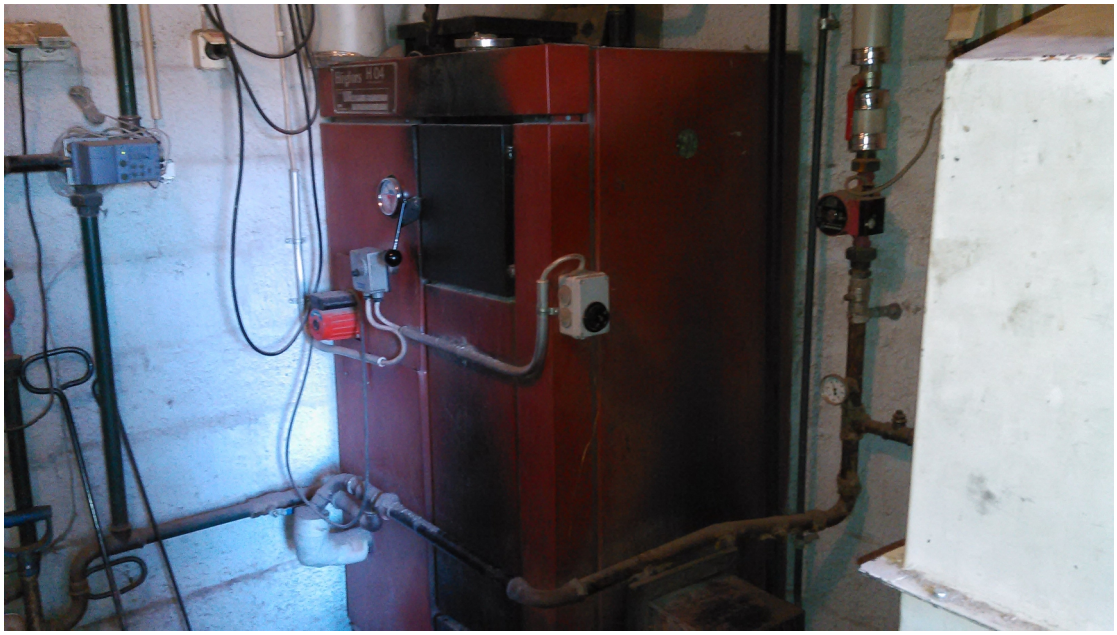
2.2.3 Hakevarasto

Pannuhuoneen yhteydessä on oltava varasto poltettavalle hakkeelle ja sinne olisi hyvä mahtua koko vuoden hakkeet. Kuvassa 5 näkyy osa vanhan lämpökeskuksen hakevarastosta ja stokeripolttimen säiliö, johon hake kuvassa näkyvien saavin ja lapion avulla kannetaan käsin. Kuiva hake pölyää melko runsaasti lapioitaessa ja sillä voi olla ikäviä vaikutuksia terveydelle. Varaston joutuu täyttämään kaksi tai kolme kertaa vuodessa. Jos varasto olisi suurempi, sen voisi täyttää keväällä tai kesällä, jolloin puut ovat kuivempia ja siten myös hake on kuivempaa.

KUVA 5. Vanhan lämpökeskuksen hakevarasto ja välineet hakkeen siirtämiseksi kuvan oikeassa reunassa näkyvään stokeriin

2.2.4 Pannuhuone

Vanhassa lämpökeskuksessa on kattilana 40kW tehoinen Högfors H04 (kuva 6), jonka kylkeen on tehty reikä hakepoltinta varten ja sisälle rakennettu tiilistä väliseinä, joka saa savukaasut kiertämään polttopuita varten olevan varastopesän kautta. Näin savukaasut kulkevat kattilan sisällä pidemmän matkan ja ehtivät luovuttamaan lämpöä enemmän ennen savuhormiin joutumista.



KUVA 6. Högfors H04 keskuslämmityskattila

Vanhan stokeripolttimen säiliöön (kuva 7) kannetaan saavilla haketta varastosta päivittäin tai joka toinen päivä riippuen lämmön tarpeesta. Säiliön pohjalla on ruuvi, joka siirtää hakkeen polttimelle ja pyörittää samalla syöttölautasta. Syöttölautanen on noin puoli metriä halkaisijaltaan oleva ratas, joka on laakeroitu säiliön pohjaan ja reunassa olevat hampaat kulkevat syöttöruuvin kierteiden välissä. Lautanen pyöriessään sekoittaa haketta säiliön pohjalla ja siirtää sitä syöttöruuville. Muuten ruuvi veisi hakkeen ympäriltään pois ja ennemmin tai myöhemmin ruuvi pyörisi tyhjiltään hakekasan alle muodostuneessa onkalossa. Tätä kutsutaan holvautumiseksi.

KUVA 7. Stokeripolttimen säiliö, josta hake siirretään polttimelle

Syöttöruuvi siirtää haketta putken sisällä (kuva 8). Ruuvin ja putken välissä pitää olla riittävä välys, jotta hake ei kiilautuisi tiukasti niiden väliin. Jos putken sisähalkaisija on esimerkiksi 100mm, sinne ei kannata laittaa 100mm ruuvia, vaan esimerkiksi 80mm ruuvi. Jos välys on liian pieni, ruuvi vaatii paljon voimakkaamman moottorin ja menee herkästi jumiin.

KUVA 8. Säiliön pohjasta lähtevä syöttöruuvin putki, jonka päässä on poltin

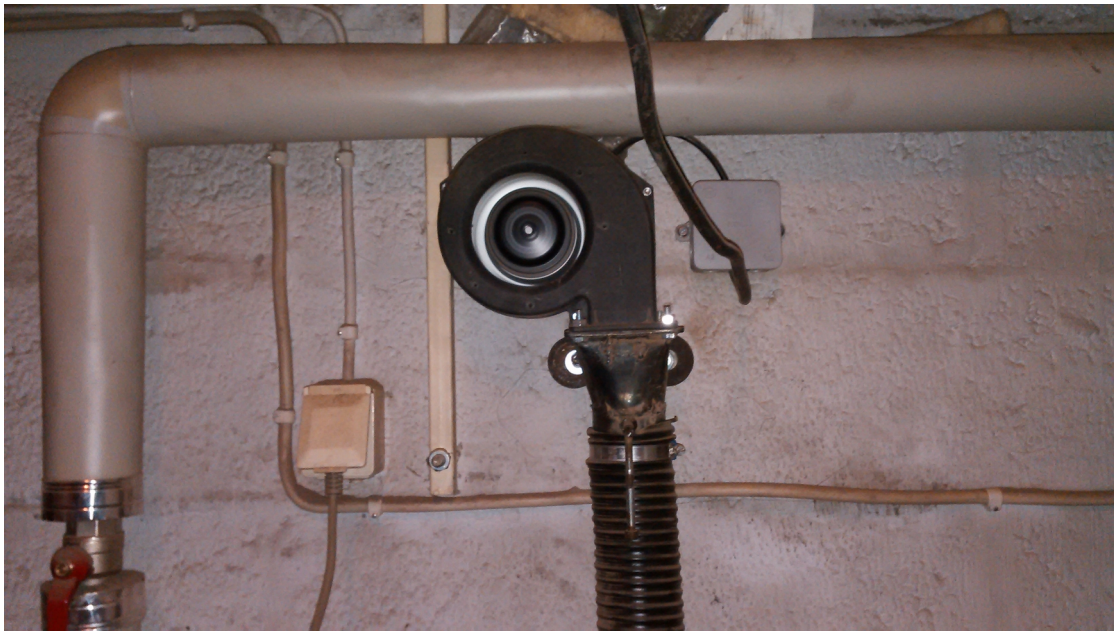
Kuvassa 9 näkyy vanha valurautainen palopää, joka on koteloitu polttimen sisälle. Ruuvin syöttämä hake menee palopäähän sen toisesta päästä ja palamisen tarvitsema ilma puhalletaan palopäähän poratuista pienistä rei'istä. Liekki ja kuumat savukaasut tulevat ulos palopään toisesta päästä. Palopää on suljettu tiiviiseen teräslaatikkoon, jolloin puhallin pystyy tekemään sinne paineen, jolla ilma puhalletaan ilmareikien läpi.

Kuvan palopää rikkoutui, kun hakkeen seassa yritettiin polttaa pilaantunutta viljaa. Rikkoutumisesta on kulunut jo paljon aikaa, joten tapauksen yksityiskohtia on enää vaikeaa muistaa, mutta viljasta saadaan paljon enemmän lämpöä, kuin hakkeesta ja sitä ehkä syötettiin liikaa polttimeen, jolloin palopää kuumeni liikaa ja halkeili yläreunastaan. Vilja muodostaa palaessaan paljon tuhkaa, joka ei lennä ilmanvirran mukana pois polttimesta, jolloin poltin täyttyy tuhkasta. Viljan polttaminen tällaisella polttimella on mahdollista, jos sitä sekoittaa pieniä määriä hakkeen sekaan.

KUVA 9. Vanha rikkinäinen palopää

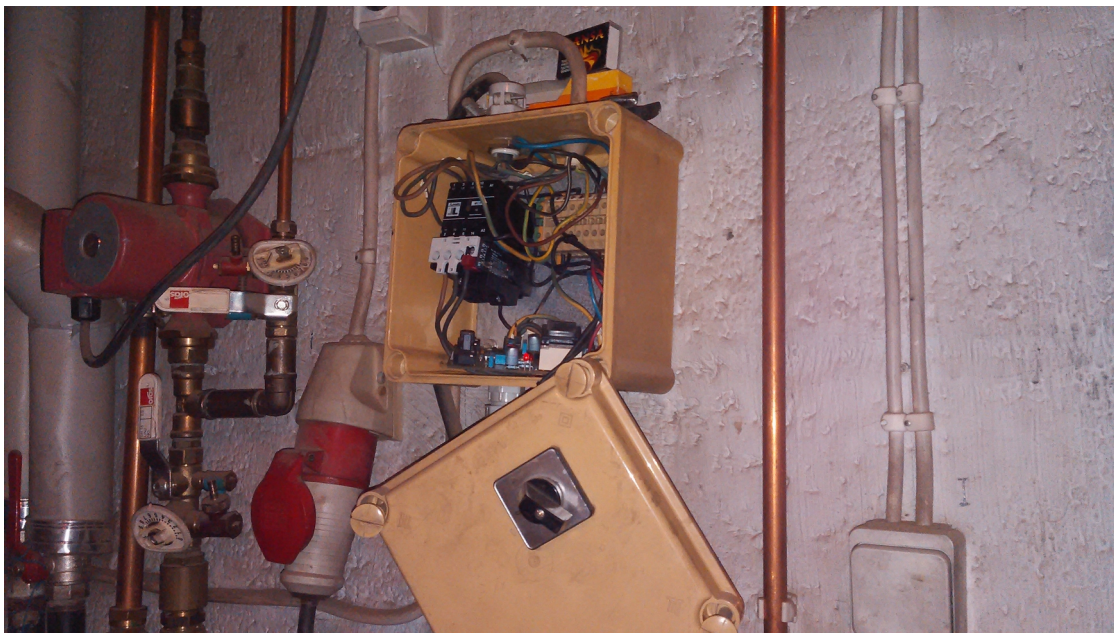
Polttimeen puhalletaan ilmaa kuvan 10 sähköpuhaltimella. Polttimen ilma otettiin alun perin lattian rajasta, mutta esimerkiksi pakkasella haketta lisättäessä poltin sai kylmää pakkasilmaa ja liekkivahdin termostaatti pysäytti polttimen. Korkeammalta puhallin imee lämminvesiputkien ja kattilan hukkalämmön lämmittämää ilmaa, jolloin osa siitä

saadaan kierrätettyä tätä kautta takaisin kattilaan ja pannuhuone pysyy hieman viileämpänä.



KUVA 10. Polttimen ilmapuhallin

Automatiikka on yksinkertainen (kuva 11), veden jäähdyttyä riittävästi mekaaninen termostaatti napsahtaa päälle ja kytkee virran ilmapuhaltimelle ja syöttöruuvin moottorin kontaktorille. Veden lämmettyä riittävästi puhallin ja syöttöruuvi pysäytetään. Polttimessa ei ole minkäänlaista automaattisytytystä, vaan siellä käyntijaksojen aikana hehkuu hiillos, joka sammuu tauon kestäessä liian pitkään. Siksi aikarele käynnistää polttimen hetkeksi tietyn ajan välein, jotta hiillos saadaan pidettyä yllä pidemmänkin aikaa.



KUVA 11. Polttimen releautomatiikka

Ruuvin pitää pyöriä melko hitaasti, jotta se ei syötä liikaa haketta, joten voima ruuvil-le välitetään kuvan 12 vivustolla. Ruuvin akselilla on vapaakytkin, joka välittää voi-maa ainoastaan pyöriessään toiseen suuntaan. Vaihdemoottori pyörii aina samalla no-peudella, joten syötön nopeutta voidaan säätää liikuttamalla akseleiden välillä olevaa tankoa eri kohtiin, jolloin vapaakytkimen vivun liike suurenee tai pienenee ja ruuvin nopeus muuttuu. Säätämällä ruuvin nopeutta saadaan muutettua ilman ja polttoaineen suhdetta.

KUVA 12. Syöttöruuvin voimansiirto

Termostaatin kotelossa on myös ylikuumenemissuoja, joka lauetessaan estää käynnin. Kuvassa 6 voi nähdä myös kattilan kyljessä toisen, liekkivahdin, termostaatin, jonka kapillaariputki menee polttimen koteloon ja estää käynnin polttimen jäähtyessä. Jos tuli polttimessa pääsee jostakin syystä sammumaan ruuvi syöttää haketta loputtomasti polttimeen ja se putoaa palamattomana kattilan tuhkatilaan. Lopulta hake täyttää kattilan ja sillä on valtava voima työntäessään kattilaa ja stokeria eroon toisistaan. Lopulta kattila, stokeri tai molemmat siirtyvät pois paikoiltaan ja hyvällä onnella stokerin vaihdemoottorin suojakytkin laukeaa ennen kuin suurempia vahinkoja tapahtuu.

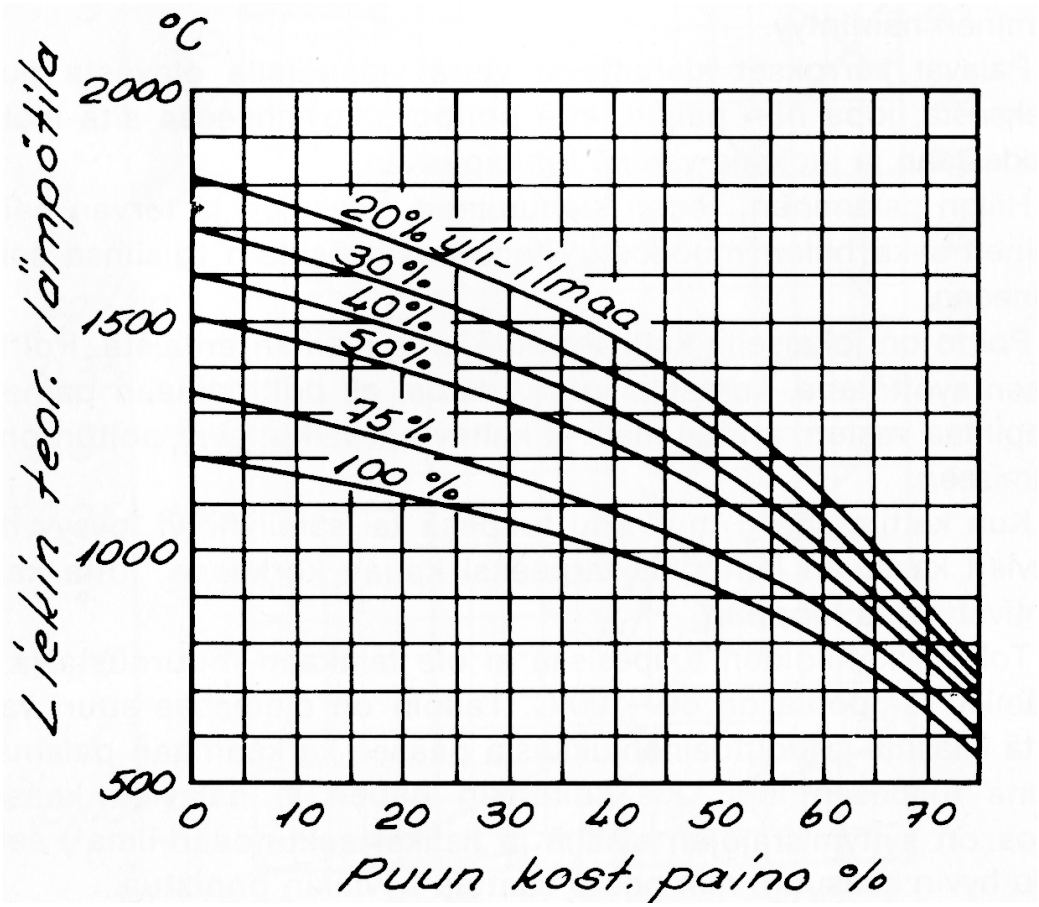
Tulet polttimeen sytytetään laittamalla hieman haketta polttimeen joko käsin tai käyttämällä syöttöruuvia. Sytykkeenä voi käyttää tuohia, paperia, sytytysnestettä tai mitä tahansa, millä puun saa parhaiten syttymään. Puhaltimen pistotulpan voi seuraavaksi laittaa semmoiseen pistorasiaan, mihin sähköä tulee jatkuvasti ja puhaltamalla ilmaa polttimeen hake alkaa palaa, jolloin pistotulpan voi laittaa takaisin termostaatin ohjaamaan pistorasiaan. Jos syöttöruuvin käynnistää ennen tulen syttymistä kunnolla, polttimeen menee haketta enemmän, kuin se ehtii polttamaan ja palava hake putoaa pois polttimesta sytykkeineen. Kun poltin on syttynyt riittävän hyvin, liekkivahdin termostaattia käännetään niin, että poltin käynnistyy ja tarkkaillaan poltinta hetken aikaa. Kun kaikki näyttää normaalilta, käännetään liekkivahdin termostaatti takaisin normaa-

liin asentoonsa ja poltin pysähtyy, koska polttimen kotelo ei ole vielä ehtinyt lämpenemään. Polttimessa hehkuva hake ja kuuma palopää alkaa pian lämmittämään liekkivahdin termostaattia ja poltin käynnistyy lopullisesti myöhemmin omia aikojaan. Poltin on pysäytettävä tällä tavalla, eikä liekkivahdin termostaattia voi jättää niin, että poltin jatkaa käyntiään, koska sen usein unohtaa myöhemmin käydä säätämässä sopivaksi.

3 UUDEN LÄMPÖKESKUKSEN SUUNNITTELU

Vanha hakepoltin on yksinkertainen, mutta sitä joutuu pitämään silmällä päivittäin ja säätelemään tarvittaessa hakkeen syöttöä sopivaksi. Ilmamäärän säätäminen sopivaksi tarkkailemalla liekkiä ja piipusta nousevaa savua on hankalaa. Erityisen hankalaksi sen tekee se, että erilaisia vaihtoehtoja polttimen säädöille on valtavasti ja useimmilla niistä poltin näyttäisi toimivan moitteettomasti. Parhaaksi keinoksi on osoittautunut kattilan seinämissä olevan tuhkan värin tarkastelu. Jos tuhka on väriltään tummaa, silloin siinä on ilmeisesti seassa palamatonta nokea ja hakkeen syöttö saattaa olla säädetty liian suureksi.

Puu tarvitsee happea palamiseen. Jos happea ei ole riittävästi, palaminen jää kesken ja silloin savupiipusta tupruaa savua. Savussa oleva terva ja muut aineet tarttuvat kattilaan ja savuhormiin, jolloin nokipalon riski kasvaa ja kattilan hyötysuhde pienenee. Palamatta jäänyt polttoaine menee hukkaan, saastuttaa ilmaa ja matalapaineella täyttää tienoon ikävällä savun hajulla, joka häiritsee itseä ja naapureita. Sopivalla määrällä ilmaa palaminen on puhdasta ja tehokasta, mutta jos polttimeen syötetään ilmaa liikaa, se laskee liekin lämpötilaa. Lämpötilan laskiessa hyötysuhde huononee ja lämpötilan laskiessa liikaa palaminen jää keskeneräiseksi, jolloin aiheutuu samoja ongelmia, kuin poltettaessa puuta liian vähäisellä ilmamäärällä. Kuvasta 13 nähdään ylimääräisen ilman vaikutus liekin lämpötilaan.



KUVA 13. Liekin teoreettinen palamislämpötila puun kosteudesta ja ilmaylimäärästä riippuvana [3].

Hakkeen syöttäminen polttimeen ei myöskään ole täysin ongelmatonta. Koska hake holvautuu helposti, ruuvi saattaa saada välillä vähemmän haketta ja polttimelle menevän hakkeen määrä vaihtelee. Kun haketta tehdään sekalaisesta puusta, sen laatu vaihtelee varaston eri kohdissa. Hakkurin puhaltaessa haketta varastoon raskaammat kappaleet, kuten kosteampi hake ja suuremmat palat lentävät varaston takanurkkaan ja kevyemmät lentävät lyhyemmän matkaa jääden varastossa eri kohtaan. Myös polttimen ilmareiät tukkeutuvat ja aukeavat satunnaisesti ja ilman määrä vaihtelee sen mukaan.

Uuteen lämpökeskukseen tarvitaan hakevarastoon tankopurkaimet ja siirtoruuvi siirtämään haketta varastosta, jolloin haketta ei tarvitse kantaa varastosta käsin, eikä sen pölylle tarvitse altistua. Poltin tarvitsee mittauksen jäännöshapelle ja taajuusmuuttajat joiden avulla hakkeen syöttöä ja ilman puhallusta voidaan säätää tarpeen mukaan.

4 UUDEN LÄMPÖKESKUKSEN RAKENTAMINEN

Talon kellarissa ei ollut tilaa pannuhuoneelle ja pienessä talossa senkin tilan mielellään käyttää muihin tarkoituksiin. Hakevarasto olisi pitänyt valaa maan pinnan alapuolelle ja siirtoruuvi varastosta tuoda talon anturan ali maan alla. Talon antura on tehty aikoinaan suurista kivistä ja kaivutyöt niiden ympärillä ja alla saattavat siirtää niitä paikoiltaan, jolloin rakennus vaurioituu.

Lämpökeskukselle rakennettiin kuvassa 14 näkyvä erillinen rakennus ja sieltä kaivettiin eristetty lämpökanaali taloon. Toinen kanaali kaivettiin lämpökeskuksen toisella puolella olevaan autotalliin, jolloin senkin lämmittäminen on tulevaisuudessa mahdollista. Pannuhuoneessa on riittävästi tilaa, jolloin nuohoaminen ja korjaustyöt ovat helppoja ja hakevarastoon mahtuu haketta koko vuodeksi.

KUVA 14. Vanhan navetan päätyyn rakennettu lämpökeskus

Hakevarasto (kuva 15) sijaitsee lämpökeskuksen ja vanhan navetan välissä. Liuku-

oven avaamalla voi tarkistaa jäljellä olevan hakkeen määrän, suuresta oviaukosta on helppoa puhaltaa hakkurilla haketta varastoon ja ovi estää lumen tuiskuamisen hakkeen sekaan.

KUVA 15. Liukuoven takana on hakevarasto

Kuvassa 16 näkyy hakevarastossa olevaa haketta. Toinen tankopurkaimista ei ole liikkunut lainkaan ja varasto on tyhjentynyt vain toiselta puolelta. Siksi on hyvä joskus tarkistaa tilanne varastossa ja samalla voi arvioida hakkeen kulutusta. Jumissa oleva purkain lähti liikkeelle sulkemalla hydrauliikkaöljyn virtaus toimivaan purkaimeen ja säätämällä painetta suuremmaksi. Aluksi purkaimen liikkeet olivat lyhyitä, mutta ajan kanssa se alkoi liikkua normaalisti. Toisenkin purkaimen voi taas ottaa käyttöön, kunhan hakkeen pinta varastossa on tasainen. Hieman kostea hake oli jäätynyt pinnasta kovaksi ja purkaimet poistivat haketta jäätyneen pinnan alapuolelta, jolloin toisen purkaimen toimimattomuutta ei nähnyt, ennen kuin pinnan alla oleva onkalo romahti. Samasta syystä varastoon yksin meneminen arveluttaa, koska yllättäen kasan alla olevaan onkaloon putoaminen vähintään säikäyttää ja se voi olla vaarallistakin, jos onkalo alkaa täyttymään romahduksen jälkeen.

KUVA 16. Kuva hakevaraston sisältä

Hakevaraston pohjalla ovat hydraulisylintereillä liikkuvat tankopurkaimet (kuva 17), jotka siirtävät haketta varaston pohjalta varastoruuville, joka siirtää hakkeen pannuhuoneessa olevaan pudotussuppiloon. Purkainten poikittaistangot ovat kiilamaiset, jolloin ne liikkuvat hakekasan alla helpommin toiseen suuntaan, mutta toiseen suuntaan liikkuessaan ne ottavat haketta mukaansa ja siirtävät sitä kohti ruuvia.

KUVA 17. Hakevaraston pohjalla olevat tankopurkaimet. Seinän alareunassa on rako, josta hake putoaa ruuville

Hakevaraston takaseinän alareunassa on rako, johon purkaimet työntävät haketta ja seinän takana on pellistä tehty laatikko, johon hake putoaa (kuva 18). Molemmilla purkaimilla on oma hydraulisylinterinsä ja ne liikkuvat vastakkaisiin suuntiin. Kun sylinterit saavuttavat liikkeensä ääripään, öljyn paine kasvaa ja hydraulipumpulla oleva venttiili vaihtaa liikkeen suunnan.

KUVA 18. Purkaimen sylinteri ja ruuvi, joka siirtää hakkeen pudotussuppiloon

Sylinterit ovat kiinnitetty toisesta päästään tukeviin teräspalkkeihin, jotka ovat valettu betoniin (kuva 19). Palkit ulottuvat betonin sisällä jonkun matkaa varastoon, jolloin ne kestävät sylintereiden tekemät melko suuret voimat. Hydraulipumpulla ei ole paine-mittaria, jonka perusteella sylinterien voiman voisi arvioida tarkemmin, mutta arviolta voima saattaa olla yli 100kN.

KUVA 19. Purkaimen toinen sylinteri ja ruuvin laatikko kansi suljettuna

Purkaimia käytetään lannanpoistokoneen hydraulikoneikolla, joka on purettu vanhasta navetasta ja se on ollut viimeksi käytössä 80-90-luvun vaihteessa (kuva 20). Koska tässä käyttötarkoituksessa toiminta on täysin sama, koneikko oli suoraan sopiva, eli ainoastaan öljynvaihto, pintapuolinen puhdistus ja koekäyttö oli tarpeen. Pumppu ei vaikuta lainkaan kuluneelta, eikä sen akselin tiivistekään vuoda.

Suunnanvaihtoventtiili vaihtaa sylinterien liikkeen suunnan mekaanisesti paineen noustessa, joten solenoidiventtiilien ohjauksesta ei tarvitse murehtia. Venttiilissä on säädöt kummallekin suunnalle säädöt paineelle, jossa suunta vaihdetaan ja ylipaineventtiili. Säädin kaikki venttiilit mahdollisimman pienelle paineelle ja nostin kaikkia paineita tarpeen mukaan, kunnes purkaimet toimivat normaalisti. Koneikon mukana tuli tarina, jonka mukaan joillakin ihmisillä on ollut paljon ongelmia lannanpoistokoneiden halkeilevien hydrauliletkujen kanssa, mutta tämän koneikon käyttöaikana niitä ei tarvinnut vaihtaa koskaan. Ehkäpä niissä paineet ovat olleet säädettynä liian suuriksi, joten koetin pitää ne mahdollisimman alhaisina. Suunnan vaihtuessa letkuissa näyttää tapahtuvan melko voimakas paineisku, joka saa ne hypähtämään. Liian kovalla

paineella paineisku ehkä saattaa rikkoa letkun.

KUVA 20. Purkaimia käyttävä hydraulikoneikko

Varastoruuvin putkeen on asennettu Danfoss AVTA -venttiili (kuva 21). Jos ruuvin putkessa oleva hake syttyy palamaan, se kuumentaa venttiilin kapillaariputkea, jolloin venttiili laskee vesijohtoverkosta vettä ruuvin putkeen, jotta palo ei pääsisi etenemään hakevarastoon ja tuhoamaan koko rakennusta.

KUVA 21. Danfoss AVTA -venttiili

Kuvassa 22 näkyvä pudotussuppilo toimii pienenä välivarastona hakkeelle, jolloin varastoruuville tulee vähemmän käynnistyksiä. Siihen on myös helppoa asentaa moottorit ja tällaisen kulman tekeminen ruuveille on helpompaa sen avulla. Lisäksi se estää jonkun verran takapalon etenemistä varastoon, mutta sitä ajatellen pudotusmatka on liian pieni ja lisäksi tarvitaan muita keinoja sen estämiseen, kuten esimerkiksi samanlainen vedellä toimivat automaattinen sammutin, kuin varastoruuvin putkessa on.

KUVA 22. Pudotussuppilo. Oikealla on varastoruuvia käyttävä vaihdemoottori ja vasemmalla syöttöruuvin vaihdemoottori. Kannessa on optinen etäisyysanturi.

Syöttöruuvia pyöritetään Bauer Danfoss -vaihdemoottorilla (kuva 23), joka on melkein samanlainen, kuin vanhassa lämpökeskuksessa, mutta vivuston tilalla on Taper-Lock ketjukäyttö. Syötön nopeutta säädetään muuttamalla moottorin nopeutta taajuusmuuttajalla, joten mekaaniselle säädölle ei ole tarvetta. Välityssuhde on hieman väärä, joten sitä pitää myöhemmin muuttaa, jotta moottori pääsisi pyörimään suuremmalla kierrosluvulla. Moottorin rasittumista sai pienennettyä huomattavasti vaihtamalla syöttöruuvin kierre loivemmaksi pudotussuppilon ja ruuvin putken liitoskohdassa. Putki ei ole silloin niin täynnä haketta ja ruuvi pyörii herkemmin, eikä se ole mennyt kierteen loiventamisen jälkeen enää omia aikojaan jumiin. Koska loivempi kierre syöttää haketta hitaammin putkeen, sillä sai myös moottorin kierroslukua nostettua hieman suuremmaksi. Tälläkin välityssuhteella moottori jaksaisi katkaista jumiin menneen syöttöruuvin 30mm akselin, joten siltä yritetään välttyä rajoittamalla moottorin virtaa taajuusmuuttajan parametreissa, koska kummallakaan hammaspyörällä ei käytetä ylikuormituksessa katkeavaa sokkapulttia.

KUVA 23. Syöttöruuvin ketjukäyttö

Pudotussuppilosta hake siirretään ruuvilla polttimeen (kuva 24). Poltin on rakenteeltaan melko samanlainen, kuin vanhassa lämpökeskuksessa. Polttimen ympärille pitää sulloa tiukasti lasivillaa, jotta sen ympäriltä ei pääse vuotamaan ilmaa kattilaan. Ilmavuoto häiritsee jäännöshapen mittausta ja muutenkin se lisää lämmön hukkaa ilman virratessa turhaan kattilan läpi.

KUVA 24. Poltin

Uudet lämmityskattilat ovat melko kalliita, joten tähän lämpökeskukseen hyödynnettiin varastossa lojunut pieni Jämä lämmityskattila. Tyyppikilven perusteella voisi kattilan päätellä olevan vuodelta 1976. Vesitilan sisältä löytyi jonkun verran pistesyöpymiä, mutta se ei silti vuoda. Jos se alkaa vuotaa, vuodon voi paikata hitsaamalla ja varastossa on toinen samanlainen kattila varalla. Uudessa kattilassa hyötysuhde olisi parempi, mutta toisaalta vanhat öljykattilat ovat lähes ilmaisia. Pientä taloa lämmitettäessä saattaa kulua pitkä aika, ennen kuin parempi kattila maksaa itsensä takaisin.

Polttimen voisi asentaa kattilan etupuolella olevaan luukkuun, mutta silloin tuhkan poistaminen ja tulen sytyttäminen olisi hyvin vaikeaa. Siksi kattilan kylkeen polttoleikattiin reikä ja siihen hitsattiin pätkä putkea, jotta vesi pysyisi kattilan vesitilassa (kuva 25). Samalla oli helppoa vaihtaa kattilan eristys, joka oli vuosikymmenien aikana mennyt huonoon kuntoon. Huono eristys tuhlaa energiaa, mutta myös lämmittää pannuhuonetta turhaan rasittaen sinne asennettuja laitteita. Polttimen aukko kannattaa tehdä niin suureksi, että poltin helposti mahtuu siihen, koska sen voi tiivistää myöhemmin.

KUVA 25. Vanhan lämmityskattilan kylkeen tehty polttimen reikä

Polttimen ilmaa puhalletaan käytettynä ostetulla Hitachi sivukanavapuhaltimella (kuva 26), jossa on 3-vaihemoottori ja sen pyörimisnopeutta on helppoa säätää taajuusmuuttajalla. Puhallin on hinnaltaan keskipakopuhallinta kalliimpi, mutta sen uudelleen laakerointi on helppoa, joten se on pitkäikäinen. Se pystyy myös tekemään suuremman paineen, kuin tavallinen keskipakopuhallin. Tarvittaessa sillä voi myös tehdä alipainetta kuristamalla ilmanottoa, mikäli sille tarvetta ilmenee.

KUVA 26. Polttimen ilmapuhallin

5 OHJAUKSEN JA KÄYTÖN RAKENTAMINEN

Polttimen ohjauksen tekemisessä tarvitaan antureita lämpötilojen ja polttoaineen pinnankorkeuksien mittaamiseen. Niiden antamien tietojen perusteella ohjelma lähettää sanomia taajuusmuuttajiin, jotka käyttävät moottoreita jotka huolehtivat hakkeen siirrosta varastosta polttimelle ja hakkeen polttamisesta. Kaikki laitteet ja ominaisuudet on lisätty yksi kerrallaan. Ensin moottoreita käytettiin käsin taajuusmuuttajien näppäimistöiltä, sitten otettiin kenttäväyläohjaus käyttöön ja seuraavaksi anturit. Näin ohjauskeskusta tekemällä kattilalla pystyi lämmittämään taloa ja samalla tehtyjen havaintojen avulla ohjelmoinnissa oli vähemmän tarvetta erilaisille kokeiluille. Kun johtoja on kytketty ohjauskeskukseen yksi kerrallaan ilman suunnitelmaa, ne ovat sekaisin ja tarvitsevat myöhemmin uudelleen kytkemisen.

5.1 Moottorit ja niiden käytöt

Pannuhuoneessa on neljä moottoria: Varastoruuvin ja syöttöruuvin vaihdemoottorit,

puhaltimen moottori ja hydraulikoneikon moottori. Niitä kaikkia käytetään taajuusmuuttajilla. Hydraulikoneikkoa ja varastoruuvin moottoria voisi käyttää kontaktoreilla ja suojata moottorit suojakytkimillä, mutta taajuusmuuttajilla saavutetaan monia etuja:

- Parempi moottoreiden suojaus ylikuumenemiselta.
- Vähemmän ohjauskaapelointeja, voidaan ohjata kenttäväylällä.
- Monipuolinen käyttöjen valvonta.
- Parempi toimintavarmuus. Ei kuluvia releiden ja kontaktoreiden kärkiä, eivätkä kontaktoreiden kelat rasita logiikan lähtöjä induktiivisilla kuormilla.
- Vääntöä voidaan rajoittaa, jolloin ketjut ja vaihteistot eivät rasitu ja ne voidaan mitoittaa pienemmiksi. Ei tarvetta sokkapulteille, jotka katkeavat ruuvin mennessä jumiin.
- Rauhalliset käynnistykset säästävät moottoreita, vaihteistoja, hydrauliikkapumppua ja akseleiden kytkimiä.
- Pyörimissuunnan vaihto on helppoa, jolloin jumiin mennyttä ruuvia voidaan peruuttaa ilman kontaktoreiden lisäämistä.

Kuten monissa muissakin taajuusmuuttajissa, Vacon 20 -taajuusmuuttajissa on Modbus RTU -kenttäväyläliitäntä vakiona, joten päätin käyttää niitä tässä työssä. Moottoreiden kaapelit ovat suojattuja, jotta niistä ei säteilisi häiriöitä analogisiin tuloihin ja muihin lähistöllä oleviin laitteisiin.

5.2 Anturit

Lämpötilojen, pinnankorkeuksen ja jäännöshapen määrää tarkkaillaan erilaisilla antureilla, joiden antaman tiedon perusteella moottoreita ohjataan. Usein hakkeen määrää tarkkaillaan kapasitiivisilla antureilla, mutta käytin tässä optisia etäisyysantureita. Lämpötilojen mittaukseen on käytetty termoparilankoja ja jäännöshapen mittaukseen happianturia.

5.2.1 Pinnankorkeuden mittaus

Tankopurkaimia ja varastoruuvia käytettäessä tarvitaan anturit, jotka valvovat hakkeen määrää. Usein käytetään kapasitiivisia antureita tai valoveräjiä, mutta löysin käytetty-

nä riittävän suuren määrän optisia Sick WT-18 -etäisyysantureita. Ne eivät tarvitse erillistä lähetintä valolle eivätkä peiliä, josta valo heijastuisi takaisin. Kun hakkeen pinta vaikkapa pudotussuppilossa tulee riittävän lähelle anturia, se muuttaa lähdön tilaa. Etäisyyttä voidaan säätää anturissa olevalla potentiometrillä. Mitattavan pinnan väri vaikuttaa hieman etäisyyteen.

Tankopurkaimia on varastossa kaksi kappaletta ja ne pudottavat haketta peltilaatikkoon, jonka pohjalla on varastoruuvi. Asensin neljä kappaletta antureita laatikon kanteen ja jokainen niistä on kytketty erikseen logiikan tuloon. Antureiden rinnankytkentä olisi ollut mahdollista erottamalla ne diodeilla, jolloin yhden anturin lähdöstä ei menisi sähköä toisen anturin lähtöön arvaamattomin seurauksin, mutta logiikassa oli riittävästi tuloja, eikä sille ollut tarvetta. Anturit saattavat toimia rinnankytkettyinä ilman diodejakin, mutta en sitä asiaa tutkinut, eikä näihin antureihin ole ohjekirjaa, mistä asian voisi tarkistaa. Pudotussuppilon kannessa on viides optinen anturi, joka valvoo siellä olevan hakkeen määrää ja varastoruuvia käytetään sen tiedon perusteella.

5.2.2 Ylikuumenemissuoja

Kattilassa oli paikka vanhalle öljypolttimen termostaatille, jossa on ylikuumenemissuoja. Ylikuumenemissuoja ja termostaatti ovat sarjaan kytkettynä, eikä niitä käytetä polttimen varsinaiseen ohjaukseen, vaan ainoastaan ylikuumenemissuojana siltä varalta, että lämpötilan mittaukseen tulee jokin vika ja kattila on vaarassa kiehua.

5.2.3 Lämpötilojen mittaus

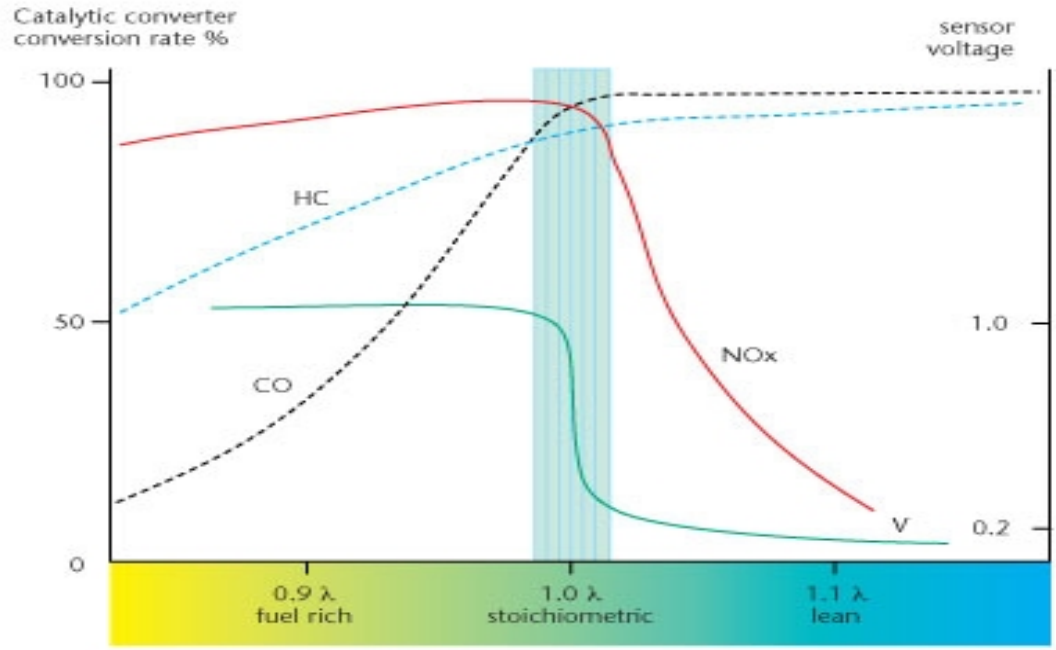
Tässä lämpökeskuksessa mitataan kattilaveden ja savukaasun lämpötilaa. Lisäksi tarkoituksena oli mitata polttimen kuoren lämpötilaa, mutta se ei ollut tarpeen ja sen mittaus jäi pois käytöstä. Sitä voi hyödyntää myöhemmin polttimen ruuvin putken lämpötilan valvontaan, jolloin logiikka pystyy havaitsemaan tulen karkaamisen syöttöruuviin.

Lämpöantureina käytetään K-tyypin termopariantureita, koska ne pystyvät mittaamaan korkeitakin lämpötiloja. Lähettiminä toimivat Analog Devices AD595 -termoparivahvistimet, jotka vahvistavat termoparin antaman pienen jännitteen riittävän suureksi,

jotta logiikka pystyy mittaamaan sen analogisella tulollaan. AD595 antaa ulos 10mV/˚C, joten esimerkiksi 200˚C savukaasua mitatessa se antaa 2V jännitteen. Veden lämpötila on paljon alhaisempi, joten vaikkapa 60˚C vesi saa vahvistimen antamaan 0,6V jännitteen, joka on melko alhainen. Siksi jännitettä kattilaveden mittauksessa vahvistetaan edelleen operaatiovahvistimella, jolloin se skaalautuu paremmin logiikan 0-10V analogiatuloon. Tarkoitusta varten tein piirikortin, jossa on kolme kappaletta AD595 vahvistimia ja operaatiovahvistin. Se vahvistaa ja puskuroi myös happianturin antaman jännitteen.

5.2.4 Jäännöshapen mittaus

Jotta haketta ja ilmaa saisi syötettyä sopivina määrinä polttimeen, pitää savukaasuista mitata jäännöshapen määrä ja säätää niiden syöttöä sen mukaan. Päätin yrittää sen mittaamista tavallisen auton happianturin avulla, jota sanotaan myös lambda-anturiksi. Zirkonium tyyppinen anturi on ollut käytössä jo pitkään ja niitä on helposti saatavilla edulliseen hintaan. Se on helppo kytkeä, koska siinä on vain kaksi johtoa mitatulle tiedolle ja kaksi hehkulle. On myös antureita ilman hehkua, mutta sellaisen sijoittaminen kattilaan olisi melko hankalaa, koska savukaasujen pitäisi lämmittää anturi riittävään toimintalämpötilaan ja liian lähellä poltinta sijaitseva anturi saattaa ehkä ylikuumentua. Halvalla hinnalla ja yksinkertaisuudella on kääntöpuolensa: Se on erittäin kapeakaistainen ja toimii lähinnä alueella, jossa polttomoottori polttaa bensiiniä stoikiometrisesti. On olemassa laajakaistaisempia anturityyppejä, mutta ne ovat kalliimpia ja poltin saattaa toimia hyvin kapeakaistaisellakin anturilla. Kuvassa 27 on piirretty vihreällä viivalla anturin antama jännite seoksen muuttuessa rikkaammaksi. Jännite muuttuu jyrkästi siinä kohdassa, missä polttomoottori polttaa polttoainetta stoikiometrisesti.



KUVA 27. Erilaisia kuvaajia happianturista polttomoottorikäytössä (NGK)

Anturissa on keraaminen elementti, jonka toisella puolella on mitattava kaasu ja toisella puolella puhdasta ilmaa. Anturi vertailee hapen määrää molemmilla puolilla elementtiä ja jos hapen määrä ei ole sama, se antaa ulos jännitteen. Jännite kasvaa eron kasvaessa. Laihalla seoksella jännite on suunnilleen 0,1V ja rikkaalla 0,9V. [2]

Anturi oli helpointa sijoittaa kattilan nuohousluukkuun, jossa polttimen kuumuus ei pääse kuumentamaan sitä liikaa. Arvelisin anturin antaman virran olevan hyvin pieni ja niin on myös jännite, joten sitä vahvistetaan suuremmaksi operaatiovahvistimella, joka samalla puskuroi signaalin. Näin logiikan analogiatulo ei kuormita suoraan anturia ja analoginen signaali ei ole niin herkkä kaapeleiden sieppaamille häiriöille.

5.3 Logiikka

Logiikaksi valitsin Wago 750-8202:n tarvittavine I/O moduuleineen. Digitaalisia 24V tuloja on 8, transistoriohjattuja 24V lähtöjä 8 ja analogisia 0-10V tuloja on 4. Koska moottoreita ohjataan kenttäväylällä, analogisia lähtöjä ei tarvita. Logiikassa on sarjaportti, jota voi käyttää RS-232 porttina tai RS-485 porttina, mutta ei molempina samaan aikaan ja sitä tarvitaan RS-485 tilassa kenttäväylää varten. Lämpökanaalin lisäk-

si maahan kaivettiin rakennusten välille sadevesiviemäriputket siltä varalta, jotta sinne voidaan vetää tarvittaessa lisää kaapeleita. Logiikan ethernet liitäntää varten vedin putkeen CAT-6 kaapelin, jolla se voidaan liittää talossa olevaan reitittimeen ohjelmointia varten. Silloin ohjelmoinnin voi tehdä mukavasti sisällä lämpimässä, mutta pitää olla huolellinen, koska koneiden toimintaa ei voi sieltä käsin valvoa.

5.4 Kenttäväylä

Kenttäväyläksi valitsin Modbus RTU:n koska sitä ei tarvitse ostaa erikseen taajuusmuuttajiin ja sen ominaisuudet riittävät mainiosti tähän tarkoitukseen. Väylä on kaapeloitu suojatulla JAMAK kaapelilla, jolloin se ei ole niin herkkä ulkoisille häiriöille. Toimivuuden kannalta väylän häiriöt aiheuttavat ongelmia, koska moottoreiden käynnistäminen ja pysäyttäminen vaikeutuu tai estyy kokonaan. Suurta vaaraa ei kuitenkaan aiheudu, koska taajuusmuuttajat menevät vikatilaan ja pysäyttävät moottorit, jos ne eivät vastaanota sanomia tietyn ajan kuluessa. Tämän ajan voi määritellä taajuusmuuttajan parametreissa.

6 OHJELMOINTI

Logiikan ohjelmoinnista Codesysillä minulla ei ollut lainkaan aikaisempaa kokemusta, mutta ohjelmaan vähän kerrallaan toimintoja lisäämällä sen pystyi opettelemaan melko helposti. Ohjelma syntyi samaa tahtia ohjauskeskuksen kaapeloinnin kanssa, eikä siitäkään ollut tarkkaa etukäteissuunnitelmaa, joten sitäkin voisi vielä selkeyden vuoksi siistiä myöhemmin.

6.1 Toimintaselostus

Tankopurkaimet, varastoruuvi ja poltin voivat toimia toisistaan riippumatta, joten niistä voi tehdä omat toimintaselostukset. Vaikka tankopurkaimet eivät jostakin syystä toimisi, varastoruuvin laatikossa on jonkun verran haketta ja se voi siirtää sitä pudotussuppiloon. Varastoruuvin lakatessa toimimasta poltin toimii sen aikaa, kuin pudotussuppilossa on haketta.

Tankopurkaimet:

Tankopurkaimet työntävät haketta varastoruuvin laatikkoon, jonka kannessa on neljä optista etäisyysanturia. Laatikko täyttyy epätasaisesti purkainten toimiessa ja siksi useampi anturi on tarpeen. Ruuvin ei tarvitse olla koko pituudeltaan hakkeen peittämä, mutta laatikko ei saa täyttyä liikaa yhdestäkään kohtaa, tai hake alkaa pursuamaan yli kannen alta.

Hydraulipumppua käytetään silloin, kun kaikki anturit ilmoittavat laatikon olevan tyhjillään ja pysäytetään, kun jokin antureista ilmoittaa havaitsevansa haketta laatikossa. Varastoruuvin pyöriessä pumppua ei käynnistetä, jotta kookkaat moottorit eivät turhaan rasittaisi sulakkeita. Jos pumppu käy liian pitkään, eikä mikään anturi havaitse laatikon täyttymistä, se pysäytetään sen varalta, että anturit eivät jostakin syystä toimi tai hake on loppu. Pumppua ei käynnistetä uudelleen, ennen kuin vika on kuitattu.

Varastoruuvi:

Varastoruuvia ohjataan pudotussuppilon kannessa olevalla optisella etäisyysanturilla. Jos anturi ilmoittaa hakkeen olevan vähissä, ruuvi käynnistetään ja sitä käytetään vähän aikaa anturin ilmoitettua havaitsevansa hakkeen pinnan olevan riittävällä tasolla. Näin haketta siirretään kerralla enemmän ja ruuvia tarvitsee käynnistellä harvemmin. Jos ruuvi käy liian pitkään, se pysäytetään anturin rikkoutumisen tai hakkeen loppumisen varalta, kunnes vika kuitataan.

Poltin:

Tätä työtä tehdessä polttimen sytyttämiseen ei ole mitään ohjelmaa, joten polttimeen sytytetään tulet käsin, kuitataan mahdolliset viat ja laitetaan poltin käyntitilaan. Jos tulet ovat syttyneet riittävän hyvin, poltin alkaa toimia normaalisti.

Jos kattilaveden lämpötila on alhaisempi, kuin asetettu alaraja, poltin on käyntitilassa, eikä vikoja ole kuittaamatta, käynnistetään syöttöruuvin moottori ja ilmapuhallin. Kun veden lämpötila nousee asetettuun ylärajaan, pysäytetään syöttöruuvi ja ilmapuhallinta käytetään hetken aikaa, jotta liika hake palaa pois polttimesta savuttamisen välttämiseksi.

Syöttöruuvin nopeutta säädetään PID-säätimellä. Säädin tarkkailee happianturin antamaa tietoa ja nostaa ruuvin nopeutta hapen määrän kasvaessa savukaasussa ja vastaavasti hidastaa sitä hapen määrän laskiessa. Ilmapuhallinta säädetään savukaasun lämpötilan mukaan PI-säätimellä niin, että savukaasun lämpötilan noustessa puhaltimen nopeutta pienennetään.

Turvallisuuden takia poltin on pysäytettävä erilaisista syistä:

- Kattila ylikuumenee ja lämpösuoja laukeaa

- PID-säätimen integroivassa osassa tapahtuu ylivuoto ja säädin lakkaa toimimasta normaalisti.

- Jokin termopariantureista ilmoittaa, että lämpötila on joko niin suuri tai pieni, ettei se ole mahdollista. Se voi tarkoittaa todennäköisimmin sitä, että anturi on viallinen, eli termoparilanka on katkennut ja se on korjattava ennen käytön jatkamista. On myös mahdollista, että savukaasua mittaavan termoparilangan ympärillä on nokipalo tai tulet polttimessa ovat jostakin syystä liian suuret, jolloin poltin on turvallisinta pysäyttää.

- Polttimen käynnistyttyä savukaasun lämpötila ei nouse tietyn ajan kuluessa asetetun arvon yläpuolelle tai laskee yhtä pitkäksi ajaksi asetetun arvon alapuolelle. Silloin tulet ovat luultavasti sammuneet ja poltin on pysäytettävä, jotta ruuvi ei jatkaisi hakkeen syöttämistä kattilaan tuhoisin seurauksin.

- Jokin taajuusmuuttajista menee vikatilaan.

6.2 I/O taulu ja moottoreiden ohjaus

TAULUKKO 1 Digitaaliset tulot

Osoite	
%IX4.0	laatikko1
%IX4.1	laatikko2
%IX4.2	laatikko3
%IX4.3	laatikko4
%IX4.4	-
%IX4.5	-
%IX4.6	termostaatti
%IX4.7	suppilo

Logiikkaan on asennettu kahdeksan 24V digitaalituloa ja niille on määritelty kuvaavammat nimet (taulukko 1). ”Laatikko” tulot ovat varastoruuvin laatikon optiset etäisyysanturit, ”termostaatti” mekaaninen ylikuumenemissuoja ja ”suppilo” pudotussuppilon optinen etäisyysanturi.

TAULUKKO 2. Analogiset tulot

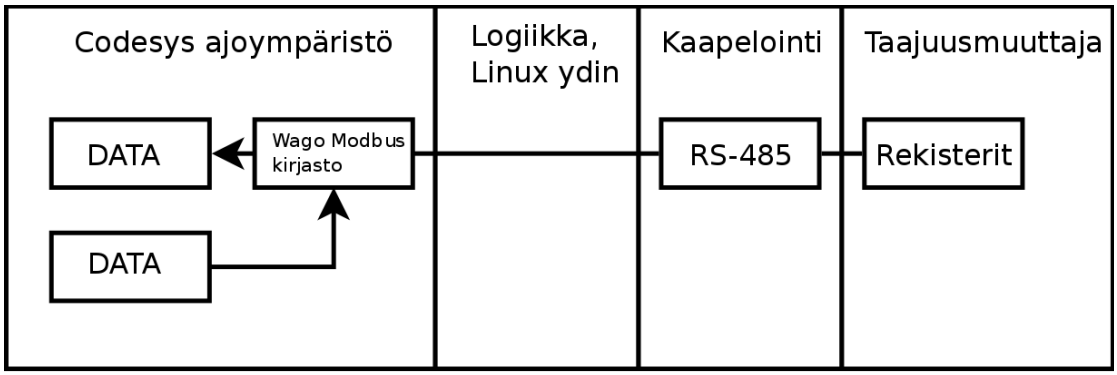
Osoite	
%IW0	savu
%IW1	-
%IW2	happi
%IW3	vesi

Taulukossa 2 näkyy analogiset tulot, joita on neljä kappaletta ja ne toimivat 0-10V jännitteellä. ”Savu” on savukaasun lämpötila, ”happi” jäännöshapen määrä ja ”vesi” kattilaveden lämpötila.

Digitaaliset lähdöt eivät ole käytössä. Hydraulipumpun kenttäväyläliitäntä ei jostakin syystä toimi tätä kirjoitettaessa ja sitä ohjataan tilapäisesti yhdellä digitaalilähdöllä %QX0.1, kunnes taajuusmuuttaja vaihdetaan.

6.3 Kenttäväylä

Moottoreita ohjataan kenttäväylällä, mutta taajuusmuuttajien Modbus-rekisterit eivät näy suoraan logiikan lähtöinä tai tuloina, vaan niitä varten pitää tehdä pieni pätkä ohjelmaa, jonka avulla rekistereihin voidaan kirjoittaa ja niitä voidaan lukea helposti ilman sekaannuksen vaaraa. Sen jälkeen voidaan esimerkiksi ilmapuhaltimen moottori käynnistää tai pysäyttää ohjaamalla kelaa ”puhallin” tikapuukaaviossa tai kirjoittamalla taajuusohje muuttujaan. Olen yrittänyt havainnollistaa tiedon kulkua kuvassa 28.



KUVA 28. Kaavio tiedon kulusta ohjelmasta taajuusmuuttajan rekisteriin

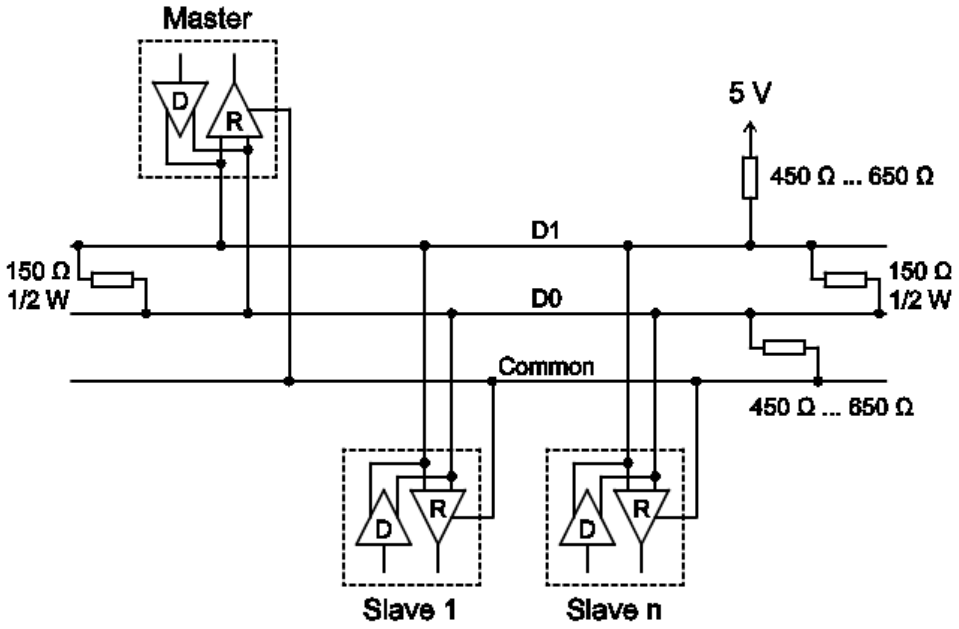
Ohjelmassa käytetään libmodbus kirjastoa ja siinä olevalle funktioblokille kerrotaan väylän parametrit, haluttu toiminto, rekisterin numero ja tarvittaessa arvot, jonka jälkeen se huolehtii liikenteestä taajuusmuuttajalle asti, joten itse Modbus-protokollaan ei siis kannata tässä paneutua kovinkaan tarkasti.

6.3.1 Kaapelointi

Modbus on protokolla tiedon siirtämiseen, joten kaapelointi ei ole Modbus-kaapelointi, vaan se tehdään RS-485-standardin mukaan. RS-485-väylä huolehtii tiedon siirtämisestä, eli tässä tapauksessa se siirtää Modbus-dataa laitteiden välillä. Kuvassa 29 nähdään väylän kytkentä.

Logiikan RS-485-liitännästä on vedetty JAMAK kaapeli taajuusmuuttajille ja kaikki laitteet ovat ketjutettu väylään. Väylä pitää päättää molemmista päistä vastuksilla, sik-

si logiikan D-liittimeen on juotettu päätevastus ja taajuusmuuttajiin se on sisäänrakennettuna, joten se piti ottaa käyttöön väylän viimeisessä taajuusmuuttajassa. Ilman päätevastuksia alkaa väylässä kulkemaan heijastuksia ja ne voivat aiheuttaa vakaviakin häiriöitä. Kuvassa 29 näkyviä ylös- ja alasvetovastuksia ei ollut tarpeen käyttää, eikä myöskään kolmatta ”Common” johdinta, vaan väylä näyttää tässä tapauksessa toimivan hyvin ilman niitäkin. Tällaisissa pienissä asennuksissa ne voidaan helposti lisätä myöhemmin, jos tarvetta ilmenee. Vastusten tarkoituksena on estää väylän jännitteen kelluminen 0-5V välillä, kun mikään laite ei pidä väylää 1- tai 0-tilassa.



KUVA 29. Kaavio RS-485 väylästä (Wago)

6.3.2 Taajuusmuuttajan rekisterit

Taajuusmuuttajassa on rekistereitä (taulukko 3), joista voidaan lukea tietoa tai niihin kirjoittamalla voidaan moottori esimerkiksi käynnistää. Vacon 20:n rekistereistä 32101-32111 voidaan lukea paljon erilaisia tietoja ja rekisterit 32104-32111 ovat muutettavissa tarpeen mukaan taajuusmuuttajan parametreja muuttamalla. Tässä työssä käytetään oletusasetuksia, mutta myöhemmin esimerkiksi aktiivinen vikakoodi on kiinnostava tietää. Silloin esimerkiksi ruuvin mentyä jumiin logiikka voi kuitata vian taajuusmuuttajassa, odottaa hetken moottorin jäähdyttämiseksi, pyörittää ruuvia takaperin tukoksen selvittämiseksi ja sen jälkeen toimintaa voi yrittää jatkaa.

TAULUKKO 3. Taajuusmuuttajan lähtöprosessidata

ID	Modbus-rekisteri	Nimi	Skaala	Tyyppi
2101	32101, 42101	FB Status Word		Binäärikoodi
2102	32102, 42102	FB General Status Word		Binäärikoodi
2103	32103, 42103	Varattu	0,01	%
2104	32104, 42104	Taajuusviite	-	-
2105	32105, 42105	Lähtötaajuus	0,01	Hz
2106	32106, 42106	Moottorin nopeus	1	Rpm
2107	32107, 42107	Moottorin jännite	0,1	V
2108	32108, 42108	Moottorin vääntömomentti	0,1	% (nimellisarvosta)
2109	32109, 42109	Moottorin virta	0,01	A
2110	32110, 42110	Moottorin teho	0,1	% (nimellisarvosta
2111	32111, 42111	DC-linkin jännite	1	V

Rekisteristä 32101 voidaan lukea tilasana (taulukko 4), joka muuttuu ymmärrettäväksi muutettaessa se binäärimuotoon. Jos kuvitellaan, että moottori käy normaalisti myötäpäivään, tilasana olisi seuraavanlainen:

B0=1 Laite on valmis
B1=1 Käy
B2=0 Myötäpäivään
B3=0 Ei vikaa
B4=0 Ei hälytystä
B5=1 Nopeusohje saavutettu
B6=0 Laite ei käy nollanopeudella

B7-B15=0

Silloin tilasana on ”1100 0100 0000 0000”, eli heksadesimaalilukuna ”C400”. Heksadesimaaliluvusta voidaan erotella bitit BOOL muuttujiin ja käyttää esimerkiksi bittiä 3 pysäyttämään polttimen taajuusmuuttajan mentyä vikatilaan.

TAULUKKO 4. Tilasana

Bitti	Arvo = 0	Arvo = 1
B0, RDY	Laite ei ole valmis	Laite on valmis
B1, KÄY	Seis	Käy
B2, DIR (suunta)	Myötäpäivään	Vastapäivään
B3, FLT (vika)	Ei vikaa	Vika aktiivinen
B4, W (varoitus)	Ei hälytystä	Hälytys aktiivinen
B5, AREF	Ramppaa	Nopeusohje saavutettu
B6, Z	-	Laite käy nollanopeudella
B7-15	-	-

Ohjaussana (taulukko 5) toimii samalla tavalla, kuin tilasana ja sillä moottori voidaan esimerkiksi pysäyttää, käynnistää ja valita pyörimissuunta. Moottorin käynnistämiseksi myötäpäivään on rekisteriin kirjoitettava ”1000 0000 0000 0000” eli ”0001” heksadesimaalimuodossa, ”0003” saa moottorin vaihtamaan pyörimissuunnan ja ”0000” pysäyttää moottorin.

TAULUKKO 5. Ohjaussana

Bitti	Arvo = 0	Arvo = 1
B0, KÄY	Seis	Käy
B1, DIR (suunta)	Myötäpäivään	Vastapäivään
B2, RST	Tämän bitin nouseva reuna kuittaa aktiivisen vian	
B5, pikaramppiaika	Normaalihidastus	Pikahidastus

Moottorin pyörimisnopeutta voidaan säätää kirjoittamalla rekisteriä 32003 (taulukko 6). Taajuusmuuttajan ohjekirjassa on virhe, ja rekisterit 32003 ja 32103 eivät ole ”varattuja”, vaan rekisteristä 32103 voidaan lukea nopeusohje ja se voidaan kirjoittaa rekisteriin 32003. Koska skaalaksi on ilmoitettu 0,01%, voi nopeusohje olla välillä 0-10000.

TAULUKKO 6. Tuloprosessidata

ID	Modbus-rekisteri	Nimi	Skaala	Tyyppi
2001	32001, 42001	FB Control Word	-	
2002	32002, 42002	FB General Control Word	-	
2003	32003, 42003	Varattu	0,01	%
2004	32004, 42004	Ohjelmoitavissa P10.9:ssä		
2005	32005, 42005	Ohjelmoitavissa P10.9:ssä		
2006	32006, 42006	Ohjelmoitavissa P10.9:ssä		
2007	32007, 42007	Ohjelmoitavissa P10.9:ssä		
2008	32008, 42008	Ohjelmoitavissa P10.9:ssä		
2009	32009, 42009	-	-	-
2010	32010, 42010	-	-	-
2011	32011, 42011	-	-	-

6.3.3 Kenttäväylän ohjelma

Kenttäväylä toimii Modb_l05-kirjaston avulla, joten käytetään Wagon toimittamaa MODBUS_MASTER_RTU-funktioblokkia. Väylän parametrit annetaan funktioblokille, jotka tässä tapauksessa ovat seuraavat:

Portti	0
Nopeus	19200
Pariteetti	Ei
Stopbittejä	2
Databittejä	8
Vuonohjaus	Ei

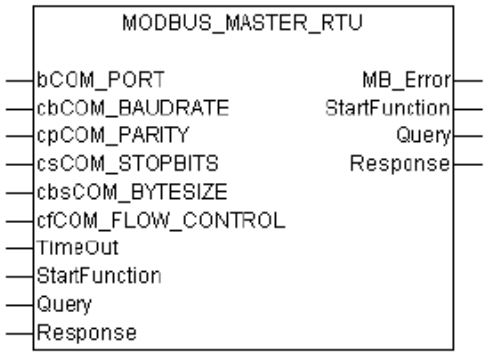
Taajuusmuuttajissa parametrit pitää olla samat ja jokaiselle niistä pitää antaa osoite, jolla ne tunnistetaan väylässä. Varastoruuvin taajuusmuuttajan osoite on 2, syöttöruuvin 3 ja puhaltimen 4. Hydraulipumpulle on varattu osoite 1 ja sen voi ottaa käyttöön, kun taajuusmuuttajan vaihtaa myöhemmin uuteen.

Vacon 20 taajuusmuuttajat ymmärtävät neljä erilaista Modbus-komentoa (taulukko 7). Tässä ohjelmassa riittävät 03, jolla rekistereitä luetaan ja 16:lla voidaan kirjoittaa useampaan peräkkäiseen rekisteriin. Koska MODBUSMASTER_RTU-funktioblokki (kuva 30) huolehtii väylässä siirrettävien sanomien muotoilusta, ei ole tarvetta laskea tarkistussummia, eikä valikoida vastaanotetusta sanomasta haluttua tietoa.

TAULUKKO 7. Modbus komennot

Toiminnon koodi	Toiminnon nimi	Osoite	Lähetä viestejä
03	Lue pitorekisterit	Kaikki tunnistenumerot	Ei
04	Lue tulorekisterit	Kaikki tunnistenumerot	Ei
06	Kirjoita yksittäiset rekisterit	Kaikki tunnistenumerot	Kyllä
16	Kirjoita useita rekistereitä	Kaikki tunnistenumerot	Kyllä

Luettavan ja kirjoitettavan tiedon muuttujat voidaan nähdä taulukosta 8. Taajuusmuuttajan osoite, Modbus-komento ja kirjoitettava tieto kirjoitetaan ensin ”ExtQuery” taulukkoon ja ”StartFunction” käynnistetään funktioblokki vaihtamalla ”StartFunction” muuttuja tilaan TRUE. Funktioblokki lähettää sanoman taajuusmuuttajaan, joka lähettää vastauksen. Vastaus on luettavissa taulukosta ”Response”. Jos yhteyttä taajuusmuuttajaan ei saada muuttujan ”TimeOut” määrittämässä ajassa, voidaan virheilmoitus lukea muuttujasta ”MB_Error”. Kun kysely on valmis, funktioblokki vaihtaa ”StartFunction” muuttujan tilaan FALSE, jolloin tiedot taulukoista voidaan kopioida johonkin muualle, kirjoittaa ”ExtQuery” taulukkoon uudet tiedot ja tehdä uusi kysely samalla tavalla.



KUVA 30. MODBUS_MASTER_RTU funktioblokki (Wago)

TAULUKKO 8. MODBUS_MASTER_RTU funktioblokin tulo- ja lähtömuuttujat

StartFunction	Bool	Set true to start execution of modbus service. The variable will be reset by the function block after execution.
ExtQuery	typModbusExtendedQuery	TYPE typModbusExtendedQuery : STRUCT SlaveAddress : BYTE; FunctionCode : BYTE; Read_StartAddress : UINT; Read_Quantity : UINT; Write_StartAddress : UINT; Write_Quantity : UINT; Write_Data : ARRAY[0..124] OF WORD; END_STRUCT END_TYPE
Response	TypModbusResponse	TYPE TypModbusResponse : STRUCT Error : WORD; SlaveAddress : BYTE; FunctionCode : BYTE; StartAddress : UINT; Quantity : UINT; Data : ARRAY[0..124] OF WORD; END_STRUCT END_TYPE

Tässä työssä jokaiselle taajuusmuuttajalle on tehty oma taulukko kirjoitettavalle ja sieltä luettavalle tiedolle ja ne ovat numeroitu Modbus-osoitteen mukaan. ”dataread2” sisältää varastoruuvin taajuusmuuttajan rekistereistä luetut tiedot ja ”datawrite2” sisältää sinne kirjoitettavat tiedot. Kenttäväylälle on tehty oma ST (structured text) -ohjelma, joka huolehtii siitä, että taulukoissa olevat tiedot ovat ajan tasalla ja rekistereihin kirjoitettavat tiedot tulevat niihin kirjoitetuiksi.

Ensin funktioblokki pitää lisätä ohjelmaan, esimerkiksi sen loppuun:

```
M1(                         (* VAR       M1: MODBUS_MASTER_RTU; *)
    bCOM_PORT:= 16#00,
    cbCOM_BAUDRATE:= 1920,
    cpCOM_PARITY:= 0,
    csCOM_STOPBITS:= 2,
    cbsCOM_BYTESIZE:= 8,
    cfCOM_FLOW_CONTROL:= 4,
    TimeOut:= t#1000ms,
    StartFunction:= StartFunction,
    Query:= Query,
    Response:= Response,
    MB_Error=>MB_Error
);
```

Kun ”datawrite2[0]” sisältää ohjaussanan ja ”datawrite2[2] nopeusohjeen, ne voidaan yksinkertaisimmillaan kirjoittaa taajuusmuuttajaan seuraavasti:

```
IF NOT StartFunction THEN
Query.FunctionCode:=16;
        Query.SlaveAddress:=02;                 (*Modbus osoite 2*)
        Query.StartAddress:=2000;               (*Ensimmäinen kirjoitettava
                                                rekisteri*)
        Query.Data[0]:=datawrite2[0];           (*rekisteri 2001*)
        Query.Data[1]:=0;                       (*rekisteri 2002*)
        Query.Data[2]:=datawrite2[2];           (*rekisteri 2003*)
StartFunction:=TRUE;
END_IF
```

Joskus eri laitteiden tapa laskea rekistereiden järjestys vaihtelee, jolloin funktioblokille saattaa joutua antamaan rekisterin 2000 kirjoitettavaksi, jotta rekisteri 2001 tulee kirjoitetuksi. Taajuusmuuttajan rekisterit voidaan lukea seuraavalla tavalla:

```
0:
IF NOT StartFunction THEN
Query.FunctionCode:=03;
Query.SlaveAddress:=02;
Query.StartAddress:=2100;
Query.Quantity:=11;
StartFunction:=TRUE;
state:=1;
END_IF
1:
IF NOT StartFunction THEN
FOR i:=0 TO Response.Quantity DO
        dataread2[i]:=Response.Data[i];         (*kopioidaan tiedot toiseen
                                                taulukkoon*)
        END_FOR
```

```
state:=0;
END_IF
```

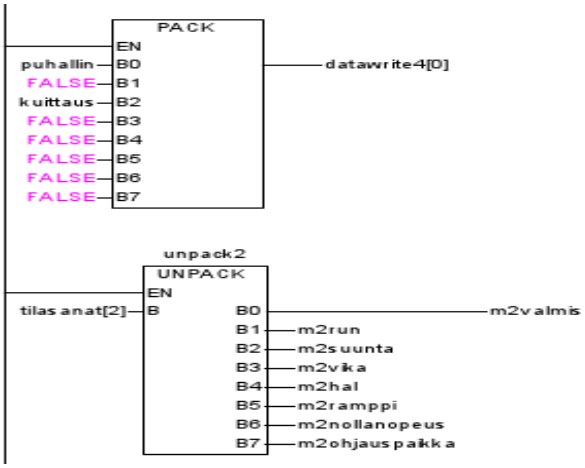
”Response.Quantity” kertoo, kuinka monta riviä ”Response.Data” sisältää. FOR silmukan jokaisella kierroksella kopioidaan yksi rivi ja ”i” kasvaa yhdellä. Kun kaikki rivit ovat kopioitu, ohjelma poistuu silmukasta. Lopullinen ohjelma on liitteessä ja siellä on myös lisätty tarkistus virheiden varalta.

Nyt rekistereitä voidaan lukea ja kirjoittaa ja taajuusmuuttajan tietoja säilyttävästä taulukosta on helppoa lukea vaikkapa moottorin momentti tai välipiirin jännite, mutta ohjaussanan tekeminen ja tilasanan lukeminen on vielä hieman hankalaa. Moottoria voi ohjata vaikkapa seuraavasti:

```
datawrite2:=1        (*moottori käy myötäpäivään*)
datawrite2:=3        (*moottori käy vastapäivään*)
datawrite2:=0        (*seis*)
```

Moottorin käyttö olisi hieman helpompaa ja selkeämpää, jos sitä voisi ohjata BOOL muuttujalla, eikä siinä olisi silloin niin suurta erehdyksen vaaraa. Jos numerot 1 ja 3 sekoittaa keskenään, moottori käynnistyy väärään suuntaan, jolloin esimerkiksi ilmapuhallin alkaa imeä polttimesta tulta ja savua. PACK-funktiossa on tuloina 8 BOOL muuttujaa ja se muodostaa niistä komentosanan tavun (BYTE), joka voidaan kirjoittaa taulukkoon komentosanan riville.

UNPACK toimii päinvastoin ja sillä voidaan tilasana purkaa BOOL muuttujiksi (kuva 31).



KUVA 31. PACK ja UNPACK funktioblokkien käyttö tila- ja ohjaussanojen käsittelyyn

Tilasana on 16-bittisessä WORD-tyypin muuttujassa, mutta UNPACK tarvitsee sen 8-bittisenä BYTE-tyypin muuttujana. Tarvittavat bitit ovat ensimmäiset kahdeksan bittiä, joten ne saadaan erotettua seuraavasti:

```
VAR
    HI AT %MB0: BYTE;
    LO AT %MB1: BYTE;
    SANA AT %MW0: WORD;
END_VAR

SANA:=dataread2[1];
tilasanat[2]:=HI;
```

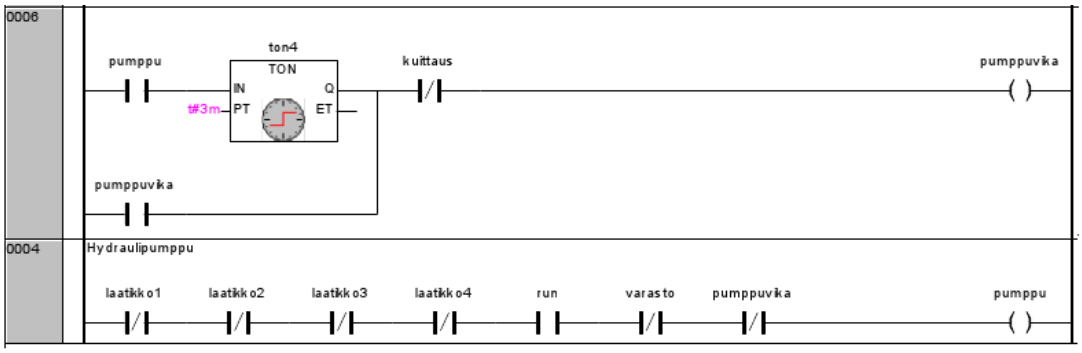
Muistiosoitteeseen 0 on määritetty SANA niminen WORD-tyypin muuttuja ja sen kanssa päällekkäin määritellään muuttujat HI ja LO. SANA käyttää muistin ensimmäiset 16 bittiä, HI käyttää ensimmäiset 8 ja LO seuraavat 8. Lukemalla muuttuja HI saadaan siis luettua ensimmäiset 8 bittiä muuttujasta SANA, joissa sijaitsevat taajuusmuuttajan tilasta kertovat bitit. Muuttujaan LO jäävät bitit kertovat, mikä on taajuusmuuttajassa valittu ohjauspaikka.

Nyt kenttäväylässä olevia taajuusmuuttajia voidaan ohjata samalla tavalla, kuin käyn-

nistettäessä ne digitaalisella I/O:lla ja antamalla nopeusohje analogisella lähdöllä, eikä väylän käyttö tee ohjelmasta monimutkaista.

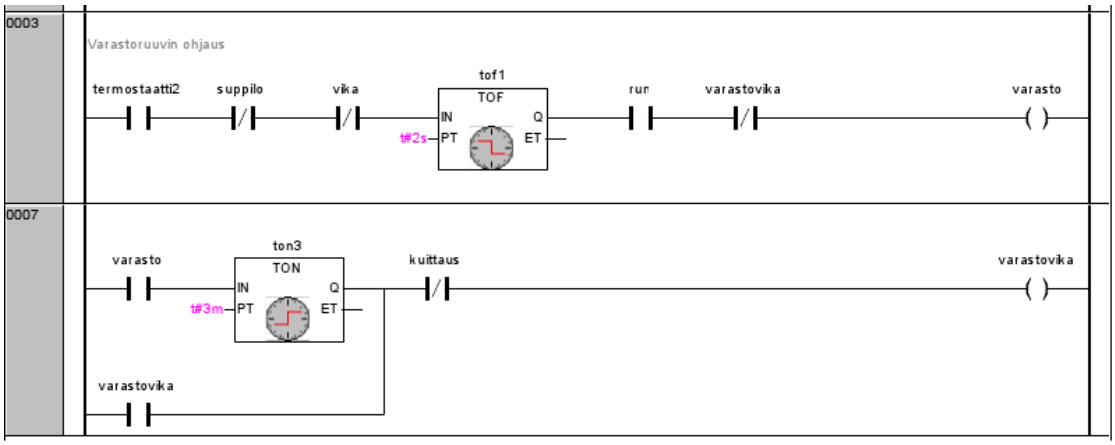
6.4 Moottoreiden ohjaus

Tankopurkainten hydraulipumppu käynnistetään, jos mikään ruuvin laatikon kannen etäisyysantureista ei havaitse haketta, varastoruuvi ei pyöri, eikä hydraulipumppu ole vikatilassa (kuva 32). Vikatila menee päälle, jos pumppu käy kolme minuuttia, eikä etäisyysanturit silti havaitse haketta. Hydraulipumpussa on 2kW moottori ja se ottaa paljon virtaa, joten sitä ei kannata käyttää samaan aikaan varastoruuvin 750W moottorin kanssa.



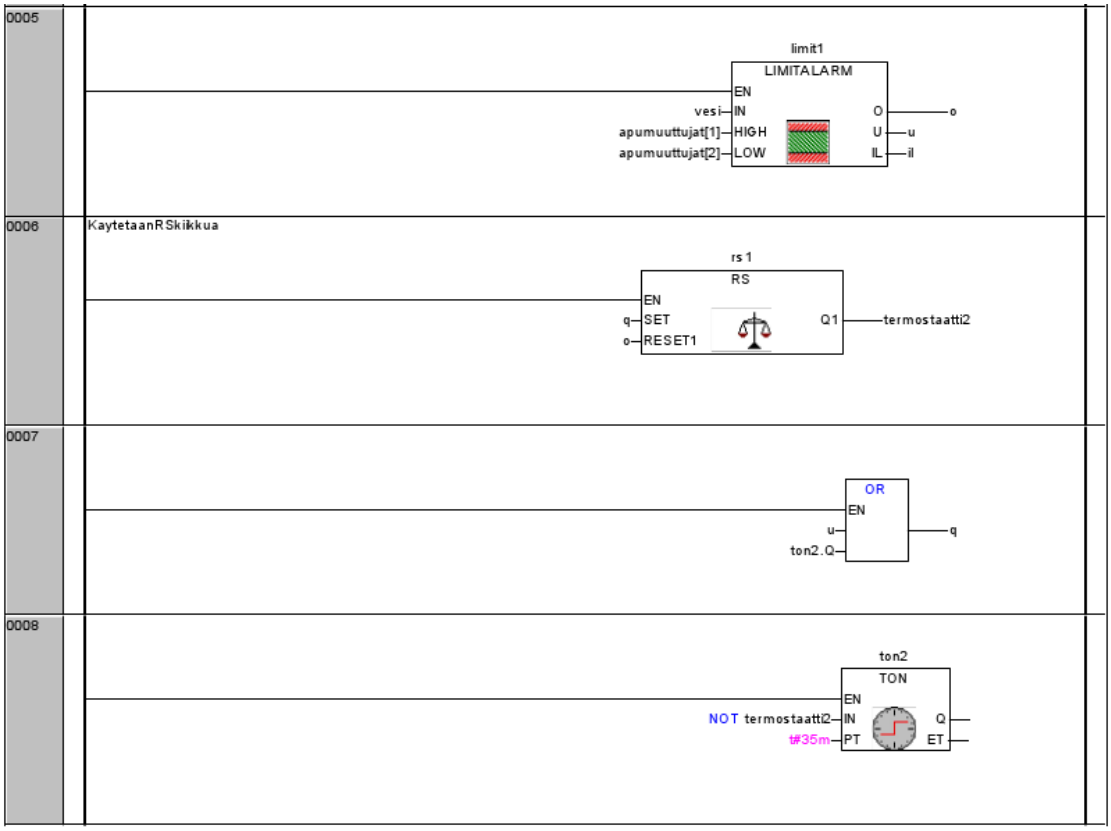
KUVA 32. Hydraulipumpun ohjaus

Varastoruuvi käynnistetään, jos poltin käy, pudotussuppilon kannen etäisyysanturi ilmoittaa hakkeen olevan vähissä, eikä vikoja ei ole päällä (kuva 33). Koska etäisyysanturin hystereesi on melko pieni, ruuvi jatkaa pyörimistä vielä kaksi sekuntia pinnan noustua riittävästi, jotta moottorille tulisi vähemmän käynnistyksiä. Kolmen minuutin pyörimisen jälkeen ruuvi menee vikatilaan hydraulipumpun tavoin.



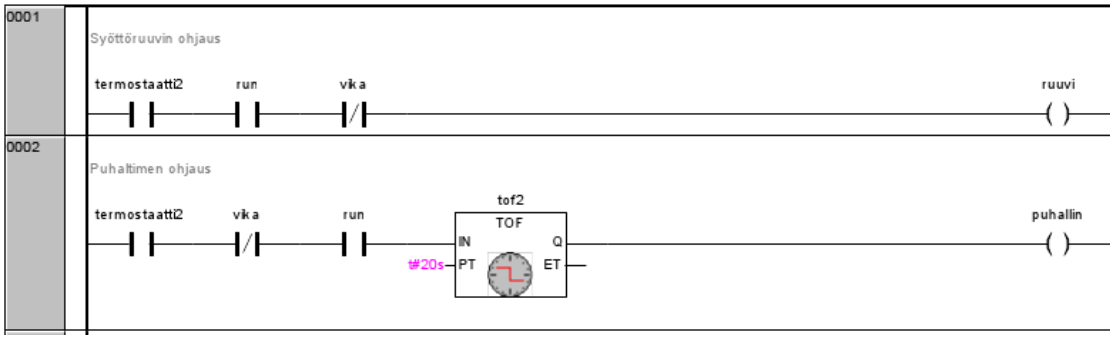
KUVA 33. Varastoruuvin ohjaus

Kattilaveden lämpötila-anturi on ohjelmassa nimeltään ”vesi”. ”apumuuttujat[1]” sisältää lämpötilan, jossa poltin pysäytetään ja ”apumuuttujat[2] lämpötilan, jossa poltin käynnistetään. LIMITALARM-funktioblokin lähtö ”O” menee päälle, veden lämpötilan saavuttaessa tai ylittäessä ylemmän asetusarvon ja ”U” lämpötilan saavuttaessa tai alittaessa alemman asetusarvon. OR-veräjän avulla RS-kiikku liipaistaan päälle joko veden lämpötilan laskiessa tai polttimen oltua käymättä tietyn ajan (kuva 34). Näin polttimessa ei tuli pääse sammumaan, vaikka lämmön kulutus on alhaisempi, eikä veden lämpötila laske riittävän nopeasti saavuttaakseen alemman rajan. Veden lämmettyä riittävästi RS-kiikku siirtyy FALSE tilaan.



KUVA 34. Kattilatermostaatti

Ilmapuhallin ja syöttöruuvi käynnistetään, kun aiemmin selostettu termostaatti käskee polttimen käynnistymään (kuva 35). Puhallinta käytetään hetki syöttöruuvin pysähdyttyä, jotta osa polttimeen jääneestä hakkeesta palaisi pois. Tarkoituksena on vähentää savuttamista pysähtymisen jälkeen. Polttimeen jäänyt hake palaa pois ja jäljelle jää vain hehkuva hiillos, mutta sen sijaan ruuvin päässä oleva hake alkaa kyteä, eikä tästä näytä olevan paljoa apua savun muodostumisen vähentämisen kannalta.



KUVA 35. Ilmapuhaltimen ja syöttöruuvin ohjaus

6.5 Vikatilat

Poltin pitää pysäyttää, jos se on sammunut, taajuusmuuttaja menee vikatilaan, tai antureista tulee sellaista tietoa, ettei se ole mahdollista. Jos esimerkiksi kattilaveden lämpötilaksi mitataan 400°C, vesi tuskin on niin kuumaa, vaan vika on luultavasti lämpötilan mittauksessa.

Liekkiä polttimessa seurataan savukaasun lämpötilasta. Kun poltin käynnistyy, pitää savukaasun lämpötilan nousta riittävästi tietyn ajan kuluessa, jotta käynti voi jatkua. Myös lämpötilan laskeminen asetetun rajan alle käynnin aikana aiheuttaa vikatilan. Jos savukaasun lämpötila nousee liikaa, se voi johtua mittausvirheestä, nokipalosta tai liian suuresta hakkeen syötöstä ja sekin aiheuttaa vikatilan. Samalla tavalla seurataan kattilaveden lämpötilan mittauksen toimivuutta.

Happianturin vikaantumista ei seurata mitenkään, koska sen rikkoutumiselle ei ole mitään selkeää merkkiä. Sen hehkulle sähkö tulee erillisestä 12V virtalähteestä, jonka unohdin kerran laittaa päälle ja ainakin sen kokemuksen perusteella tulet eivät palaneet kunnolla ja liian alhainen savukaasun lämpötila pysäytti polttimen. Sitä pitäisi vielä testata lisää, jotta asiasta voisi olla varma.

PID-säätimissä on lähtö, jonka pitäisi mennä päälle ylivuodon sattuessa integroivassa osassa ja myös se aiheuttaa myös vikatilan.

Kuten aiemmin on kerrottu, hydraulipumpun tai varastoruuvin käydessä liian pitkään, ne pysäytetään. Se voi johtua etäisyysanturin häiriöstä tai hakkeen loppumisesta. Molemmat syöttävät haketta suurella voimalla ja voivat aiheuttaa vahinkoja, jos ne eivät pysähdy jostakin syystä ajoissa. Niiden meneminen vikatilaan ei kuitenkaan pysäytä poltinta, koska poltin voi toimia pitkänkin aikaa ilman niitä. Poltin pysähtyy vasta syöttöruuvin tai ilmapuhaltimen taajuusmuuttajan mennessä vikatilaan.

6.6 PID säätimet

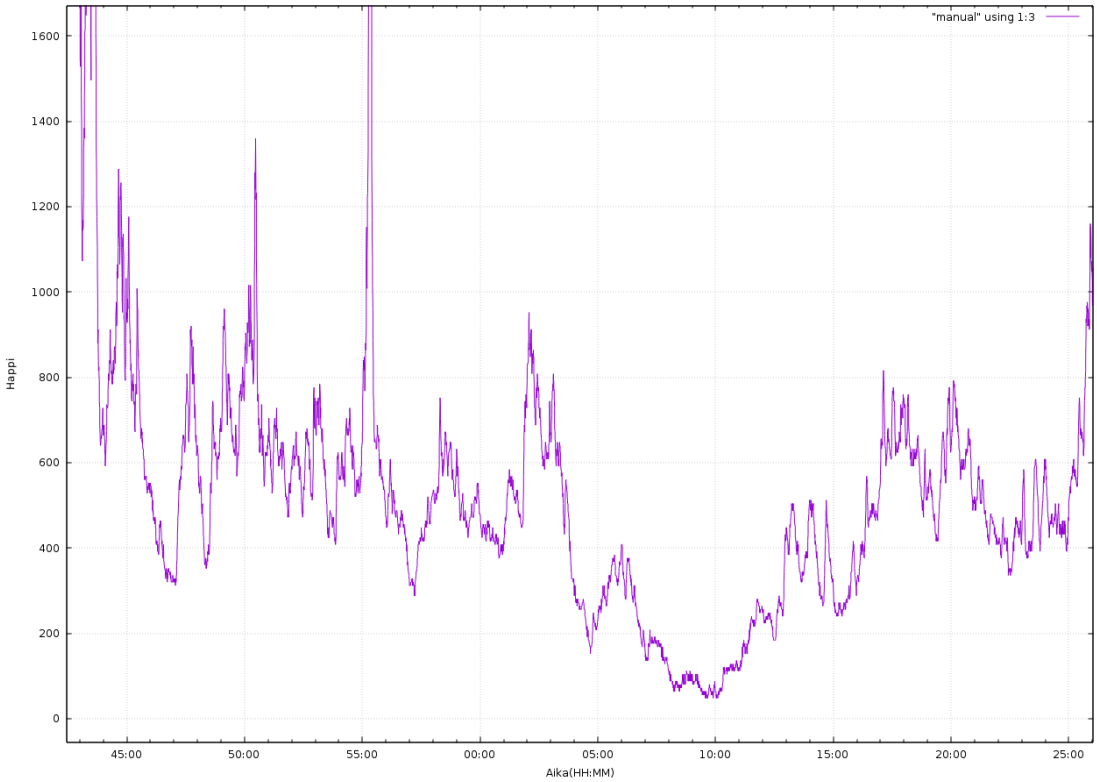
Syöttöruuvin ja puhaltimen nopeutta säädetään PID-säätimien avulla. Syöttöruuvin säädin saa oloarvonsa happianturilta ja puhaltimen säädin savukaasun lämpötilan an-

turilta. Aluksi tuntui johdonmukaisimmalta säätää ilman määrää happianturin mukaan ja säätää tulet sopiviksi syöttämällä haketta savukaasun lämpötilan mukaan, mutta se ei toiminut kovinkaan hyvin. Vaikka puhallusta säätäisi isommalle jäännöshapen määrän laskiessa, sillä ei ollut siihen kovinkaan suurta vaikutusta, mutta se vaikuttaa voimakkaasti savukaasujen lämpötilaan ja silloin hakkeen syötön muuttaminen vaikuttaisi savukaasujen lämpötilaan pakottaessaan puhaltimen pyörimään eri nopeudella.

Kun autokäyttöön asennetun happianturin asensi lämmityskattilaan ja kokeili sen toimintaa käytännössä, se ei loppujen lopuksi ollutkaan niin kapeakaistainen, kuin olisi voinut olettaa. Seoksen muuttuessa rikkaammaksi jännitteen jyrkkä nousu tapahtuu vasta 0,2-0,3V jännitteen yläpuolella ja anturi näyttäisi toimivan jokseenkin lineaarisesti välillä 0-0,2V. Operaatiovahvistin vahvistaa jännitteen suuremmaksi ja tämä alue näkyy logiikassa analogiatulon arvoina 0-1000. Jäännöshapen vähentyessä tästä edelleen, anturin antama jännite nousee erittäin jyrkästi ja analogiatulo ilmoittaa mittaamakseen arvokseen suunnilleen 20000. Poltinta olisi mahdollista käyttää tälläkin alueella, mutta silloin happea ei ilmeisesti ole riittävästi kaiken puun polttamiseen, koska polttimesta alkaa tuhkan sijasta tippumaan hiiltä, joka täyttää kattilan tuhkatilan nopeasti. Tämän polttimen kanssa sopiva ohjearvo oli aluksi noin 600, mutta ilmeisesti anturi kului käytössä ja se laski ajan saatossa 150-200 suuruusluokkaan. On myös mahdollista, että polttimen ilmareiät ovat tukkeutuneet, eikä se pysty enää polttamaan haketta niin tarkasti, tai kattilaan vuotaa jostakin ilmaa.

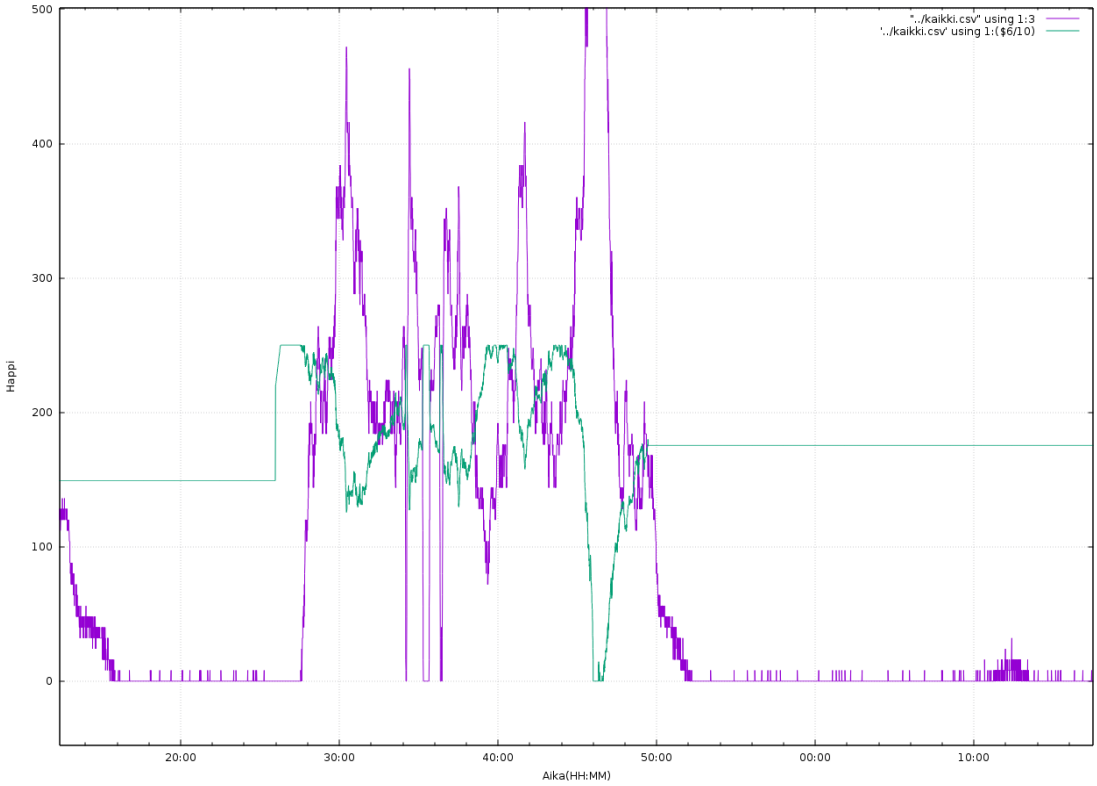
Logiikka tallentaa erilaisia tietoja SQL-tietokantaan ja tietokannasta voidaan ottaa tiedot ulos CSV-tiedostona, jota voi tarkastella vaikkapa Gnuplotilla. Logiikka ehtii tallentamaan tietokantaan 2-3 riviä sekunnissa, joten niitä kertyy sinne pian satoja tuhansia ja niiden tarkastelu on hankalaa taulukkolaskentaohjelmassa. Jos tallennusta jatketaan viikkoja, rivejä on lopulta useita miljoonia.

Ensiksi voidaan vaihtaa ohjauspaikka taajuusmuuttajissa kenttäväylältä näppäimistölle ja ajaa moottoreita vakionopeudella, jolloin saamme kiinnostavaa tietoa siitä, kuinka tulet ovat palaneet esimerkiksi vanhassa lämpökeskuksessa ja miten oloarvo käyttäytyy.



KUVA 36. Oloarvo käsikäytöllä. X-akselilla aika ilmoitetaan minuutteina ja sekunteina, ei tunteina ja minuutteina.

Kuvasta 36 näkee, että oloarvo vaihtelee hieman erikoisesti. Se pyrkii jatkuvasti joko kasvamaan tai pienenemään. PI-säätö toimi melko hyvin ja se pystyi pitämään seossuhteen suunnilleen sopivana, mutta se ei pystynyt estämään käsikäytöllä nähtyä rajua vaihtelua. Pieni säätimen kerroin ei riittänyt estämään heilahteluja ja kerrointa kasvattamalla säätö lähti värähtelemään hyvin helposti.

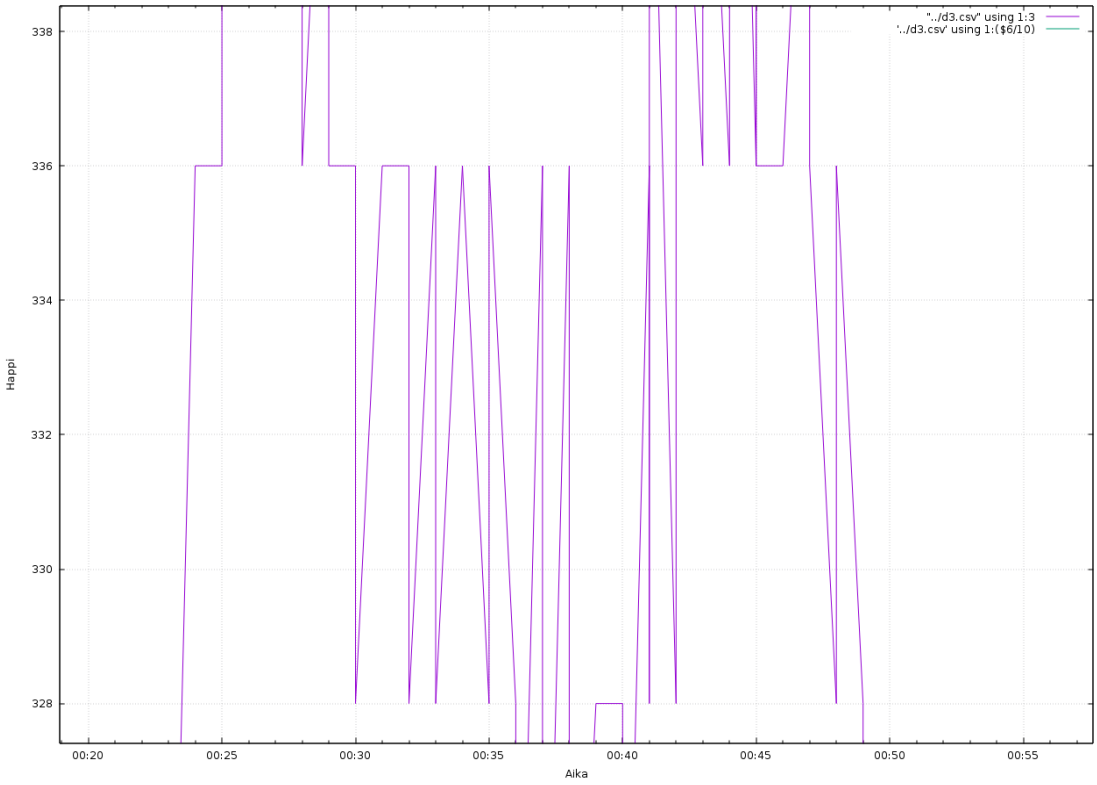


KUVA 37. PI-säätö. Hapen mittaus piirretty violetilla ja moottorin nopeusohje vihreällä. X-akselilla aika ilmoitetaan minuutteina ja sekunteina, ei tunteina ja minuutteina.

Kuvasta 37 voi nähdä, että PI-säätimen ryhtyessä korjaamaan virhettä, se on jo myöhäistä. Derivoiva säädin pystyisi ennakoimaan muutoksia paremmin, mutta se toimi hyvin huonosti. Util.lib-kirjastossa olevassa PID-säätimessä on käytetty DERIVATIVE-funktioblokkia, joka laskee lähtönsä arvon kolmesta peräkkäin mitatusta oloarvosta. Oloarvossa on myös paljon pieniä muutoksia molempiin suuntiin (kuva 38), jotka johtuvat luultavasti siitä, ettei logiikan analoginen tulo osaa päättää, onko oloarvo 328 vai 336. Siksi derivoiva säädin oli hyvin rauhaton ja saattoi määrätä moottorin nostamaan ja laskemaan kierroslukua monta kertaa sekunnissa, vaikka merkittävät muutokset kestävät useita sekunteja.

Lisäämällä suodatus säätimen tuloon muuttui säädin liian hitaaksi ja suodatusta pienentämällä säädin muuttui rauhattomaksi. Kun tutki DERIVATIVE-funktioblokkia tarkemmin avaamalla util.lib kirjasto Notepadilla, se näytti lukevan tulonsa ohjelman kiertonopeuden mukaan, eli suunnilleen 50ms välein, jolloin se seurasi oloarvon muu-

tosta suunnilleen 150ms ajalta, mikä on hyvin lyhyt aika. Oloarvon kuvaajalla kahden 150ms välein olevan pisteen välille piirretty viiva voi kulkea täysin eri suuntaan, kuin itse kuvaaja merkittävällä aikavälillä. Jos muutoksia seurattaisiin vaikkapa kahden sekunnin aikana, derivoiva säädin toimisi paremmin.



KUVA 38. Pieniä ja nopeita muutoksia happianturin signaalissa

DERIVATIVE-funktioblokissa on tulo TM, josta se saa mittausten välillä kuluneen ajan millisekunteina. PID-funktioblokissa on TON-ajastin, josta kulunut aika luetaan tähän tuloon ja ajastin nollataan. Tein kokeilumielessä uuden funktioblokin kopioimalla ja liittämällä util.lib-kirjastosta DERIVATIVE:n ohjelmakoodin ja annoin sille nimeksi ”derivaattori”. Samoin tein PID-funktioblokille ja se sai nimen ”uusipid”.

Uuteen PID-säätimeen lisäsin DAIKA-nimisen tulon, johon annetaan aika millisekunteina, jonka välein derivoiva säädin lukee tulon ja laskee uudelleen muutosnopeuden. Enää TM-tuloon ei anneta kulunutta aikaa, vaan säätimen DAIKA-tulosta käsin annettu aika. Säätimen integroiva osa tarvitsee edelleen ajastimen, joten sitä ei voi poistaa. Uuteen muutosnopeuden laskevaan funktioblokkiin on lisätty TON-ajastin ja sille an-

netaan ohjeeksi aika tulosta TM, jolloin se viivästyttää ohjelman kiertoa ja muutosnopeus lasketaan aina ajastimeen määrätyn ajan kuluttua. Nyt kulunutta aikaa ei mitata millään tavalla ja siitä voi seurata jossakin tilanteessa kummallisia ilmiöitä, mutta nyt se näyttäisi toimivan riittävän tarkasti tässä kokeilussa. Funktioblokkiin voisi lisätä toisen TON-ajastimen, joka mittaisi ajan ja sitä voisi käyttää laskemisessa.

Alun perin PID-funktioblokki käytti muutosnopeuden laskemiseen säätimen virhettä:

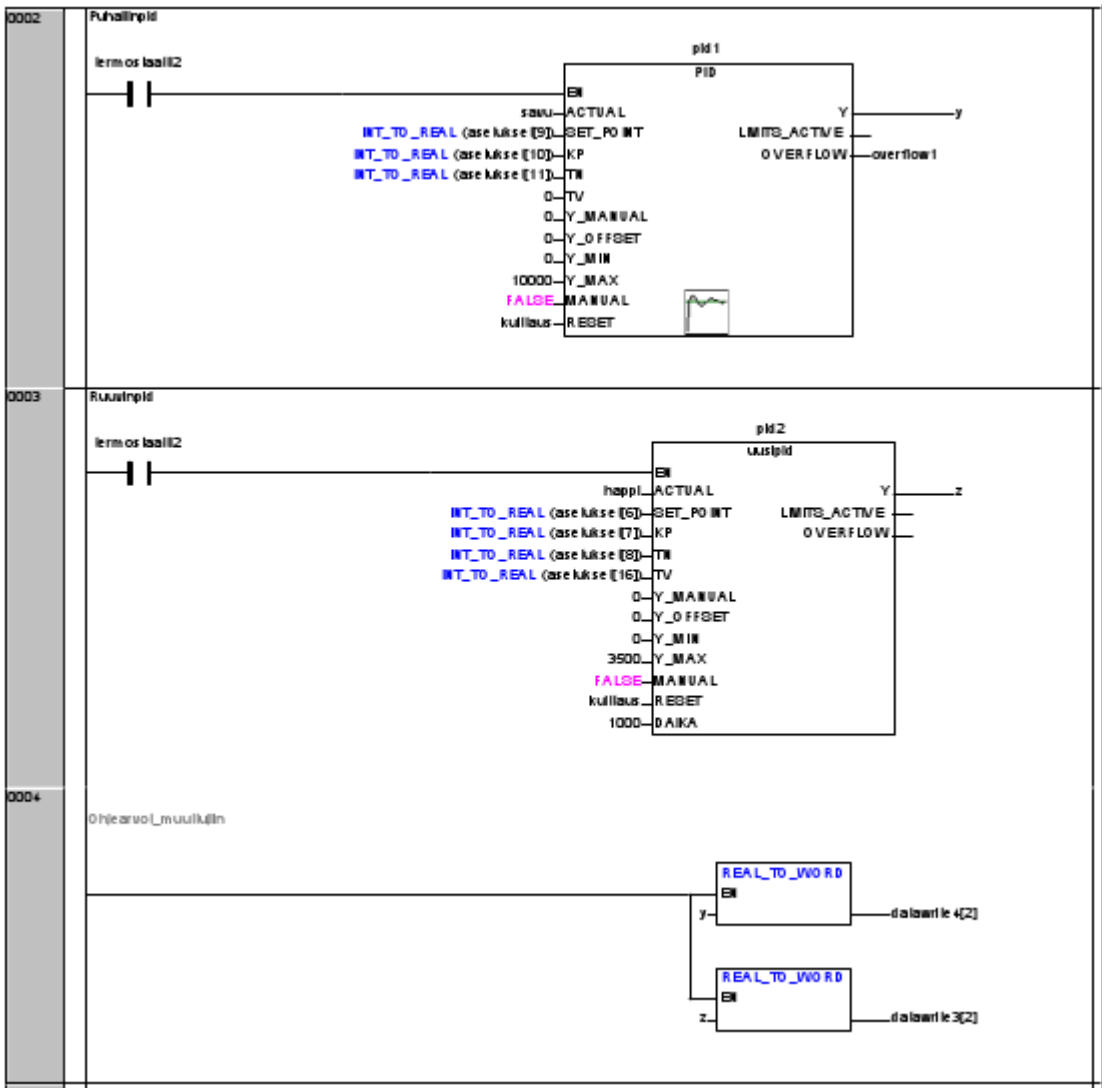
virhe = asetusarvo - oloarvo

Derivoiva säädin ei toiminut kovinkaan voimakkaasti, jos virhettä ei ollut paljoa. Koska oloarvo tahtoo lähteä satunnaisesti harhailemaan johonkin suuntaan ja siihen olisi hyvä reagoida heti voimakkaasti, oli parempi vaihtaa sen tilalle oloarvo sellaisenaan. Derivoivan osan lähtö muuttuu silloin käänteiseksi, ja siitä syystä piti myös säätimen lähdön laskukaavaa muuttaa niin, että derivoivan osan lähtö vähennetään summaamisen sijaan. Nyt säätimen lähtö hieman yksinkertaistettuna lasketaan:

$$P * \left(virhe + \frac{I.OUT}{TN} - D.OUT * TV \right)$$

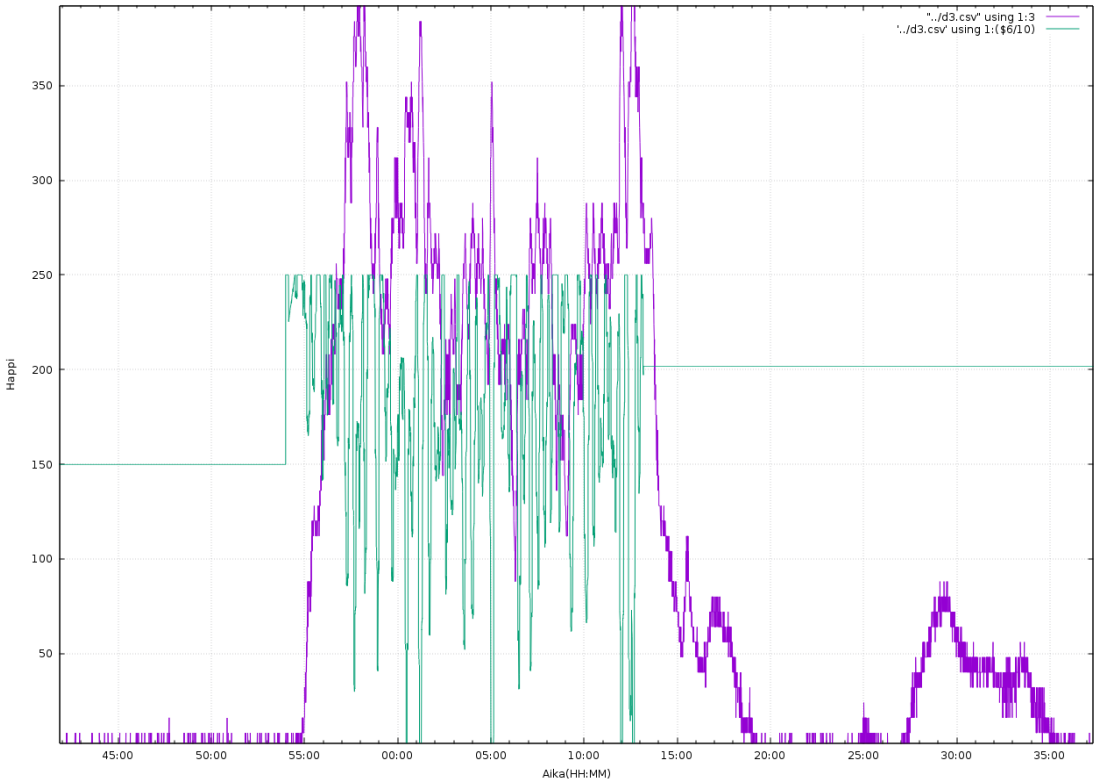
I.OUT on integroivan osan lähtö, TN sille annettu asetusarvo, D.OUT derivoivan osan lähtö ja TV sille annettu asetusarvo. Säätimen kerroin P vaikuttaa myös derivoivan säätimen voimakkuuteen, joten sen voisi siirtää pois sulkujen sisältä ja antaa sille oman kertoimensa. Silloin kerrointa voisi koettaa suurentaa ilman, että derivoiva osa alkaisi reagoida liian voimakkaasti.

Kuvassa 39 nähdään PID-säätimet. Säätimet antavat ohjearvon REAL-muuttujana, joten se pitää vaihtaa WORD-muuttujaan myöhempää käyttöä varten. Funktioblokit pysäytetään termostaatin kytkeytyessä pois päältä, jotta säätimien integroivat osat eivät ajaisi itseään ääriasentoon polttimen ollessa pysähdyksissä.



KUVA 39. Moottoreiden PID-säätö

Hieman muunnellulla PID-säätimellä moottorin nopeutta säädetään voimakkaasti ja säädin pystyy paremmin ennakoimaan muutokset (kuva 40).



KUVA 40. PID-säätö, oloarvo violetilla ja moottorin nopeusohje vihreällä. X-akselilla aika ilmoitetaan minuutteina ja sekunteina, ei tunteina ja minuutteina.

6.7 Tiedon tallentaminen SQL-tietokantaan

Logiikka pystyy myös tallentamaan tietoa SQL-tietokantaan. Tarkoitusta varten asensin vanhaan Thinkpad R50P -kannettavaan tietokoneeseen MariaDB-tietokantapalvelimen ja sen käyttämiseen Phpmyadminin, koska en kovinkaan hyvin osaa sitä käyttää komentorivin kautta. Tietokantapalvelin olisi yhtä hyvin voinut olla MySQL, mutta asensin MariaDB:n kokeilumielessä. Molemmat toimivat tässä tarkoituksessa täysin samalla tavalla.

Tein uuden ”mydb” nimisen tietokannan ja sinne uuden, ”kattila” nimisen taulun. Taulun rakenne on nähtävissä taulukossa 9.

TAULUKKO 9. Taulun rakenne

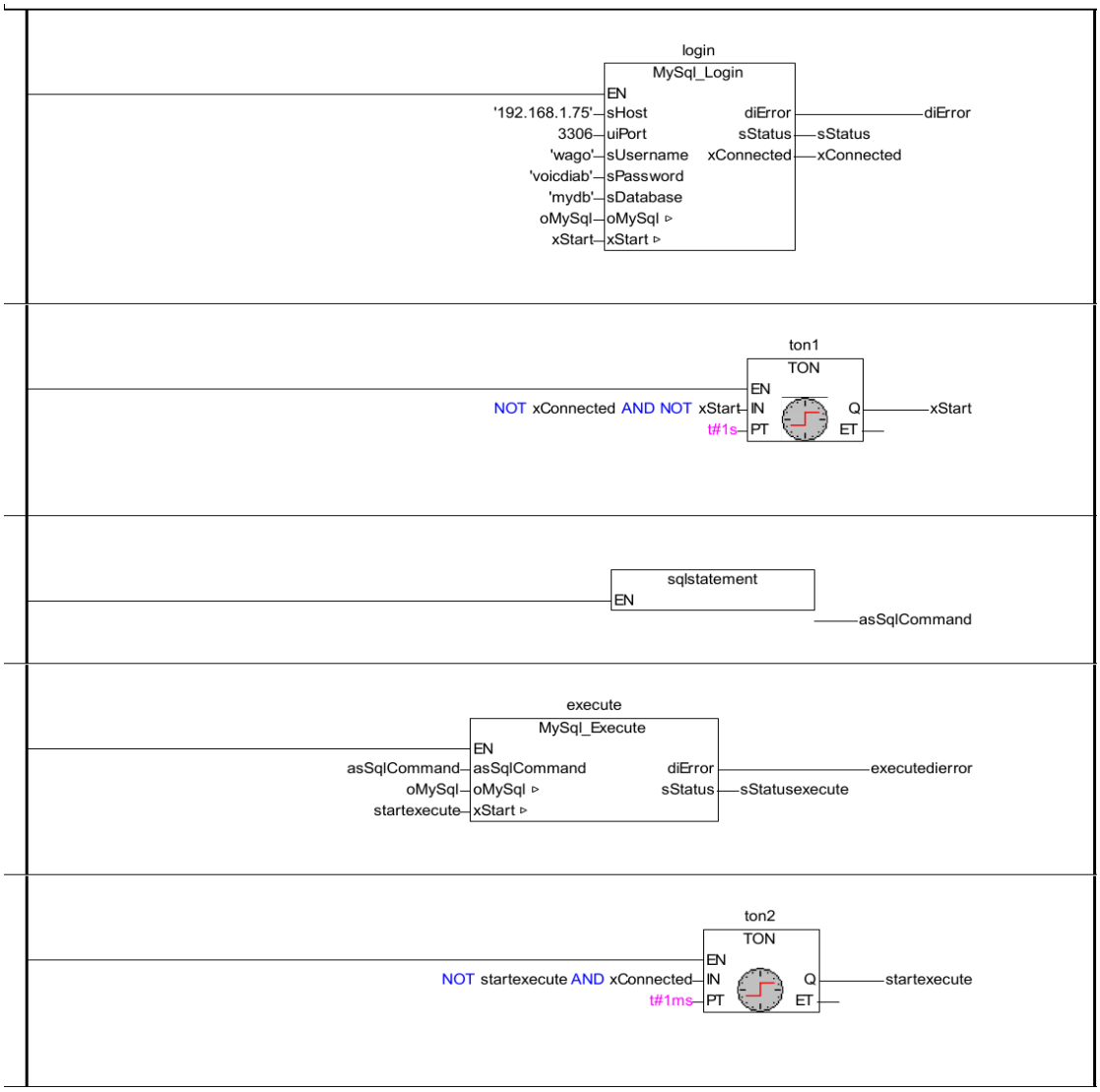
Column	Type	Null	Default
aika	timestamp	No	CURRENT_TIMESTAMP
idtable1	int(11)	No	
happi	smallint(5)	Yes	*NULL*
vesi	tinyint(3)	Yes	*NULL*
savu	tinyint(3)	Yes	*NULL*
ruuviohje	smallint(5)	Yes	*NULL*
puhallinohje	smallint(5)	Yes	*NULL*
ruuvipidvirhe	smallint(5)	Yes	*NULL*
ruuvipidint	smallint(5)	Yes	*NULL*
suodatettu	smallint(5)	Yes	*NULL*
suppilo	bit(1)	Yes	*NULL*
termostaatti	bit(1)	Yes	*NULL*
hydrpumppu	varchar(45)	Yes	*NULL*

Sarakkeiden tyypit yritin asettaa mahdollisimman tarkasti vastaamaan tallennettavan tiedon kokoa, jotta levytilaa säästyisi ja vanhan tietokoneen työ helpottuisi. Näin tietokanta vie levytilaa vain muutamia satoja megatavuja, vaikka siinä on miljoona riviä. Hydraulipumpun käynnille on unohtunut varchar muuttuja ja siinä tarpeettoman suureen muuttujaan tallennetaan kyllä/ei tietoa.

Reitittimen dhcp palvelin on määritelty antamaan aina sama IP-osoite SQL-palvelimelle, jotta se ei vaihdu yllättäen ja logiikka ei yritä löytää sitä vanhasta osoitteesta. Kiinteän IP-osoitteen voi myös määritellä palvelimeen, mutta silloin pitää varoa käyttämästä sellaista osoitealuetta, josta DHCP palvelin valitsee osoitteet, jotta se ei anna samaa osoitetta jollekin muulle laitteelle.

Itse ohjelma logiikkaan on tehty WagoLibMysql_03.lib-kirjaston ohjeessa olevan esimerkin mukaisesti. Ohjelmassa tarvitaan kaksi funktioblokkia, ”MySql_Login”, joka huolehtii kirjautumisesta palvelimelle ja ”MySql_Execute” joka antaa palvelimelle käskyn, joka on tässä tapauksessa uuden rivin lisääminen tallennettavilla tiedoilla. Se pystyisi myös lukemaan tietokantaa ja esimerkiksi käyttämään jotakin konetta sen mukaan.

Ajastimella käynnistetään kirjautuminen uudelleen sekunnin kuluttua ja toista ajastinta voidaan lisätä rivien lisäämisten välillä kuluvaa aikaa. Kuvassa 41 rivejä yritetään kirjoittaa yhden millisekunnin välein, mutta käytännössä rivin lisääminen kestää niin kauan, että niitä syntyy vain kolme kappaletta sekunnissa. MySql_Login-funktioblokille annetaan palvelimen ip-osoite, portti, palvelimelle määritelty käyttäjänimi ja salasana, sekä tietokanta, jota käytetään.



KUVA 41. SQL-tietokannan käyttöön tarvittava ohjelma

Itse SQL-komennon tekeminen on hieman työlästä. Tein uuden funktioblokin, jossa se muotoillaan asSqlCommand-muuttujaan, josta MySql_Execute-funktioblokki voi sen lukea (kuva 42). Komento on jaettu ”sqlstatement” taulukon eri riveille. ”INSERT INTO kattila” on komennon alku, jossa INSERT-komennolla kirjoitetaan ”kattila” ni-

miseen tauluun. Seuraavaksi suluissa on pilkuilla erotettuina kirjoitettavat taulun sarakkeet. Sen jälkeen ”VALUES” ilmoittaa samassa järjestyksessä arvot, jotka ilmoitettuihin sarakkeisiin kirjoitetaan. Koska on tehtävä paljon muunnoksia STRING-muuttujaan ja erotettava ne pilkulla, ne kannattaa jakaa eri riveille. Teksti pitää olla suljettuna heittomerkkien sisälle ja rivin päättyä puolipisteeseen, eli myös arvoja erottava pilkku pitää olla heittomerkkien välissä. Kuvan komennossa on hieman erilaiset muuttujat, kuin varsinaisessa ohjelmassa.

```
0001  FUNCTION sqlstatement :ARRAY [0..gcMySql_iSqlUpperBound] OF STRING(gcMySql_iSqlLength)
0002  VAR_INPUT
0003  END_VAR
0001  (* Prepare sql-INSERT-Statement *)
0002  sqlstatement[0] := 'INSERT INTO kattila ';
0003  sqlstatement[1] := '(happi, savu, vesi, ruuviohje, puhallinohje, ruuvipidvirhe, ruuvipidint, suppilo, termostaatti, hydrpumppu, suodatettu) ';
0004  sqlstatement[2] := 'VALUES (';
0005  sqlstatement[3] := REAL_TO_STRING(happi);
0006  sqlstatement[4] := ',';
0007  sqlstatement[5] := INT_TO_STRING(apumuuttujat[3]);
0008  sqlstatement[6] := ',';
0009  sqlstatement[7] := INT_TO_STRING(apumuuttujat[0]);
0010  sqlstatement[8] := ',';
0011  sqlstatement[9] := WORD_TO_STRING(datawrite3[2]);
0012  sqlstatement[10] := ',';
```

```
0013  sqlstatement[11] := WORD_TO_STRING(datawrite4[2]);
0014  sqlstatement[12] := ',';
0015  sqlstatement[13] := INT_TO_STRING(apumuuttujat[5]);
0016  sqlstatement[14] := ',';
0017  sqlstatement[15] := INT_TO_STRING(apumuuttujat[6]);
0018  sqlstatement[16] := ',';
0019  sqlstatement[17] := BOOL_TO_STRING(suppilo);
0020  sqlstatement[18] := ',';
0021  sqlstatement[19] := BOOL_TO_STRING(termostaatti2);
0022  sqlstatement[20] := ',';
0023  sqlstatement[21] := BOOL_TO_STRING(pumppu);
0024  sqlstatement[22] := ',';
0025  sqlstatement[23] := WORD_TO_STRING(dataread3[7]);
0026  sqlstatement[24] := ')';
0027  sqlstatement[25] := ''; (* End of SQL -Statement *)
```

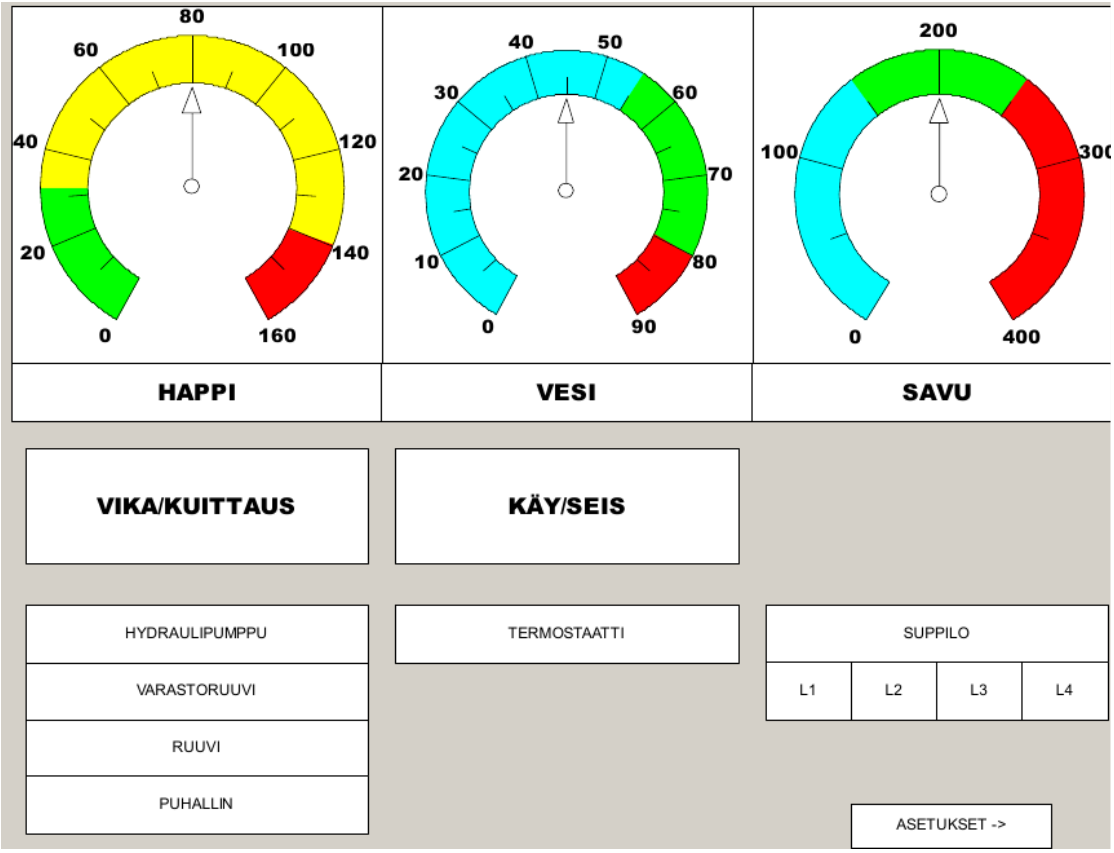
KUVA 42. SQL-komennon luominen

Oletuksena sqlstatement taulukossa ei ole kovinkaan montaa riviä ja rivien pituus on rajoitettu melko lyhyeksi. Niitä pystyy muuttamaan määrittelemällä seuraavat muuttujat:

```
VAR_GLOBAL CONSTANT
gcMySql_iSqlUpperBound: INT := 50;
gcMySql_iSqlLength: INT := 120;
END_VAR
```

6.8 Käyttöliittymä

Poltinta ei tässä vaiheessa voi ohjata painikkeilla lainkaan, vaan se tapahtuu esimerkiksi tietokoneen verkkoselaimella. Logiikkaan voidaan tehdä verkon yli toimiva käyttöliittymä ja lisätä sinne tarvittavat painikkeet, mittarit ja ilmaisimet. Myöhemmin painikkeet on lisättävä ohjauskeskukseen, koska vaikkapa edullisen kotikäyttöön tarkoitetun reitittimen rikkoutuminen estää, tai ainakin hankaloittaa polttimen käynnistystä. Kuvassa 43 nähdään Codesys Webvisualization -käyttöliittymä, josta nähdään happianturin antama tieto, veden ja savukaasun lämpötilat, painikkeet, moottoreiden käynti ja joitakin digitaalisia tietoja. L1, L2 jne ovat tankopurkainten toimintaa ohjaavien etäisyysantureiden ”merkkivalot”.



KUVA 43. Polttimen käyttöliittymä

Pääsy logiikkaan on rajoitettava omaan lähiverkkoon, koska se ei ole riittävän turvallinen kestämään internetistä tulevia hyökkäyksiä ja sen ohjelmissa on monia vanhoja

haavoittuvuuksia, koska sitä ei päivitetä. Samalla polttimen tarkkailu ja ohjaus internetin yli hankaloituu huomattavasti. Se on kuitenkin mahdollista muodostamalla yhteys lähiverkkoon VPN:n tai SSH:n avulla, tosin se on hieman työlästä ja vaatii perehtymistä.

Koska lämpökeskusta käytetään ainoastaan oman talon lämmittämiseen, eikä lämpöä toimiteta asiakkaille, ei kovinkaan suurta tarvetta etävalvontaan ole. Polttimen jostakin syystä pysähtyessä sähkövastukset alkavat lämmittää kiertovettä, putkien jäätymisvaaraa ei ole, eikä tilapäinen lämmittäminen sähköllä maksa paljoa. Vaikka tiedon häiriöstä saisi puhelimeen poissa ollessa, ei sille kuitenkaan voi tehdä mitään. Ennen vian kuittausta on viisasta selvittää sen syy, eikä se onnistu etäkäytöllä. Omasta mielestäni polttimen pitäisi toimia ilman jatkuvaa valvontaa, joten sen tarkkailu on turhaa työtä ja häiritsee mielenrauhaa.

7 YHTEENVETO

Jäännöshapen määrän perusteella toimiva hakepoltin näyttää toimivan melko hyvin ja työtä tehdessä syntynyt lämpökeskus tulee toimimaan hyvin vuosikausia, kunhan kaapeloinnit asentaa kunnolla ja siististi lopullisille paikoilleen. Hakkeen kulutus on ollut yllättävän vähäistä ja varastoon mahtuu pelkän talon lämmittämiseen haketta melkein kahdeksi vuodeksi.

Autokäyttöön tehty halpa kapeakaistainen happianturi ei tämän työn perusteella ole ominaisuuksiltaan täydellinen hakepolttimen ohjaukseen, mutta riittävästi sen antamaa jännitettä vahvistamalla se toimii tarpeeksi hyvin varsinaisen toiminta-alueensa vieressä, enkä osaa mainita syytä, miksi sen tilalla pitäisi käyttää laajakaistaisempaa anturia. Myös anturin käyttöikä näyttäisi olevan yllättävän pitkä. Päätin ensin kokeilla kaverilta saatua käytettyä happianturia ja ostaa sen rikkouduttua tilalle uuden, jos se toimii riittävän hyvin. Uutta anturia ei tarvinnut vaihtaa, vaan tätä yhteenvetoa kirjoitettaessa vanha anturi on käytössä edelleen, tosin sen antama jännite on laskenut hieman, jonka voi korjata muuttamalla asetusarvoa. Jännitteen laskeminen voi myös johtua ilmavuodosta kattilan luukussa. Tarkoituksena oli lisätä anturin hehkuvirran syötölle rele, joka sammuttaisi hehkun polttimen pysähdyttyä, mutta en ehtinyt sitä siihen koskaan lisäämään ja anturin hehku on päällä jatkuvasti. Jatkuva hehkuttaminenkaan ei siis näytä

tuhoavan anturia erityisen nopeasti.

Pannuhuoneessa tarvitaan polttimen ja syöttöruuvien lisäksi myös shunttiventtiileitä moottoreineen ja helposti laajennettavan logiikan ansiosta niiden lisääminen myöhemmin tulee olemaan helppoa, eikä niille tarvitse hankkia erillistä ohjausta. Silloin myös verkon yli toimivan käyttöliittymän avulla voidaan esimerkiksi autotallin lämmitys kytkeä helposti päälle puhelimella, verkkoselaimella tai tavallisella kytkimellä, eikä ole tarvetta mennä pannuhuoneeseen säätämään shunttiventtiiliä ja käynnistämään kiertovesipumppua.

Työssä jäi selvittämättä, kuinka paljon hakkeen lämpöenergiasta saadaan talteen. Kattilasta otettu lämpöenergia on helppoa mitata, mutta hakkeen määrän mittaamiseen en keksinyt hyvää ja luotettavaa keinoa. Koska haketta on tehty sekalaisesta puusta, jossa on seassa enemmän tai vähemmän lahonnutta puuta, myös hakkeen sisältämän energian määrää on hankalaa arvioida. Hakkeen kulutus oli kuitenkin vähäisempi, kuin olisin odottanut ja savupiipusta nousee polttimen käydessä sankka vesihöyry, joka tiivistyy piipun hattuun ja muodostaa pakkasella jääpuikkoja piipun ympärille. Se voisi olla vihje siitä, että ilmaa ei puhalleta turhaan kattilan läpi ja hyötysuhde on hyvä. Sitä ei tapahtunut ennen jäännöshapen mittauksen ottamista käyttöön.

Hakkeen syötön säätö ei pysty pitämään jäännöshapen määrää vakiona, mutta parannus on valtava verrattuna vakionopeudella pyörivään ruuviin. Savukaasuja ei ole analysoitu millään tarkemmilla mittalaitteilla, eikä virheiden suuruusluokasta ole tarkempaa tietoa. Tein säädön niin tarkaksi, kuin vain pystyin tässä ajassa tekemään ja se yrittää parhaansa mukaan pitää jäännöshapen määrän vakiona. Anturin antaman tiedon perusteella tiedetään se, että aivan kaikkea happea ei käytetä palamisessa ja koska poltin ei pysty polttamaan kaikkea haketta täydellisesti jäännöshapen määrää pienennettäessä, se käyttää polttimeen puhalletun hapen niin tarkasti, kuin mahdollista. Kattila pysyy puhtaana polttimen käydessä ja peittyy harmaaseen tuhkaan, eikä savupiipusta nouse savua, joten palaminen on melko puhdasta. Jos tällainen säätö rakennettaisiin suurempaan polttimeen, se luultavasti käyttäytyisi eri tavalla ja sitä saattaisi olla helpompaa ohjata. Olisi ehkä mahdollista, että jokin toinen poltin voisi toimia niinkin pienellä ilmaylimäärällä, että happianturi voisi toimia varsinaisella toiminta-alueellaan.

Koska työ onnistui paremmin, kuin uskalsin odottaa, aion tulevaisuudessa muuttaa vanhemmankin lämpökeskuksen toimimaan jäännöshapen mittauksen avulla. En työhön ryhtyessäni osannut ohjelmoida Codesysillä, enkä koskaan aiemmin ole ollut tekemisissä kenttäväylien kanssa. Niiden oppiminen tätä opinnäytetyötä tehdessä auttaa tulevaisuudessa monessa muussa asiassa.

Tein ohjauskeskuksen mielenkiinnosta ja omaksi hyödykseni. Sain siitä vastaukset mieltä vuosikausia askarruttaneisiin asioihin, enkä aio tehdä siitä kaupallista tuotetta. Sen sijaan toivon, että tästä työstä olisi apua jollekin muulle.

LÄHTEET

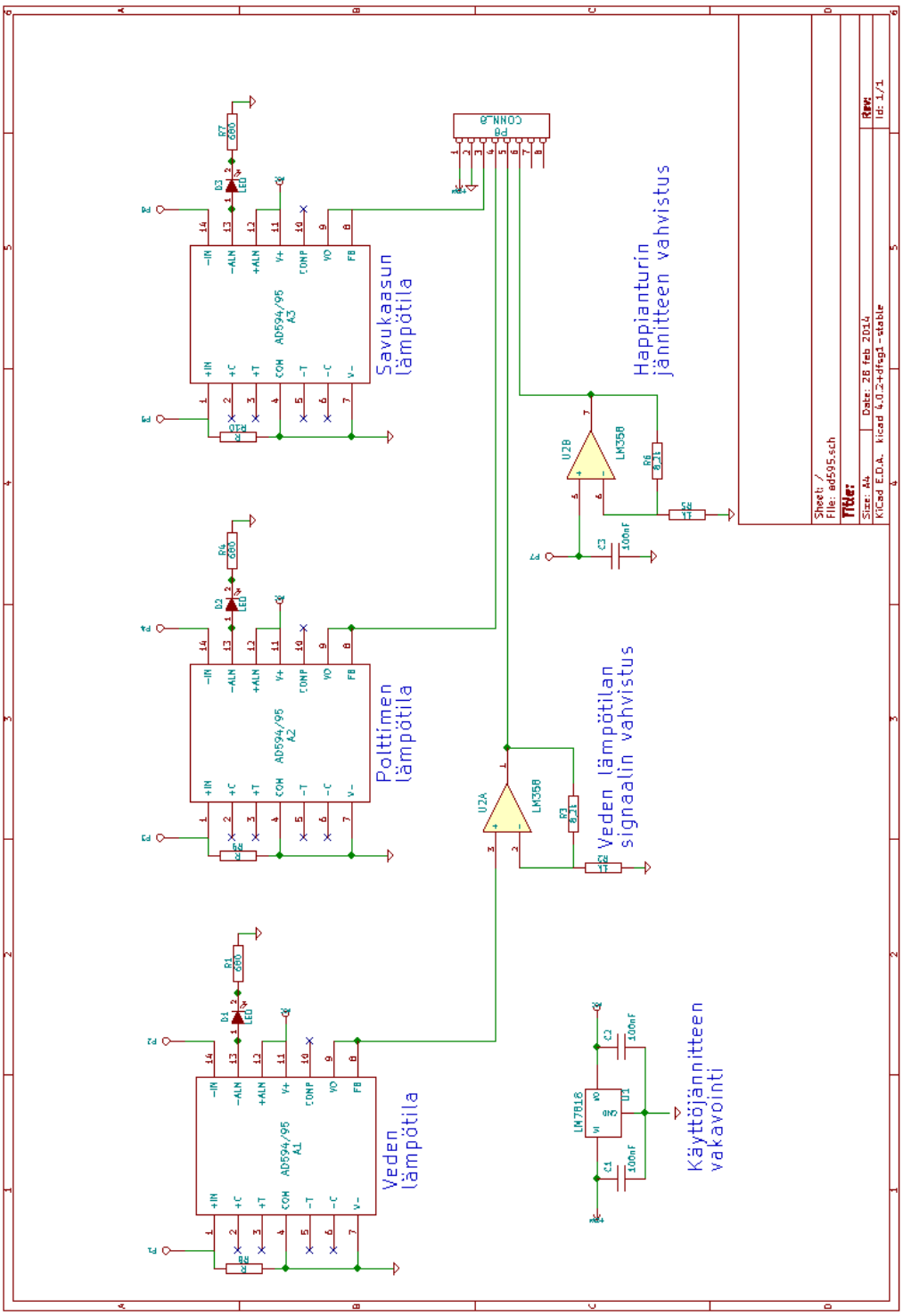
1 Metsäntutkimuslaitos. Metsätilastotiedote 25/2014. Verkkodokumentti. http://www.metla.fi/tiedotteet/metsatilastotiedotteet/2014/energiapuu14_1-3.htm Päivitetty 16.10.2014. Luettu 27.3.2016.

2 NGK. How does the Lambda sensor work? Verkkodokumentti. http://www.ngkntk.co.uk/index.php/technical-centre/lambda-sensors/how-does-the-lambda-sensor-work/
Päivitetty 14.1.2013. Luettu 15.4.2016.

3 Wahlroos Lasse 1980. Kotimaiset polttoaineet ja keskuslämmityskattilat. Pori: Energiakirjat.

LIITE 1.

Analogisten tietojen mittaus ja vahvistus



LIITE 2

Logiikan ohjelma

Filename:	valmis.pro
Directory:	E:\codesys
Change date:	24.11.16 19:10:53 / V2.3
Title:	
Author:	
Version:	
Description:	

```
0001  FUNCTION_BLOCK derivaattori
0002  VAR_INPUT
0003      IN:REAL;                    (* input variable *)
0004      TM:DWORD;                   (* time since last call in msec *)
0005      RESET:BOOL;                 (* reset: set OUT to zero *)
0006  END_VAR
0007  VAR_OUTPUT
0008      OUT:REAL;               (* derivative *)
0009  END_VAR
0010  VAR
0011      X3,X2,X1:REAL;
0012      T2,T1:DWORD;
0013      INIT:BOOL:=TRUE;
0014      TON1:TON;
0015      tonq:BOOL;
0016
0017      ajastus: BOOL;
0018  END_VAR
```

```
0001  IF INIT OR RESET THEN
0002      X1:=IN;
0003      X2:=IN;
0004      X3:=IN;
0005      OUT:=0;
0006      INIT:=FALSE;
0007
0008  ELSE
0009
0010      TON1(IN:=ajastus, PT:=DWORD_TO_TIME(TM) , Q=>tonq);
0011
0012      ajastus:=TRUE;
0013
0014      IF TM>0 AND tonq=TRUE THEN
0015          OUT:=(3*(IN-X3)+X1-X2)/((3*T2+4*T1+3*TM))*1000;
0016          X3:=X2;
0017          X2:=X1;
0018          X1:=IN;
0019          T2:=T1;
0020          T1:=TM;
0021          ajastus:=FALSE;
0022      END_IF;
0023  END_IF;
0024
```

modbus (PRG-ST)

```
0001  PROGRAM modbus
0002  VAR
0003      M1: MODBUS_MASTER_RTU;
0004  (*    send:typRING_BUFFER; *)
0005      state: INT:=1;
0006  (*    start: BOOL:=FALSE;*)
0007
0008      i                   :   INT;
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018  END_VAR
0019  (*VAR
0020      SendActive          :   BOOL;
0021      ReceiveBuffer       :   typRING_BUFFER;
0022      SendBuffer          :   typRING_BUFFER;
0023      Count               :   INT;
0024      CRC                 :   WORD;
0025      Schnittstelle_1     :   SERIAL_INTERFACE;
0026      ExpectedResponse    :   INT;
0027      Timer               :   TON;
0028      TimeOutPointer      :   INT;
```

```
0029        CALC_CRC        :    CRC16;
0030        i               :    INT;
0031        TriggerTimeOut  :    BOOL;
0032        Reset           :    BOOL := TRUE;
0033
0034    END_VAR *)
0035
```

```
0001    CASE state OF
0002
0003    (*0:
0004
0005
0006    IF NOT StartFunction THEN
0007    FOR i:=0 TO M1.ReceiveBuffer.Data[3] DO
0008    dataread1[i]:=M1.ReceiveBuffer.Data[3+i];
0009    END_FOR
0010
0011    Query.FunctionCode:=03;
0012    Query.SlaveAddress:=02;
0013    Query.StartAddress:=2100;
0014    Query.Quantity:=11;
0015    StartFunction:=TRUE;
0016    state:=1;
0017    END_IF
0018    *)
0019
0020    1:
0021
0022    IF NOT StartFunction THEN
0023
0024
0025
0026    Query.FunctionCode:=03;
0027    Query.SlaveAddress:=02;
0028    Query.StartAddress:=2100;
0029    Query.Quantity:=11;
0030    StartFunction:=TRUE;
0031    state:=2;
0032    END_IF
0033
0034
0035    2:
0036
0037    IF NOT StartFunction THEN
0038        IF M1.ReceiveBuffer.Data[1]=Query.FunctionCode AND M1.ReceiveBuffer.Data[0]=Query.SlaveAddress AND MB_Error=0 THEN
0039            FOR i:=0 TO Response.Quantity DO
0040            dataread2[i]:=Response.Data[i];
0041            END_FOR
0042
0043        Query.FunctionCode:=03;
0044        Query.SlaveAddress:=03;
0045        Query.StartAddress:=2100;
0046        Query.Quantity:=11;
0047        StartFunction:=TRUE;
0048        state:=3;
0049        ELSE
0050        StartFunction:=TRUE;
0051        END_IF
0052    END_IF
0053
0054    3:
0055
0056    IF NOT StartFunction THEN
0057        IF M1.ReceiveBuffer.Data[1]=Query.FunctionCode AND M1.ReceiveBuffer.Data[0]=Query.SlaveAddress AND MB_Error=0 THEN
0058
0059            FOR i:=0 TO Response.Quantity DO
0060            dataread3[i]:=Response.Data[i];
0061            END_FOR
0062
0063        Query.FunctionCode:=03;
0064        Query.SlaveAddress:=04;
0065        Query.StartAddress:=2100;
0066        Query.Quantity:=11;
```

```
0067        StartFunction:=TRUE;
0068        state:=4;
0069        ELSE
0070        StartFunction:=TRUE;
0071        END_IF
0072    END_IF
0073    4:
0074
0075    IF NOT StartFunction THEN
0076        IF M1.ReceiveBuffer.Data[1]=Query.FunctionCode AND M1.ReceiveBuffer.Data[0]=Response.SlaveAddress AND MB_Error=0 THEN
0077
0078            FOR i:=0 TO Response.Quantity DO
0079            dataread4[i]:=Response.Data[i];
0080            END_FOR
0081
0082            Query.FunctionCode:=16;
0083            Query.SlaveAddress:=02;
0084            Query.StartAddress:=2000;
0085            Query.Data[0]:=datawrite2[0];
0086            Query.Data[1]:=0;
0087            Query.Data[2]:=datawrite2[2];
0088
0089
0090            (*IF varasto=FALSE THEN
0091            Query.FunctionCode:=06;
0092            Query.SlaveAddress:=02;
0093            Query.StartAddress:=2000;
0094            Query.Data[0]:=0;
0095            Query.Data[1]:=0;
0096            END_IF
0097
0098            IF varasto=TRUE THEN
0099            Query.FunctionCode:=06;
0100            Query.SlaveAddress:=02;
0101            Query.StartAddress:=2000;
0102            Query.Data[0]:=1;
0103            END_IF*)
0104        ELSE
0105        StartFunction:=TRUE;
0106        END_IF
0107    StartFunction:=TRUE;
0108    state:=5;
0109    END_IF
0110
0111    5:
0112
0113    IF NOT StartFunction THEN
0114
0115        Query.FunctionCode:=16;
0116        Query.SlaveAddress:=03;
0117        Query.StartAddress:=2000;
0118        Query.Data[0]:=datawrite3[0];
0119        Query.Data[1]:=0;
0120        Query.Data[2]:=datawrite3[2];
0121
0122
0123        (*IF ruuvi=TRUE THEN
0124        Query.FunctionCode:=06;
0125        Query.SlaveAddress:=03;
0126        Query.StartAddress:=2000;
0127        Query.Data[0]:=1;
0128        Query.Data[1]:=0;
0129        END_IF
0130
0131        IF ruuvi=FALSE THEN
0132        Query.FunctionCode:=06;
0133        Query.SlaveAddress:=03;
0134        Query.StartAddress:=2000;
0135        Query.Data[0]:=0;
0136        Query.Data[1]:=0;
0137        END_IF*)
0138
0139    StartFunction:=TRUE;
```

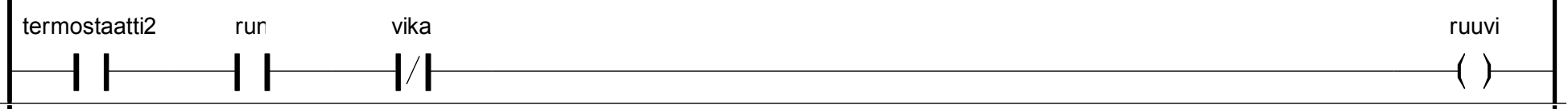
```
0140  state:=6;
0141  END_IF
0142
0143  6:
0144
0145  IF NOT StartFunction THEN
0146
0147
0148      Query.FunctionCode:=16;
0149      Query.SlaveAddress:=04;
0150      Query.StartAddress:=2000;
0151      Query.Data[0]:=datawrite4[0];
0152      Query.Data[1]:=0;
0153      Query.Data[2]:=datawrite4[2];
0154
0155
0156  (*
0157      IF puhallin=TRUE THEN
0158      Query.FunctionCode:=06;
0159      Query.SlaveAddress:=04;
0160      Query.StartAddress:=2000;
0161      Query.Data[0]:=1;
0162      Query.Data[1]:=0;
0163      END_IF
0164
0165      IF puhallin=FALSE THEN
0166      Query.FunctionCode:=06;
0167      Query.SlaveAddress:=04;
0168      Query.StartAddress:=2000;
0169      Query.Data[0]:=0;
0170      Query.Data[1]:=0;
0171      END_IF
0172  *)
0173  StartFunction:=TRUE;
0174  state:=1;
0175  END_IF
0176
0177
0178  END_CASE;
0179
0180
0181  M1(
0182      bCOM_PORT:= 16#00,
0183      cbCOM_BAUDRATE:= 1920,
0184      cpCOM_PARITY:= 0,
0185      csCOM_STOPBITS:= 2,
0186      cbsCOM_BYTESIZE:= 8,
0187      cfCOM_FLOW_CONTROL:= 4,
0188      TimeOut:=t#1000ms,
0189      StartFunction:= StartFunction,
0190      Query:= Query,
0191      Response:= Response,
0192      MB_Error=>MB_Error
0193   );
```

moottorit (PRG-LD)

```
0001  PROGRAM moottorit
0002  VAR
0003
0004
0005
0006
0007      unpack2: UNPACK;
0008      unpack3: UNPACK;
0009      unpack4: UNPACK;
0010
0011      tof1: TOF;
0012      tof2: TOF;
0013
0014  END_VAR
```

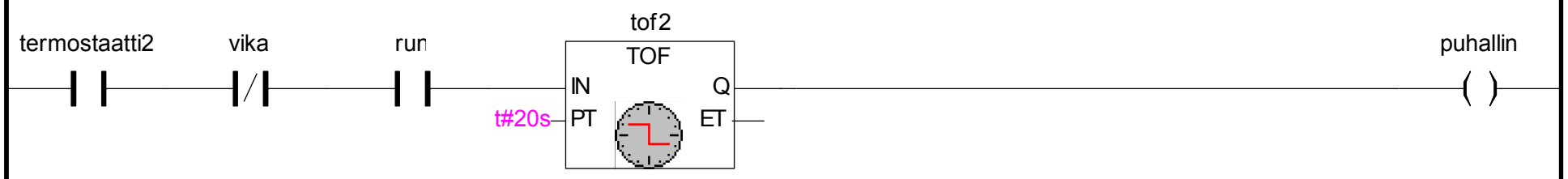
0001

Syöttöruuvin ohjaus



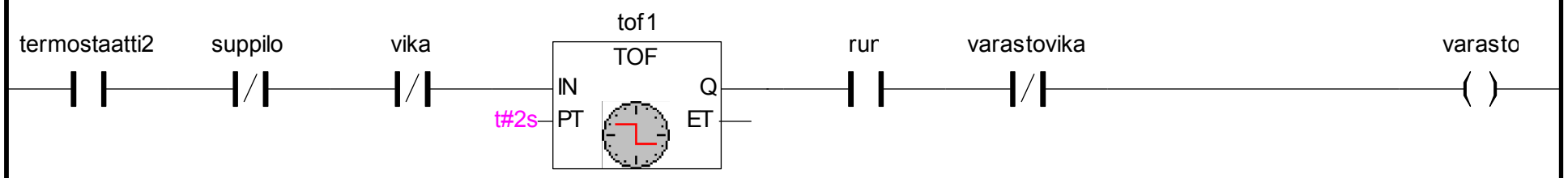
0002

Puhaltimen ohjaus



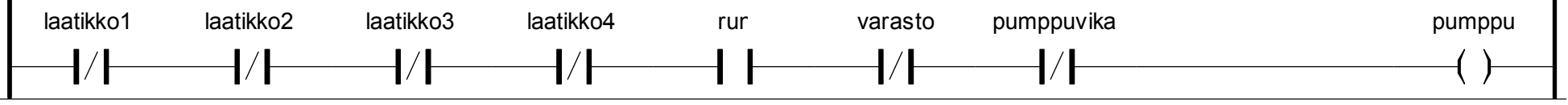
0003

Varastoruuvin ohjaus



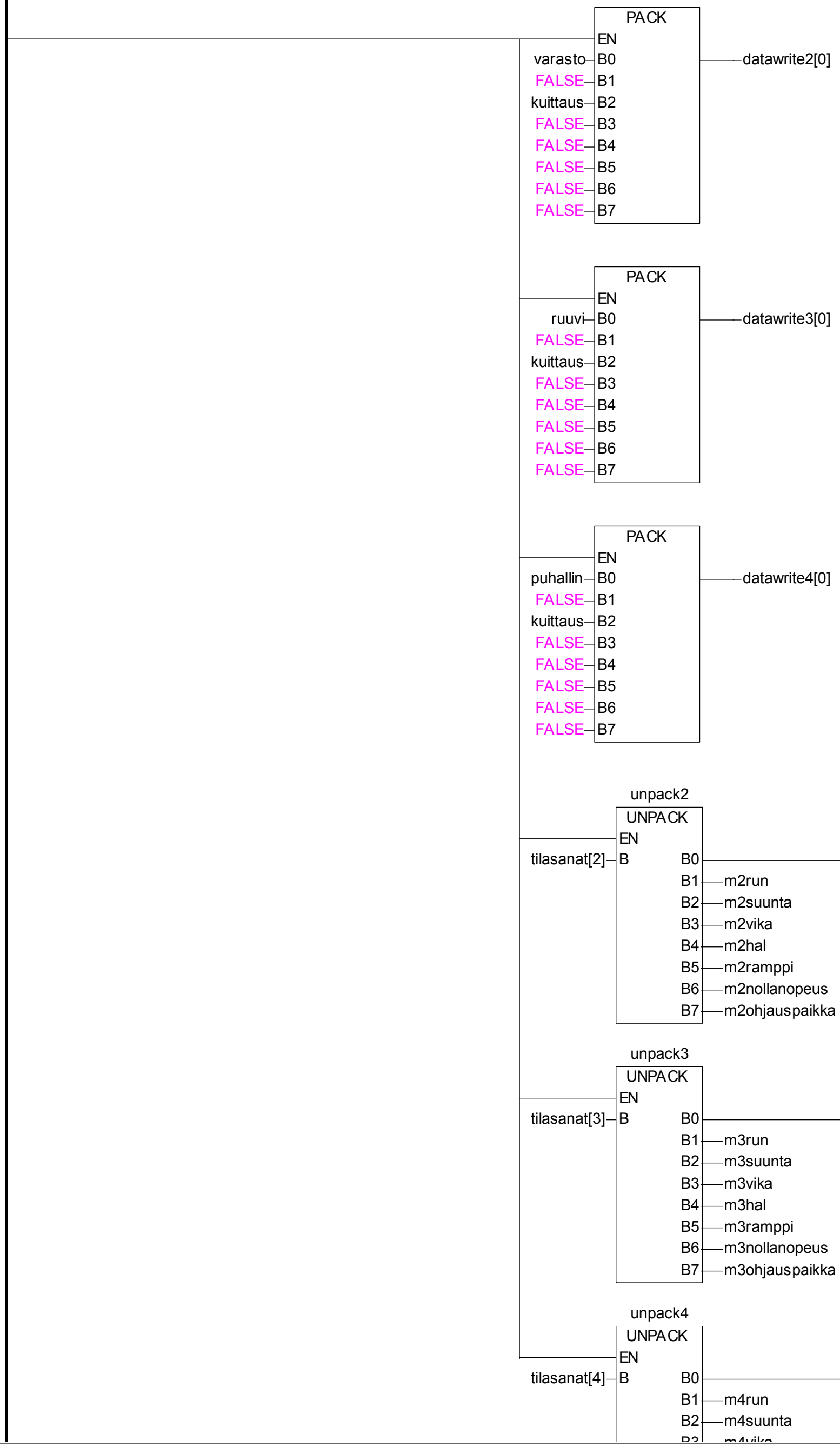
0004

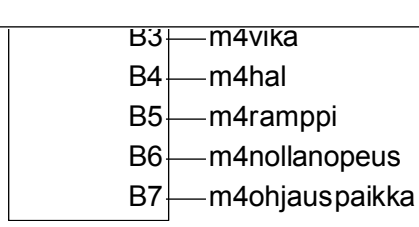
Hydraulipumppu



0005

komentosanan_bitit





muunnokset (PRG-ST)

```
0001  PROGRAM muunnokset
0002  VAR
0003      HI AT %MB0: BYTE;
0004      LO AT %MB1: BYTE;
0005      SANA AT %MW0: WORD;
0006  END_VAR
0007
```

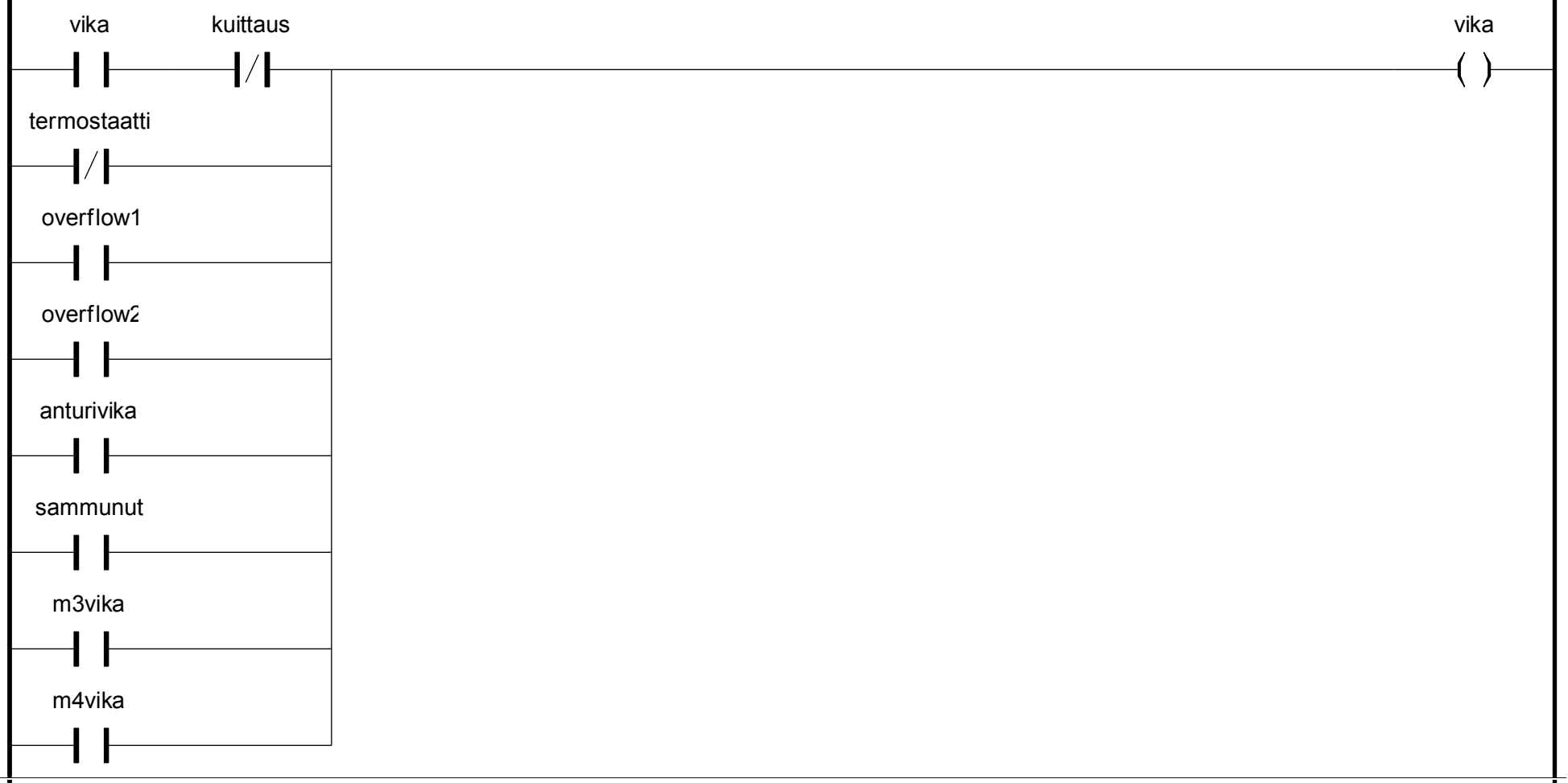
```
0001  (* termostaatin lämpötilojen muunnos asetuksista *)
0002
0003  apumuuttujat[2]:=asetukset[0]*260; (* alempi *)
0004  apumuuttujat[1]:=asetukset[1]*260; (* ylempi *)
0005
0006
0007  (* veden lämpötila celsiuksiksi *)
0008
0009  apumuuttujat[0]:=vesi/260;
0010
0011
0012  (* savun lämpötila celsiuksiksi
0013  10V 0-32767 10mV/C
0014  32,76/C *)
0015
0016  apumuuttujat[3]:=savu/33;
0017
0018
0019  (* ruuvin pid säätimen virhe apumuuttujaan*)
0020
0021  apumuuttujat[5]:=REAL_TO_INT(ohjaus.pid2.ERROR);
0022
0023
0024  (* I säädin*)
0025
0026  apumuuttujat[6]:=REAL_TO_INT(ohjaus.pid2.I.OUT);
0027
0028
0029  (* Ramp1 suodatus muuttujat*)
0030
0031  aputime[0]:=INT_TO_TIME(asetukset[19]);
0032
0033
0034
0035  (* muunnetaan ohjaussanat tavuiksi *)
0036
0037  SANA:=dataread2[1];
0038  tilasanat[2]:=HI;
0039
0040  SANA:=dataread3[1];
0041  tilasanat[3]:=HI;
0042
0043  SANA:=dataread4[1];
0044  tilasanat[4]:=HI;
0045
0046  (* otetaan neliöjuuri hapen arvosta mittaria varten *)
0047
0048  apumuuttujat[4]:=TRUNC(SQRT(happi));
0049
0050
```

ohjaus (PRG-LD)

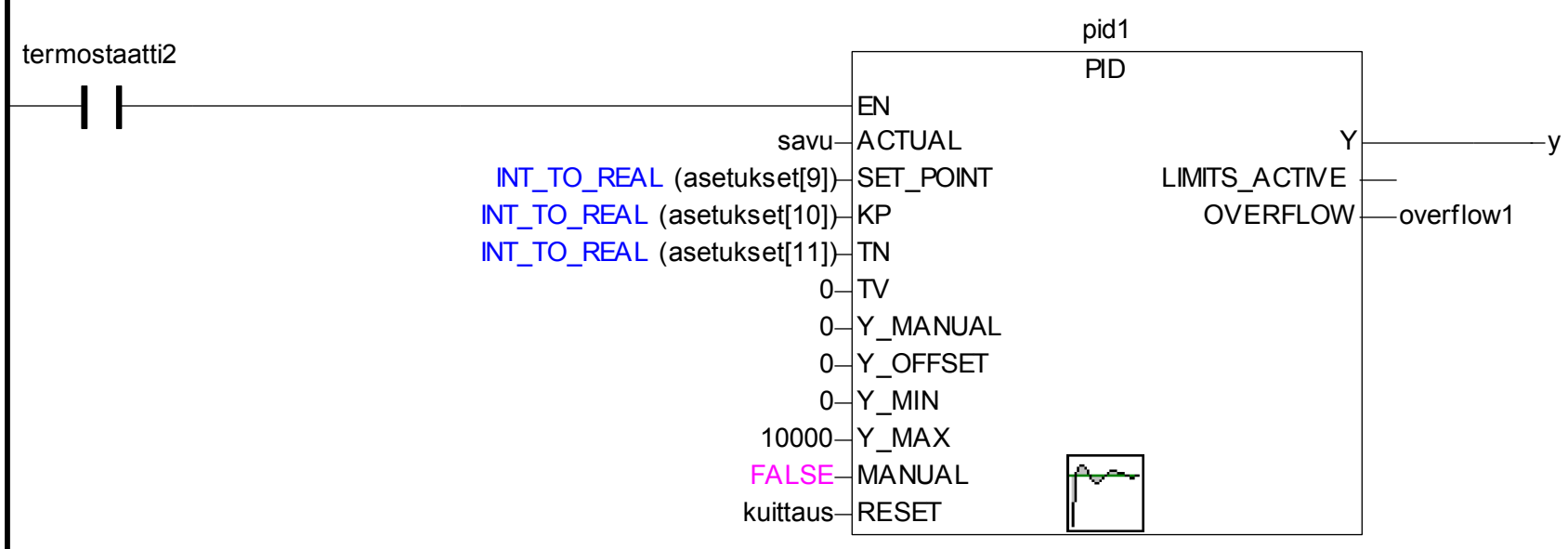
```
0001  PROGRAM ohjaus
0002  VAR
0003
0004
0005      pid1: PID;
0006      overflow1: BOOL;
```

```
0007      pid2: uusipid;
0008      overflow2: BOOL;
0009      y:REAL;
0010      z:REAL;
0011
0012
0013  (*  pid1manual: BOOL;*)
0014
0015
0016
0017      q: BOOL;
0018      rs1: RS;
0019      limit1: LIMITALARM;
0020      ton2: TON;
0021
0022      o: BOOL;
0023      u: BOOL;
0024      il: BOOL;
0025
0026
0027
0028  (*  suodatettu: INT;*)
0029
0030  END_VAR
```

0001 Viat

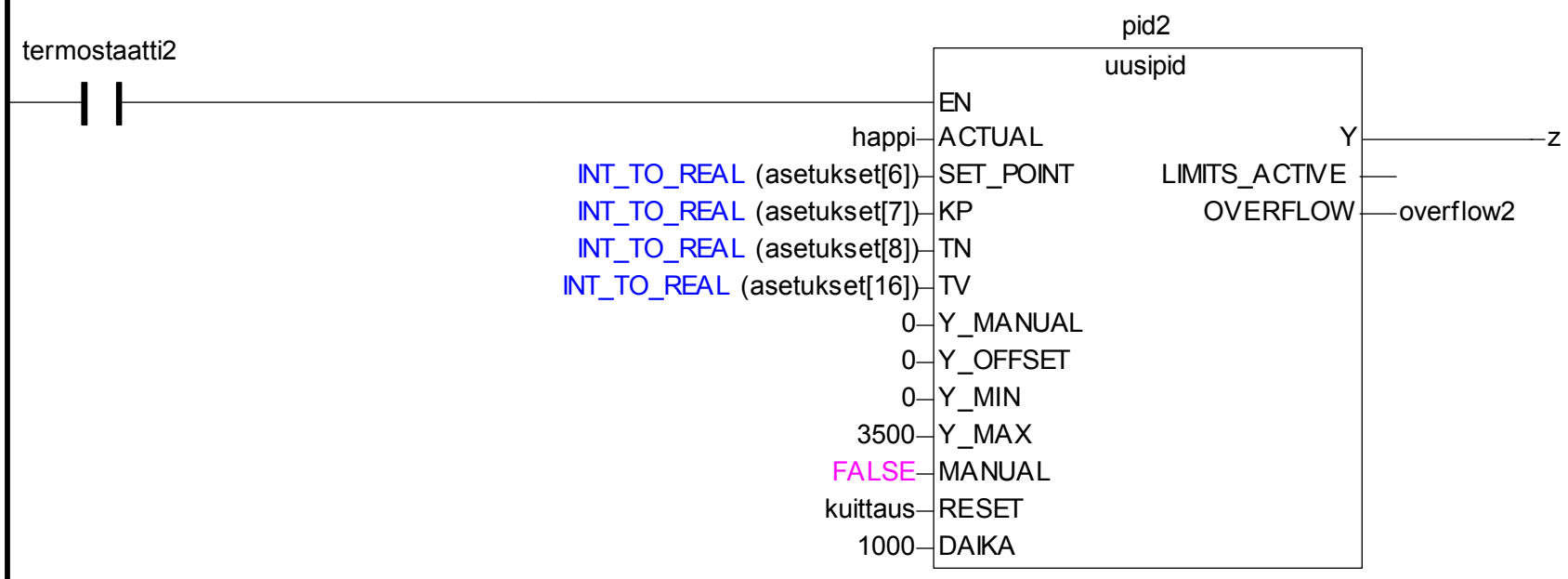


0002 Puhallinpid



0003

Ruuvinpid



0004

Ohjearvot_muuttujiin

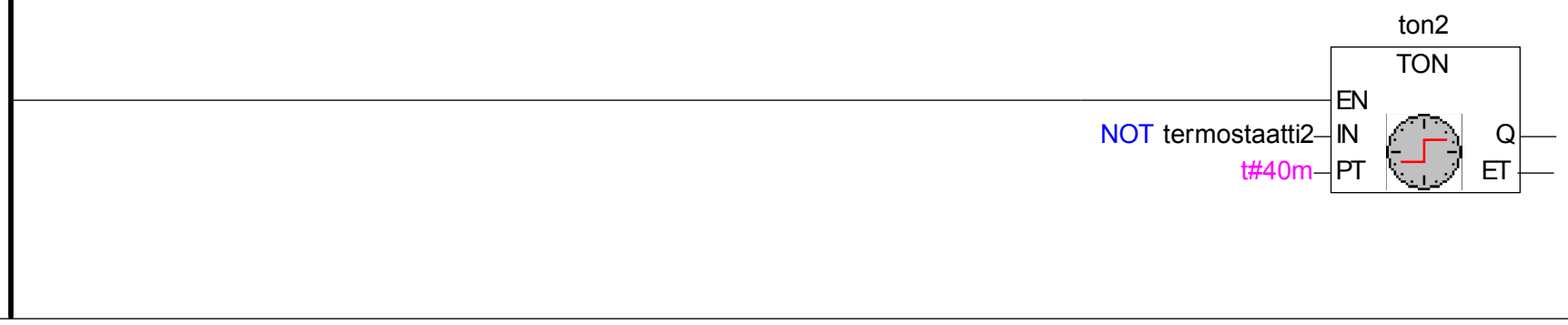
0005

0006

KaytetaanRSkiikkua

0007

0008



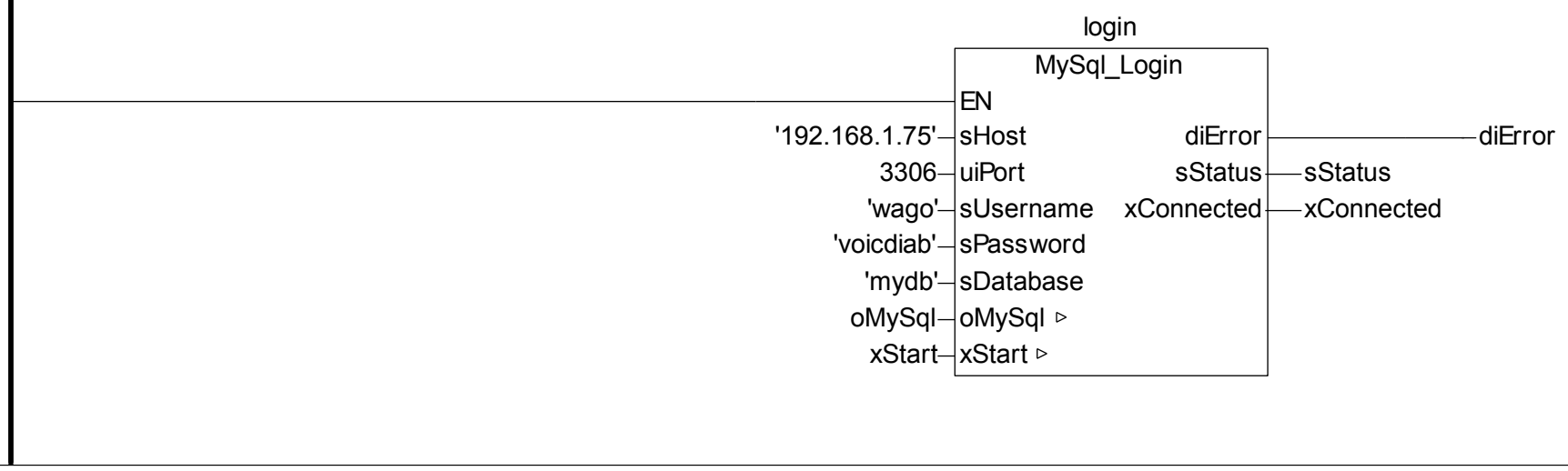
PLC_PRG (PRG-LD)

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 END_VAR
```

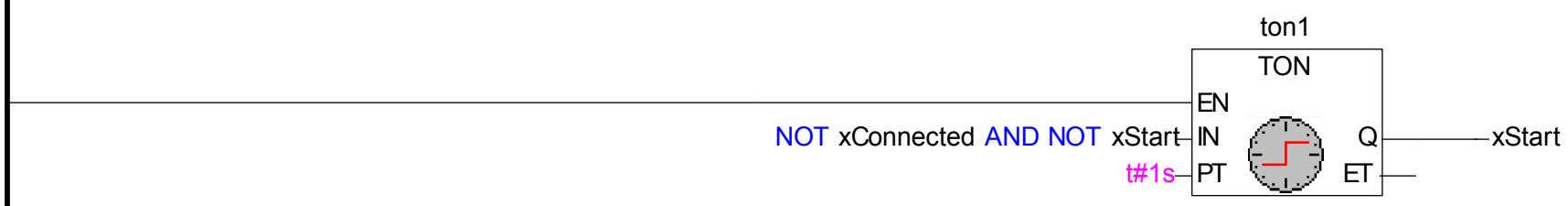
sql (PRG-LD)

```
0001 PROGRAM sql
0002 VAR
0003
0004
0005
0006
0007
0008
0009
0010     login: MySql_Login;
0011
0012     xConnected: BOOL;
0013     sStatus: STRING;
0014     diError: DINT;
0015     oMySql: MySql_Context;
0016     xStart: BOOL;
0017     ton1: TON;
0018     execute: MySql_Execute;
0019
0020     asSqlCommand: ARRAY [0..gcMySql_iSqlUpperBound] OF STRING(gcMySql_iSqlLength);
0021     startexecute: BOOL;
0022     sStatusexecute: STRING;
0023     ton2: TON;
0024     executedierror: DINT;
0025
0026 END_VAR
0027 (*VAR
0028     ExpectedResponse  :  INT;
0029     Timer             :  TON;
0030     TimeOutPointer    :  INT;
0031     CALC_CRC          :  CRC16;
0032     i                 :  INT;
0033
0034     TriggerTimeOut    :  BOOL;
0035     Reset             :  BOOL := TRUE;
0036 END_VAR *)
0037
```

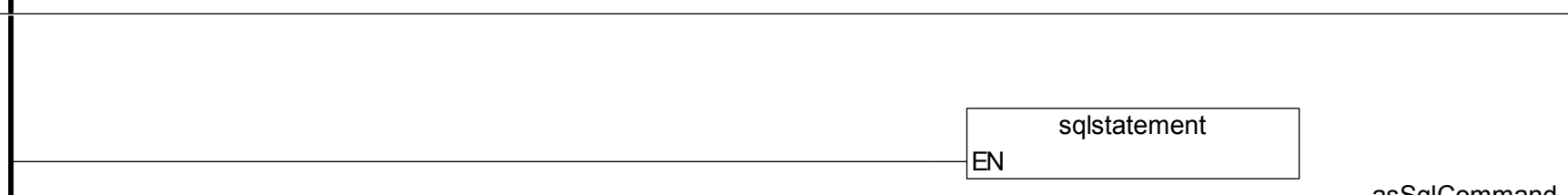
0001



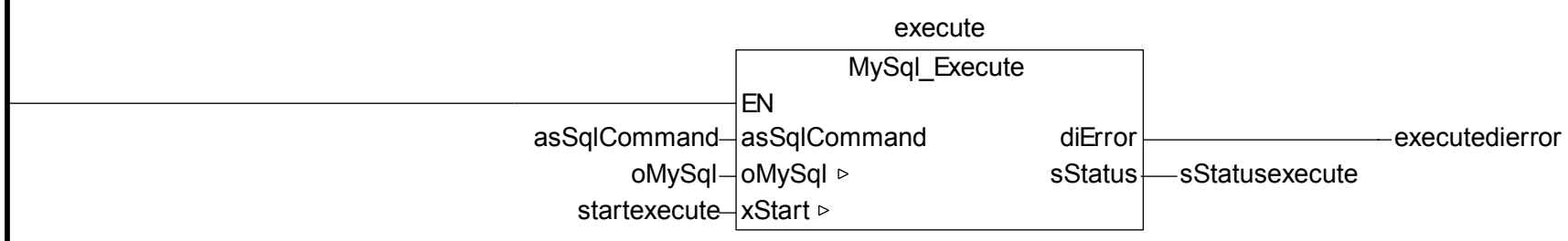
0002



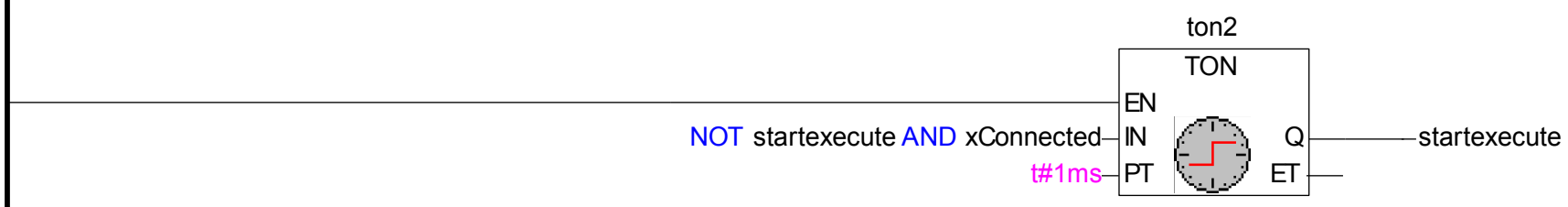
0003



0004



0005



sqlstatement (FUN-ST)

```
0001  FUNCTION sqlstatement :ARRAY [0..gcMySql_iSqlUpperBound] OF STRING(gcMySql_iSqlLength)
0002  VAR_INPUT
0003  END_VAR
```

```
0001  (* Prepare sql-INSERT-Statement *)
0002  sqlstatement[0] := 'INSERT INTO kattila ';
0003  sqlstatement[1] := '(happi, savu, vesi, ruuviohje, puhallinohje, ruuvipidvirhe, ruuvipidint, suppilo, termostaatti, hydrpumppu, suodatettu) ';
0004  sqlstatement[2] := 'VALUES (';
0005  sqlstatement[3] := REAL_TO_STRING(happi);
0006  sqlstatement[4] := ',';
0007  sqlstatement[5] := INT_TO_STRING(apumuuttujat[3]);
0008  sqlstatement[6] := ',';
0009  sqlstatement[7] := INT_TO_STRING(apumuuttujat[0]);
0010  sqlstatement[8] := ',';
0011  sqlstatement[9] := WORD_TO_STRING(datawrite3[2]);
0012  sqlstatement[10] := ',';
0013  sqlstatement[11] := WORD_TO_STRING(datawrite4[2]);
0014  sqlstatement[12] := ',';
0015  sqlstatement[13] := INT_TO_STRING(apumuuttujat[5]);
0016  sqlstatement[14] := ',';
0017  sqlstatement[15] := INT_TO_STRING(apumuuttujat[6]);
0018  sqlstatement[16] := ',';
0019  sqlstatement[17] := BOOL_TO_STRING(suppilo);
0020  sqlstatement[18] := ',';
0021  sqlstatement[19] := BOOL_TO_STRING(termostaatti2);
0022  sqlstatement[20] := ',';
0023  sqlstatement[21] := BOOL_TO_STRING(pumppu);
0024  sqlstatement[22] := ',';
0025  sqlstatement[23] := WORD_TO_STRING(dataread3[7]);
0026  sqlstatement[24] := ')';
0027  sqlstatement[25] := ''; (* End of SQL-Statement *)
```

uusipid (FB-ST)

```
0001  FUNCTION_BLOCK uusipid
0002
0003  VAR_INPUT
0004      ACTUAL :REAL;          (* actual value, process variable *)
0005      SET_POINT:REAL;  (* desired value, set point *)
0006      KP:REAL;               (* proportionality const. (P)*)
0007      TN:REAL;               (* reset time (I) in sec *)
0008      TV:REAL;               (* rate time, derivative time (D) in sec*)
0009      Y_MANUAL:REAL;         (* Y is set to this value as long as MANUAL=TRUE *)
0010      Y_OFFSET:REAL;         (* offset for manipulated variable *)
0011      Y_MIN:REAL;            (* minimum value for manipulated variable *)
0012      Y_MAX:REAL;            (* maximum value for manipulated variable *)
0013      MANUAL:BOOL;           (*    TRUE: manual: Y is not influenced by controller,
0014                                   FALSE: controller determines Y *)
0015      RESET:BOOL;            (* reset: set Y output to Y_OFFSET and reset integral part *)
0016      DAIKA:DWORD;
0017  END_VAR
0018
0019  VAR_OUTPUT
0020      Y:REAL;                (* manipulated variable, set value*)
0021      LIMITS_ACTIVE:BOOL:=FALSE; (* true set value would exceed limits Y_MIN, Y_MAX *)
0022      OVERFLOW:BOOL:=FALSE;      (* overflow in integral part *)
0023  END_VAR
0024
0025  VAR
0026      CLOCK:TON;
0027      I: INTEGRAL;
0028      D: derivaattori;
0029      TMDIFF: DWORD;
0030      ERROR: REAL;
0031      INIT: BOOL:=TRUE;
0032      Y_ADDOFFSET: REAL;
0033  END_VAR
```

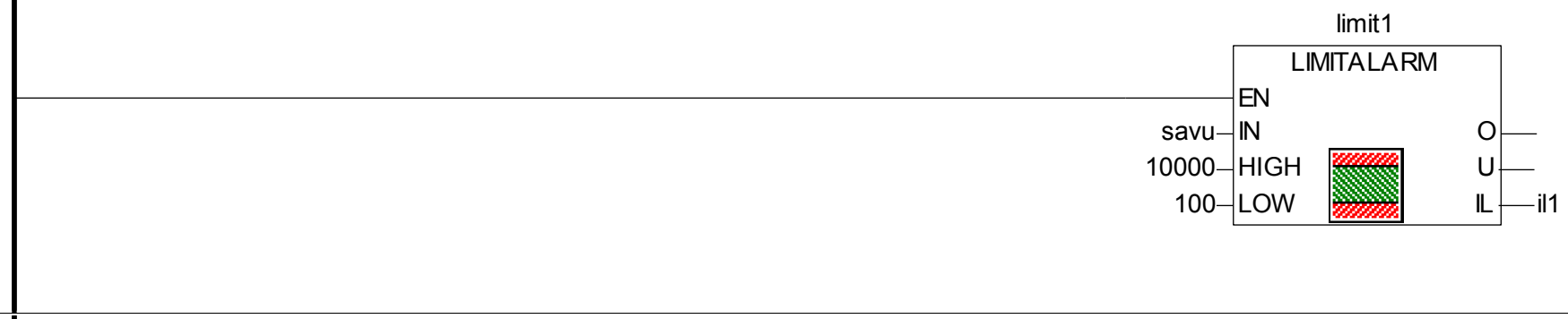
```
0001  IF TN>0 AND KP<> 0 AND (NOT OVERFLOW OR RESET OR MANUAL) THEN
0002      ERROR := SET_POINT-ACTUAL;                              (* Regeldifferenz *)
0003
0004      IF RESET OR MANUAL OR INIT THEN                         (* Reset oder Handbetrieb *)
0005          I(RESET:=TRUE);
0006          D(RESET:=TRUE);
0007          OVERFLOW:=FALSE;
0008          LIMITS_ACTIVE:=FALSE;
0009          IF RESET OR INIT THEN
0010              Y := Y_OFFSET;
0011              INIT:=FALSE;
0012              Y_ADDOFFSET := 0;
0013          ELSE
0014              Y := Y_MANUAL;
0015              Y_ADDOFFSET := Y_MANUAL-(Y_OFFSET+KP*(ERROR+I.OUT/TN+D.OUT*TV));
0016          END_IF
0017          TMDIFF:=0;
0018          CLOCK(IN:=FALSE);                                   (* Timer neu starten *)
0019          CLOCK(PT:=t#1h, IN:=TRUE);
0020      ELSE
0021          CLOCK;                                              (* Timer abfragen *)
0022  TMDIFF:=TIME_TO_DWORD(CLOCK.ET);                        (* Zeitdifferenz seit letztem Aufruf *)
0023      END_IF;
0024
0025      IF TMDIFF>0 THEN
0026          CLOCK(IN:=FALSE);                                   (* Timer neu starten *)
0027          CLOCK(PT:=t#1h, IN:=TRUE);
0028
0029  (*        D(IN:=ERROR, TM:=TMDIFF, RESET:=FALSE);                 (* Differential abschätzen *) *)
0030  D(IN:=ACTUAL, TM:=DAIKA, RESET:=FALSE);
0031          I(IN:=ERROR, TM:=TMDIFF, RESET:=FALSE);                 (* Integral abschätzen *)
0032
0033          OVERFLOW := I.OVERFLOW;
0034          IF NOT OVERFLOW THEN
0035              Y:=Y_OFFSET+KP*(ERROR+I.OUT/TN-D.OUT*TV) + Y_ADDOFFSET;
0036              IF Y>1E30 OR Y<-1E30 THEN                       (* Overflow steht bevor, darf aber eigentlich nicht passieren *)
0037                  OVERFLOW:=TRUE;
0038              END_IF;
0039
0040              LIMITS_ACTIVE:=FALSE;
```

```
0041            IF Y_MAX>Y_MIN AND Y>Y_MAX  THEN                 (* Stellwert-Obergrenze überschritten *)
0042                Y:=Y_MAX;
0043                LIMITS_ACTIVE:=TRUE;
0044                I(IN:=-ERROR,TM:=TMDIFF,RESET:=FALSE);       (* Integral korrigieren *)
0045            END_IF;
0046
0047            IF Y_MAX>Y_MIN AND Y<Y_MIN THEN                  (* Stellwert-Untergrenze unterschritten *)
0048                Y:=Y_MIN;
0049                LIMITS_ACTIVE:=TRUE;
0050                I(IN:=-ERROR,TM:=TMDIFF,RESET:=FALSE);       (* Integral korrigieren *)
0051            END_IF;
0052        END_IF;
0053        CLOCK(PT:=t#1h,IN:=TRUE);
0054    END_IF;
0055
0056 END_IF;
```

valvonta (PRG-LD)

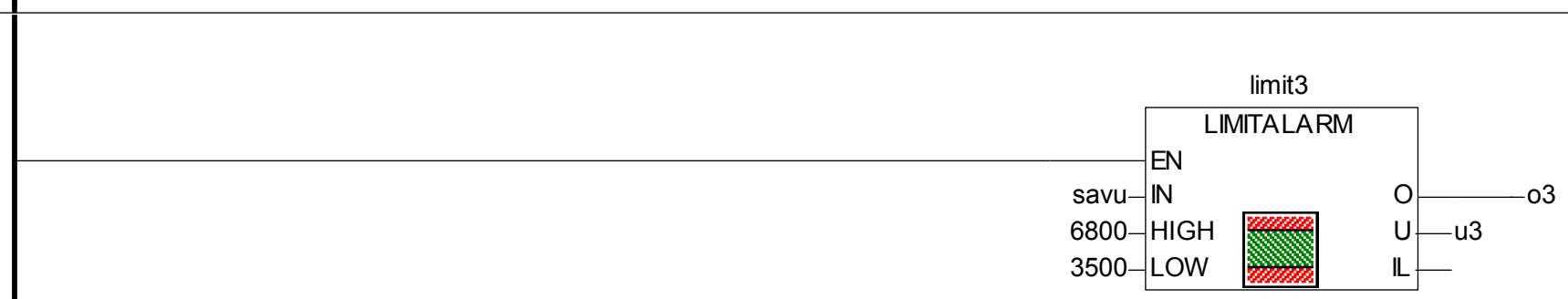
```
0001 PROGRAM valvonta
0002 VAR
0003
0004     limit1: LIMITALARM; (* O=OBER=YLEMPI U=UNTERE=ALEMPI*)
0005     limit2: LIMITALARM;
0006
0007     limit3: LIMITALARM;
0008     il2: BOOL;
0009     il1: BOOL;
0010     ton1: TON;
0011     ton2: TON;
0012     ton3: TON;
0013     u3: BOOL;
0014     ton4: TON;
0015
0016     o3: BOOL;
0017 END_VAR
0018
```

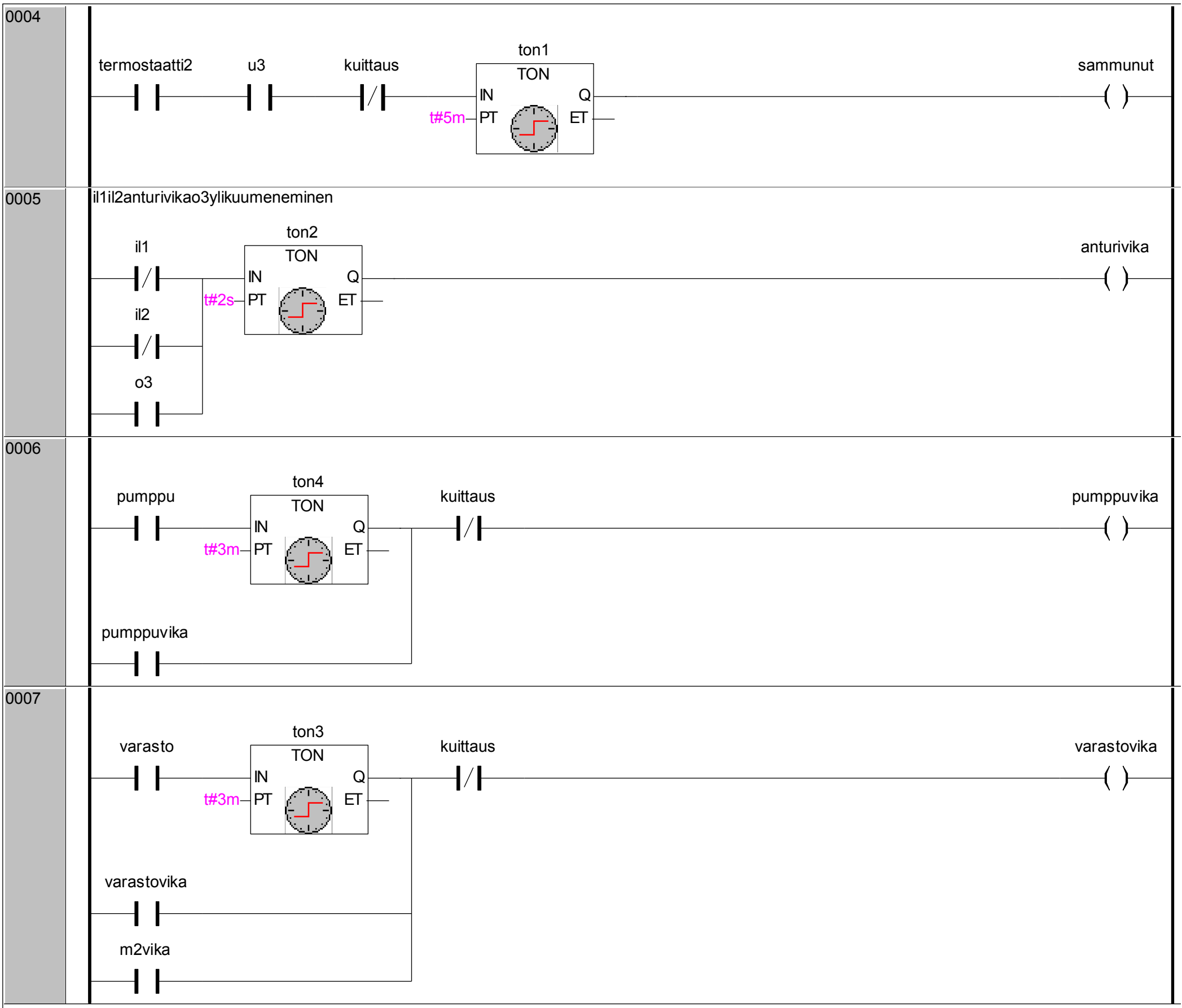
0001



0002

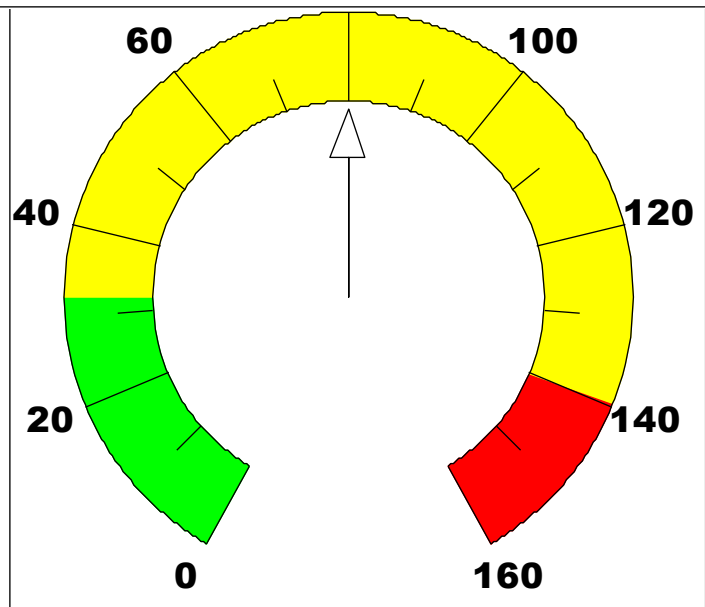
0003

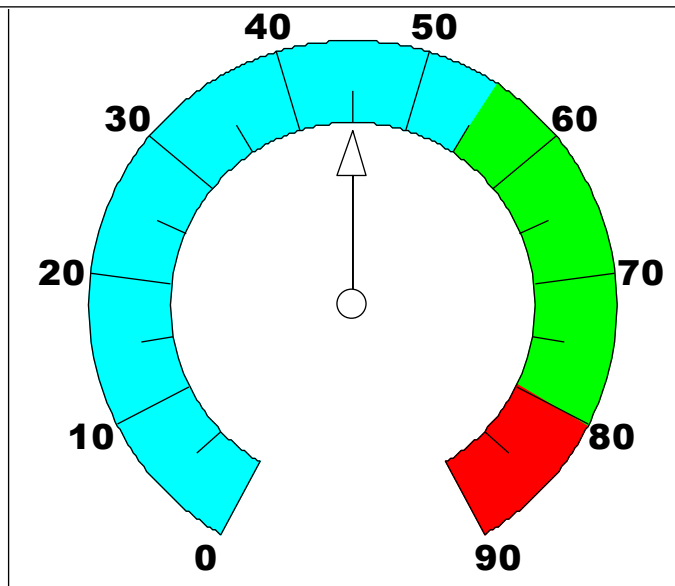


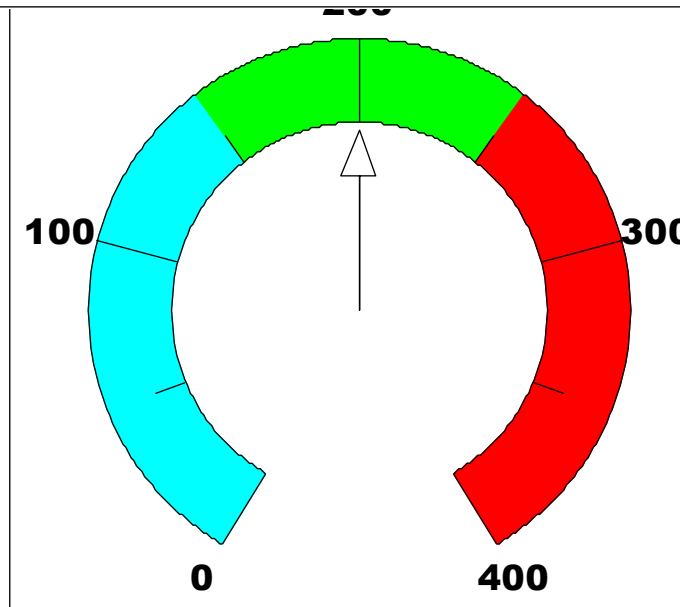


ASETUKSET

Termostaatti	**%s**	**-**	**%s**
Tauko (Min)	**%s**		
Käynti (Sek)	**%s**		
Ruuvi min (Hz)	**%s**		
Ruuvi max (Hz)	**%s**		
Hapen asetus ruuvi	**%s**		
Hapen asetus puhallin	**%s**		
Puhallus min. (Hz)	**%s**		
Puhallus max. (Hz)	**%s**		
Liekkivahti (C)	**%s**		
Liekkivahti aika (Sek)	**%s**		







HAPPI	VESI	SAVU

VIKA/KUITTAUS	KÄY/SEIS

HYDRAULIPUMPPU
VARASTORUUVI
RUUVI
PUHALLIN

TERMOSTAATTI

SUPPILO			
L1	L2	L3	L4

ASETUKSET ->

Global_Variables

```
0001  VAR_GLOBAL
0002  (*    bCOM_PORT: BYTE := 0;
0003        cbCOM_BAUDRATE: COM_BAUDRATE := BAUDRATE_TERMINAL_DEFAULT;
0004        cpCOM_PARITY: COM_PARITY := PARITY_TERMINAL_DEFAULT;
0005        csCOM_STOPBITS: COM_STOPBITS := STOPBITS_TERMINAL_DEFAULT;
0006        cbsCOM_BYTESIZE: COM_BYTESIZE := BYTESIZE_TERMINAL_DEFAULT;
0007        cfCOM_FLOW_CONTROL: COM_FLOW_CONTROL := HALFDUPLEX; (* FLOW_CONTROL_TERMINAL_DEFAULT ; *)
0008        TimeOut: TIME := t#1000ms; *)
0009        StartFunction: BOOL := FALSE;
0010        Query: typModbusQuery;
0011        Response: typModbusResponse;
0012        MB_Error: enumMB_ERROR := MB_NO_ERROR;
0013
0014        dataread1: ARRAY [0..124] OF WORD; (* luettu modbus data *)
0015        dataread2: ARRAY [0..124] OF WORD;
0016        dataread3: ARRAY [0..124] OF WORD;
0017        dataread4: ARRAY [0..124] OF WORD;
0018        datawrite1: ARRAY[0..124] OF WORD; (* kirjoitettava modbus data *)
0019        datawrite2: ARRAY[0..124] OF WORD;
0020        datawrite3: ARRAY[0..124] OF WORD;
0021        datawrite4: ARRAY[0..124] OF WORD;
0022
0023        varasto: BOOL;
0024        ruuvi: BOOL;
0025        termostaatti2: BOOL;
0026        puhallin: BOOL;
0027
0028        apumuuttujat: ARRAY[0..20] OF INT;
0029        aputime:      ARRAY[0..5] OF TIME;
0030
0031        kuittaus: BOOL;
```

```
0032        sammunut: BOOL;
0033        anturivika: BOOL;
0034        varastovika: BOOL;
0035
0036
0037
0038        m2valmis: BOOL;
0039        m2run: BOOL;
0040        m2suunta: BOOL;
0041        m2hal: BOOL;
0042        m2ramppi: BOOL;
0043        m2ohjauspaikka: BOOL;
0044        m2vika: BOOL;
0045        m2nollanopeus: BOOL;
0046
0047        m3valmis: BOOL;
0048        m3run: BOOL;
0049        m3suunta: BOOL;
0050        m3hal: BOOL;
0051        m3ramppi: BOOL;
0052        m3ohjauspaikka: BOOL;
0053        m3vika: BOOL;
0054        m3nollanopeus: BOOL;
0055
0056        m4valmis: BOOL;
0057        m4run: BOOL;
0058        m4suunta: BOOL;
0059        m4hal: BOOL;
0060        m4ramppi: BOOL;
0061        m4ohjauspaikka: BOOL;
0062        m4vika: BOOL;
0063        m4nollanopeus: BOOL;
0064        tilasanat: ARRAY[1..4] OF BYTE;
0065
0066
0067        pumppuvika: BOOL;
0068    (*  UpperBound: INT := 20;*)
0069
0070    END_VAR
0071
0072    VAR_GLOBAL CONSTANT
0073        gcMySql_iSqlUpperBound: INT := 50;
0074        gcMySql_iSqlLength: INT := 120;
0075    END_VAR
0076
0077    VAR_GLOBAL RETAIN PERSISTENT
0078        asetukset: ARRAY [0..20] OF INT;
0079        vika: BOOL;
0080        run: BOOL;
0081    END_VAR
```

Variable_Configuration

```
0001    VAR_CONFIG
0002    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Global_Constants

```
0001    VAR_GLOBAL CONSTANT
0002        MB_MAX_BUFFER_SIZE          :   INT := 512;             (* internal buffer size *)
0003
0004        MB_SIZE_SLAVEBUFFER         :   INT := 255;             (* size of the slavebuffer *)
0005
0006        MB_START_ADDRESS_RANGE:     WORD := 16#0000;  (* this constant represents the start of the
0007                                                                  available modbus address range *)
```

```
0008
0009        MB_MAX_JOBS                 :     INT := 5;
0010    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Globale_InterfaceConstant

```
0001    VAR_GLOBAL CONSTANT
0002        RING_BUFFER_SIZE        :     INT := 255; (* Default-Size of the RingBuffer *)
0003    END_VAR
```

CAM Data

```
0001    (* Automatic generated CAM-Data *)
0002    VAR_GLOBAL
0003    END_VAR
```

CNC Data

```
0001    (* Automatic generated CNC-Data *)
0002    VAR_GLOBAL
0003    END_VAR
```

Drive Configuration Data

```
0001    (* Automatic generated Drive-Data *)
0002    VAR_GLOBAL
0003    END_VAR
```

Global Variables 0

```
0001    VAR_GLOBAL
0002    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Version

```
0001    VAR_GLOBAL CONSTANT
0002        rVersion:REAL:=1.001;
0003    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL CONSTANT
0002        SOCKET_INVALID:DINT:=-1;
0003
0004        (* AddressFamily *)
0005        SOCKET_AF_UNSPEC:INT:=     0;              (* unspecified *)
0006        SOCKET_AF_LOCAL:INT:=  1;                  (* local to host (pipes, portals) *)
0007        SOCKET_AF_UNIX:INT:=SOCKET_AF_LOCAL;       (* backward compatibility *)
0008        SOCKET_AF_INET:INT:=2;                     (* internetwork: UDP, TCP, etc. *)
0009        SOCKET_AF_IMPLINK:INT:=3;                  (* arpanet imp addresses *)
0010        SOCKET_AF_PUP:INT:=4;                      (* pup protocols: e.g. BSP *)
0011        SOCKET_AF_CHAOS:INT:=5;                    (* mit CHAOS protocols *)
0012        SOCKET_AF_NS:INT:=6;                       (* XEROX NS protocols *)
0013        SOCKET_AF_ISO:INT:=7;                      (* ISO protocols *)
0014        SOCKET_AF_OSI:INT:=SOCKET_AF_ISO;
0015        SOCKET_AF_ECMA:INT:=8;                 (* european computer manufacturers *)
0016        SOCKET_AF_DATAKIT:INT:=9;                  (* datakit protocols *)
0017        SOCKET_AF_CCITT:INT:=10;                   (* CCITT protocols, X.25 etc *)
0018        SOCKET_AF_SNA:INT:=11;                     (* IBM SNA *)
0019        SOCKET_AF_DECnet:INT:=12;                  (* DECnet *)
0020        SOCKET_AF_DLI:INT:=13;                     (* DEC Direct data link interface *)
0021        SOCKET_AF_LAT:INT:= 14;                    (* LAT *)
```

```
0022    SOCKET_AF_HYLINK:INT:=15;               (* NSC Hyperchannel *)
0023    SOCKET_AF_APPLETALK:INT:=16;        (* Apple Talk *)
0024    SOCKET_AF_ROUTE:INT:=17;                (* Internal Routing Protocol *)
0025    SOCKET_AF_LINK:INT:=18;             (* Link layer interface *)
0026    SOCKET_pseudo_AF_XTP:INT:=19;       (* eXpress Transfer Protocol (no AF) *)
0027    SOCKET_AF_COIP:INT:=20;                 (* connection-oriented IP, aka ST II *)
0028    SOCKET_AF_CNT:INT:=21;              (* Computer Network Technology *)
0029    SOCKET_pseudo_AF_RTIP:INT:=22;          (* Help Identify RTIP packets *)
0030    SOCKET_AF_IPX:INT:=23;                  (* Novell Internet Protocol *)
0031    SOCKET_AF_SIP:INT:=24;                  (* Simple Internet Protocol *)
0032    SOCKET_pseudo_AF_PIP:INT:=25;       (* Help Identify PIP packets *)
0033    SOCKET_AF_MAX:INT:=26;
0034    SOCKET_AF_INET_BSD:INT:=100;        (* BSD-specific INET af *)
0035    SOCKET_AF_INET_STREAMS:INT:=101;    (* STREAMS-specific INET af *)
0036
0037    (* Level number for (get/set)sockopt() to apply to socket itself. *)
0038    SOCKET_SOL:WORD:=16#ffff;
0039
0040    (* Socket options *)
0041    SOCKET_SO_DEBUG:DINT:=16#0001;              (* turn on debugging info recording *)
0042    SOCKET_SO_ACCEPTCONN:DINT:=16#0002;   (* socket has had listen() *)
0043    SOCKET_SO_REUSEADDR:DINT:=16#0004;      (* allow local address reuse *)
0044    SOCKET_SO_KEEPALIVE:DINT:=16#0008;          (* keep connections alive *)
0045    SOCKET_SO_DONTROUTE:DINT:=16#0010;      (* just use interface addresses *)
0046    SOCKET_SO_BROADCAST:DINT:=16#0020; (* permit sending of broadcast msgs *)
0047    SOCKET_SO_USELOOPBACK:DINT:=16#0040;  (* bypass hardware when possible *)
0048    SOCKET_SO_LINGER:DINT:=16#0080;             (* linger on close if data present *)
0049    SOCKET_SO_OOBINLINE:DINT:=16#0100;          (* leave received OOB data in line *)
0050    SOCKET_SO_REUSEPORT:DINT:=16#0200; (* allow local address & port reuse *)
0051    SOCKET_SO_SNDBUF:DINT:=16#1001;         (* send buffer size *)
0052    SOCKET_SO_RCVBUF:DINT:=  16#1002;           (* receive buffer size *)
0053    SOCKET_SO_SNDLOWAT:DINT:=16#1003;   (* send low-water mark *)
0054    SOCKET_SO_RCVLOWAT:DINT:=16#1004;   (* receive low-water mark *)
0055    SOCKET_SO_SNDTIMEO:DINT:=16#1005;   (* send timeout *)
0056    SOCKET_SO_RCVTIMEO:DINT:=16#1006;   (* receive timeout *)
0057    SOCKET_SO_ERROR:DINT:=16#1007;          (* get error status and clear *)
0058    SOCKET_SO_TYPE:DINT:=16#1008;               (* get socket type *)
0059    SOCKET_SO_PROTOTYPE:DINT:=16#1009; (* get/set protocol type *)
0060    (* TCPIP socket options *)
0061    SOCKET_TCP_NODELAY:DINT:=16#01; (* don't delay send to coalesce packets *)
0062    SOCKET_TCP_MAXSEG:DINT:=16#02;  (* set maximum segment size *)
0063
0064    (* Socket types *)
0065    SOCKET_STREAM:DINT:=1;              (* stream socket *)
0066    SOCKET_DGRAM:DINT:=2;           (* datagram socket *)
0067    SOCKET_RAW:DINT:=3;                 (* raw-protocol interface *)
0068    SOCKET_RDM:DINT:=4;                 (* reliably-delivered message *)
0069    SOCKET_SEQPACKET:DINT:=5;       (* sequenced packet stream *)
0070
0071    (* Inet address definitions *)
0072    SOCKET_INADDR_ANY:UDINT:=16#00000000;
0073    SOCKET_INADDR_LOOPBACK:UDINT:=16#7f000001;
0074    SOCKET_INADDR_BROADCAST:UDINT:=16#ffffffff;
0075    SOCKET_INADDR_NONE:UDINT:=16#ffffffff;
0076
0077    (* Protocols *)
0078    SOCKET_IPPROTO_IP:DINT:=0;              (* dummy for IP *)
0079    SOCKET_IPPROTO_ICMP:DINT:=1;        (* control message protocol *)
0080    SOCKET_IPPROTO_IGMP:DINT:=2;        (* group management protocol *)
0081    SOCKET_IPPROTO_GGP:DINT:=3;         (* gateway^2 (deprecated) *)
0082    SOCKET_IPPROTO_TCP:DINT:=6;         (* tcp *)
0083    SOCKET_IPPROTO_PUP:DINT:=12;        (* pup *)
0084    SOCKET_IPPROTO_UDP:DINT:=17;        (* user datagram protocol *)
0085    SOCKET_IPPROTO_IDP:DINT:=22;        (* xns idp *)
0086    SOCKET_IPPROTO_ND:DINT:=77;         (* UNOFFICIAL net disk proto *)
0087    SOCKET_IPPROTO_RAW:DINT:=255;   (* raw IP packet *)
0088    SOCKET_IPPROTO_MAX:DINT:=256;
0089
0090    (* Flags *)
0091    SOCKET_MSG_OOB:DINT:=16#1;              (* process out-of-band data *)
0092    SOCKET_MSG_PEEK:DINT:=16#2;             (* peek at incoming message *)
0093    SOCKET_MSG_DONTROUTE:DINT:=16#4;  (* send without using routing tables *)
0094
```

```
0095        (* Ioctl commands *)
0096        SOCKET_FIONREAD:DINT:=1;                (* get num chars available to read *)
0097        SOCKET_FIONBIO:DINT:=2;                     (* set to non-blocking *)
0098
0099        (* For SysSockSelect() descriptors *)
0100        SOCKET_FD_SETSIZE :DINT :=      64;
0101        MAX_SOCKET_FD_SETSIZE : DINT := 63;
0102    END_VAR
```

Globale_Variablen

```
0001    VAR_GLOBAL
0002    END_VAR
```

Global_Variables

```
0001    VAR_GLOBAL CONSTANT
0002        (* Global constants could be modified by hiding with a global constant of the same name in your project *)
0003
0004        gcMySql_dwMaxColumnCount: DWORD := 15; (* Defines the maximum number of columns in a result set
0005                    this library can process.
0006        *)
0007
0008        gcMySql_dwMaxRowCount : DWORD := 40; (* Defines the maximum number of rows in a result set
0009                    this library can process.
0010        *)
0011
0012        gcMySql_dwMaxRowSize : DWORD := 500; (* Defines the maximum length (size in bytes) of one row in a result set
0013                    this library can process.
0014        *)
0015
0016        gcMySql_xClearResultSetBeforeUse : BOOL := FALSE; (* Clear the "stResultSet" before use(need approximatly 400ms),
0017                    used by function block "MySql_Query".
0018        *)
0019
0020        gcMySql_tResponseCompliedlyReceived : TIME := t#50ms;       (* Wait time - before processing received response data.
0021                    Used to be shure complied response are received. The Wait time is started with first response package
0022                    and restarted on every received response package. Helpful when dialing with slow servers.
0023                    Used by function block "MySql_Query".
0024        *)
0025
0026        gcMySql_dwMaxIdentifierLength : DWORD := 80; (* Defines the maximum length (size in bytes) of a column name,
0027                    used by structure "MySql_FieldInfo", this library can process.
0028        *)
0029
0030        gcMySql_iSqlUpperBound: INT := 10; (* Defines the upper bound of "asSqlStatement",
0031                    used by function blocks "MySql_ExecSql" and "MySql_Query"
0032                    to provide an SQL statement as "ARRAY [0..UpperBound] OF STRING(Size)".
0033        *)
0034
0035        gcMySql_iSqlLength: INT := 100; (* Defines the size in byte of an array element
0036                    of "asSqlStatement", used by function blocks "MySql_ExecSql" and "MySql_Query"
0037                    to provide an SQL statement as "ARRAY [0..UpperBound] OF STRING(Size)".
0038        *)
0039
0040        gcMySql_tTimeOut : TIME := t#10s;     (* WatchDog time -
0041                    Defines the maximum time a function block is operating in the same stage without progress.
0042        *)
0043
0044        gcMySql_dwTxBufferSize: DWORD := 8000; (* Defines the transmit buffer size
0045        *)
0046
0047        gcMySql_dwRxBufferSize: DWORD := 40000; (* Defines the receive buffer size
0048        *)
0049
0050        gcMySql_dwMaxPacketSize: DWORD := 1024; (* The maximum number of bytes in a packet for the client.
0051        *)
0052
0053        gcMySql_bCharsetNumber : BYTE := 16#08; (*
0054                    +------+-------------------+-------------------+
0055                    | ID   | CHARACTER_SET_NAME | COLLATION_NAME    |
0056                    +------+-------------------+-------------------+
0057                    | 16#08| latin1            | latin1_swedish_ci |
0058                    +------+-------------------+-------------------+
```

```
                To operate in the same domain as the server, check
                "pstMySql^.stServerInfo.bServerLanguage", that the
                server passes in the Handshake Initialization packet
    *)

    gcMySql_wClientFlags : WORD := 16#8601; (* CLIENT_xxx options.

                The list of flags taken from ethereal sniff with server version "5.0.67-community-nt" ):

                    CLIENT_LONG_PASSWORD        .... .... .... ...1      | ON  | new more secure passwords */
                    CLIENT_FOUND_ROWS           .... .... .... ..0.      | OFF | Found instead of affected rows */
                    CLIENT_LONG_FLAG            .... .... .... .0..      | OFF | Get all column flags */
                    CLIENT_CONNECT_WITH_DB      .... .... .... 0...      | OFF | One can specify db on connect */
                    CLIENT_NO_SCHEMA            .... .... ...0 ....      | OFF | Don't allow database.table.column */
                    CLIENT_COMPRESS             .... .... ..0. ....      | OFF | Can use compression protocol */
                    CLIENT_ODBC                 .... .... .0.. ....      | OFF | Odbc client */
                    CLIENT_LOCAL_FILES          .... .... 0... .... | OFF | Can use LOAD DATA LOCAL */
                    CLIENT_IGNORE_SPACE         .... ...0 .... .... | OFF | Ignore spaces before '(' */
                    CLIENT_PROTOCOL_41          .... ..1. .... .... | ON  | New 4.1 protocol */
                    CLIENT_INTERACTIVE          .... .1.. .... .... | ON  | This is an interactive client */
                    CLIENT_SSL                  .... 0... .... .... | OFF | Switch to SSL after handshake */
                    CLIENT_IGNORE_SIGPIPE       ...0 .... .... .... | OFF | IGNORE sigpipes */
                    CLIENT_TRANSACTIONS         ..0. .... .... .... | OFF | Client knows about transactions */
                    CLIENT_RESERVED         .0.. .... .... .... | OFF | Old flag for 4.1 protocol  */
                    CLIENT_SECURE_CONNECTION  1... .... .... .... | ON  | New 4.1 authentication */

                                                    1... .11. .... ...1 ==> 16#8601

            Values is in the description of the Handshake
            Initialisation Packet, for server_capabilities.
            For some of the bits, the server passed "what
            it's capable of". The client leaves some of the
            bits on, adds others, and passes back to the server.
            One important flag is: whether compression is desired.
            Another interesting one is CLIENT_CONNECT_WITH_DB,
            which shows the presence of the optional databasename.
    *)

    gcMySql_wExtClientFlags : WORD := 16#0003; (* Extend CLIENT_xxx options.

                            CLIENT_MULTI_STATEMENTS      .... .... .... ...1 | ON  | Enable/disable multi-stmt support */
                            CLIENT_MULTI_RESULTS         .... .... .... ..1. | ON  | Enable/disable multi-results */
        *)


END_VAR
VAR_GLOBAL
END_VAR
```

Globale_Variablen

```
VAR_GLOBAL
END_VAR
```

Alarm configuratior

- Alarm configuratior
  - Alarm classes
  - System

PLC Configuration

*PLC Configuration (Id.: 8765)
Node number: -1
Input address: %IB0
Output address: %QB0
Diagnostic address: %MB0
Download: 1
AutoAdr: 1

K-Bus (PFC200 CS 2ETH RS *) [FIX] (Id.: 11994)
Node number: 0
Input address: %IB0
Output address: %QB0
Diagnostic address: %MB0

Download: 1
AutoAdr: 1
Parameter:
Bus Mode: 1
Cycle time (us): 10000
KBus Thread Priority: 0
Set outputs to zero on stop: 1

*0750-0530 8 DO 24 V DC 0.5A[VAR] (Id.: 2000010008)
Node number: 0
Input address: %IB8
Output address: %QB0
Diagnostic address: %MB0
Download: 1
AutoAdr: 1
Channels:
AT %QX0.0: BOOL; (* Ch_1 Digital output *) [CHANNEL (Q)]
pumppu AT %QX0.1: BOOL; (* Ch_2 Digital output *) [CHANNEL (Q)]
AT %QX0.2: BOOL; (* Ch_3 Digital output *) [CHANNEL (Q)]
AT %QX0.3: BOOL; (* Ch_4 Digital output *) [CHANNEL (Q)]
AT %QX0.4: BOOL; (* Ch_5 Digital output *) [CHANNEL (Q)]
AT %QX0.5: BOOL; (* Ch_6 Digital output *) [CHANNEL (Q)]
AT %QX0.6: BOOL; (* Ch_7 Digital output *) [CHANNEL (Q)]
AT %QX0.7: BOOL; (* Ch_8 Digital output *) [CHANNEL (Q)]

*0750-0430 8 DI 24 V DC 3.0ms[VAR] (Id.: 2000001008)
Node number: 1
Input address: %IB8
Output address: %QB0
Diagnostic address: %MB0
Download: 1
AutoAdr: 1
Channels:
laatikko1 AT %IX4.0: BOOL; (* Ch_1 Digital input *) [CHANNEL (I)]
laatikko2 AT %IX4.1: BOOL; (* Ch_2 Digital input *) [CHANNEL (I)]
laatikko3 AT %IX4.2: BOOL; (* Ch_3 Digital input *) [CHANNEL (I)]
laatikko4 AT %IX4.3: BOOL; (* Ch_4 Digital input *) [CHANNEL (I)]
AT %IX4.4: BOOL; (* Ch_5 Digital input *) [CHANNEL (I)]
AT %IX4.5: BOOL; (* Ch_6 Digital input *) [CHANNEL (I)]
termostaatti AT %IX4.6: BOOL; (* Ch_7 Digital input *) [CHANNEL (I)]
suppilo AT %IX4.7: BOOL; (* Ch_8 Digital input *) [CHANNEL (I)]

*0750-0468 4 AI 0-10V S.E.[VAR] (Id.: 2000008004)
Node number: 2
Input address: %IB0
Output address: %QB0
Diagnostic address: %MB0
Download: 1
AutoAdr: 1
Channels:
savu AT %IW0: WORD; (* Ch_1 Input word *) [CHANNEL (I)]
AT %IW1: WORD; (* Ch_2 Input word *) [CHANNEL (I)]
happi AT %IW2: WORD; (* Ch_3 Input word *) [CHANNEL (I)]
vesi AT %IW3: WORD; (* Ch_4 Input word *) [CHANNEL (I)]

*Modbus variables[FIX] (Id.: 2010330001)
Node number: 1
Input address: %IB0
Output address: %QB0
Diagnostic address: %MB0
Download: 1
AutoAdr: 1
Parameter:
PLC stop behaviour: 1
Field Bus error behaviour: 1
Modbus TCP operation: 1
TCP port: 502
TCP Timeout [in 100 ms]: 600
Modbus UDP operation: 1
UDP port: 502
Modbus RTU operation: 0
Node ID: 1
Response timeout [ms]: 5000

Interface name: 0
Baudrate: 9600
Number of stop bits: 2
Parity: 0
Flow Ccntrol: 0
Physical Interface: 1

Sampling Trace

No trace loaded

Task configuration

```
Task configuration
    System events
    main (PRIORITY := 6, INTERVAL := T#50ms)
        ohjaus();
        moottorit();
        muunnokset();
        PLC_PRG();
        valvonta();
        sql();
    modbus (PRIORITY := 20, INTERVAL := T#50ms)
        modbus();
```

Watch- and Recipe Manager

Standard
Workspace

Parameter Manager

0001	Parameter-Manager
0002	==========

Cross Reference List

z		
	_global_init (6)	Local Write
	OHJAUS (3)	Local Write
	OHJAUS (4)	Local Read
Y_MIN		
	UUSIPIDinit (15)	Local Write
	UUSIPID (41)	Local Read
	UUSIPID (47)	Local Read
	UUSIPID (48)	Local Read
	OHJAUS (3)	Local Write
Y_OFFSET		
	UUSIPIDinit (14)	Local Write
	UUSIPID (10)	Local Read
	UUSIPID (15)	Local Read
	UUSIPID (35)	Local Read
	OHJAUS (3)	Local Write
Y_MAX		
	UUSIPIDinit (16)	Local Write
	UUSIPID (41)	Local Read
	UUSIPID (42)	Local Read
	UUSIPID (47)	Local Read
	OHJAUS (3)	Local Write
Y_MANUAL		
	UUSIPIDinit (13)	Local Write
	UUSIPID (14)	Local Read
	UUSIPID (15)	Local Read
	OHJAUS (3)	Local Write
y		
	UUSIPIDinit (20)	Local Write
	UUSIPID (10)	Local Write
	UUSIPID (14)	Local Write
	UUSIPID (35)	Local Write
	UUSIPID (36)	Local Read
	UUSIPID (41)	Local Read
	UUSIPID (42)	Local Write
	UUSIPID (47)	Local Read
	UUSIPID (48)	Local Write
	OHJAUS (3)	Local Read
Y		
	_global_init (5)	Local Write
	OHJAUS (2)	Local Write
	OHJAUS (4)	Local Read

xStart	
_global_init (6)	Local Write
SQL (1)	Local Write
SQL (1)	Local Read
SQL (2)	Local Read
SQL (2)	Local Write
vika	
MOOTTORIT (1)	Global Read
MOOTTORIT (2)	Global Read
MOOTTORIT (3)	Global Read
_global_init (4)	Global Write
OHJAUS (1)	Global Read
OHJAUS (1)	Global Write
vesi%IW3	
MUUNNOKSET (9)	Global Read
_global_init (80)	Global Write
VALVONTA (2)	Global Read
OHJAUS (5)	Global Read
vesi%IW3	
MUUNNOKSET (9)	Global Read
_global_init (80)	Global Write
VALVONTA (2)	Global Read
OHJAUS (5)	Global Read
Y_ADDOFFSET	
UUSIPIDinit (7)	Local Write
UUSIPID (12)	Local Write
UUSIPID (15)	Local Write
UUSIPID (35)	Local Read
varastovika	
MOOTTORIT (3)	Global Read
_global_init (42)	Global Write
VALVONTA (7)	Global Read
VALVONTA (7)	Global Write
xConnected	
_global_init (2)	Local Write
SQL (1)	Local Write
SQL (2)	Local Read
SQL (5)	Local Read
varasto	
MOOTTORIT (3)	Global Write
MOOTTORIT (4)	Global Read
MOOTTORIT (5)	Global Read
_global_init (29)	Global Write
VALVONTA (7)	Global Read
X2	
DERIVAATTORI (3)	Local Write
DERIVAATTORI (15)	Local Read
DERIVAATTORI (16)	Local Read
DERIVAATTORI (17)	Local Write
DERIVAATTORIinit (2)	Local Write
unpack2	
MOOTTORIT (5)	Local Write
MOOTTORIT (5)	Local Read
_global_init (1)	Local Read
u	
_global_init (12)	Local Write
OHJAUS (5)	Local Write
OHJAUS (7)	Local Read
X3	
DERIVAATTORI (4)	Local Write
DERIVAATTORI (15)	Local Read
DERIVAATTORI (16)	Local Write
DERIVAATTORIinit (1)	Local Write
TV	
UUSIPIDinit (12)	Local Write
UUSIPID (15)	Local Read
UUSIPID (35)	Local Read
OHJAUS (3)	Local Write
X1	
DERIVAATTORI (2)	Local Write
DERIVAATTORI (15)	Local Read
DERIVAATTORI (17)	Local Read
DERIVAATTORI (18)	Local Write

DERIVAATTORIinit (3)	Local Write
unpack4	
MOOTTORIT (5)	Local Write
MOOTTORIT (5)	Local Read
_global_init (3)	Local Read
unpack3	
MOOTTORIT (5)	Local Write
MOOTTORIT (5)	Local Read
_global_init (2)	Local Read
tonq	
DERIVAATTORI (10)	Local Write
DERIVAATTORI (14)	Local Read
DERIVAATTORIinit (8)	Local Write
u3	
_global_init (9)	Local Write
VALVONTA (3)	Local Write
VALVONTA (4)	Local Read
ton4	
_global_init (10)	Local Read
VALVONTA (6)	Local Write
VALVONTA (6)	Local Read
ton2	
_global_init (7)	Local Read
VALVONTA (5)	Local Write
VALVONTA (5)	Local Read
ton2	
_global_init (14)	Local Read
SQL (5)	Local Write
SQL (5)	Local Read
ton3	
_global_init (8)	Local Read
VALVONTA (7)	Local Write
VALVONTA (7)	Local Read
ton1	
_global_init (6)	Local Read
VALVONTA (4)	Local Write
VALVONTA (4)	Local Read
tof1	
MOOTTORIT (3)	Local Write
MOOTTORIT (3)	Local Read
_global_init (4)	Local Read
TON1	
DERIVAATTORI (10)	Local Write
DERIVAATTORIinit (7)	Local Read
tof2	
MOOTTORIT (2)	Local Write
MOOTTORIT (2)	Local Read
_global_init (5)	Local Read
TMDIFF	
UUSIPIDinit (4)	Local Write
UUSIPID (17)	Local Write
UUSIPID (22)	Local Write
UUSIPID (25)	Local Read
UUSIPID (31)	Local Read
UUSIPID (44)	Local Read
UUSIPID (50)	Local Read
ton2	
_global_init (10)	Local Read
OHJAUS (7)	Local Read
OHJAUS (8)	Local Write
OHJAUS (8)	Local Read
tilasanat	
MUUNNOKSET (38)	Global Write
MUUNNOKSET (41)	Global Write
MUUNNOKSET (44)	Global Write
MOOTTORIT (5)	Global Read
_global_init (68)	Global Write
TN	
UUSIPIDinit (11)	Local Write
UUSIPID (1)	Local Read
UUSIPID (15)	Local Read
UUSIPID (35)	Local Read
OHJAUS (3)	Local Write

termostaatti2	
MOOTTORIT (1)	Global Read
MOOTTORIT (2)	Global Read
MOOTTORIT (3)	Global Read
_global_init (31)	Global Write
SQLSTATEMENT (21)	Global Read
VALVONTA (4)	Global Read
OHJAUS (2)	Global Read
OHJAUS (3)	Global Read
OHJAUS (6)	Global Write
OHJAUS (8)	Global Read
TM	
DERIVAATTORI (10)	Local Read
DERIVAATTORI (14)	Local Read
DERIVAATTORI (15)	Local Read
DERIVAATTORI (20)	Local Read
DERIVAATTORIinit (11)	Local Write
UUSIPID (30)	Local Write
termostaatti%IX4.6	
_global_init (76)	Global Write
OHJAUS (1)	Global Read
termostaatti%IX4.6	
_global_init (76)	Global Write
OHJAUS (1)	Global Read
T2	
DERIVAATTORI (15)	Local Read
DERIVAATTORI (19)	Local Write
DERIVAATTORIinit (4)	Local Write
state	
MODBUS (20)	Local Read
MODBUS (31)	Local Write
MODBUS (35)	Local Read
MODBUS (48)	Local Write
MODBUS (54)	Local Read
MODBUS (68)	Local Write
MODBUS (73)	Local Read
MODBUS (108)	Local Write
MODBUS (111)	Local Read
MODBUS (140)	Local Write
MODBUS (143)	Local Read
MODBUS (174)	Local Write
_global_init (2)	Local Write
T1	
DERIVAATTORI (15)	Local Read
DERIVAATTORI (19)	Local Read
DERIVAATTORI (20)	Local Write
DERIVAATTORIinit (5)	Local Write
suppilo%IX4.7	
MOOTTORIT (3)	Global Read
_global_init (77)	Global Write
SQLSTATEMENT (19)	Global Read
suppilo%IX4.7	
MOOTTORIT (3)	Global Read
_global_init (77)	Global Write
SQLSTATEMENT (19)	Global Read
StartFunction	
MODBUS (22)	Global Read
MODBUS (30)	Global Write
MODBUS (37)	Global Read
MODBUS (47)	Global Write
MODBUS (50)	Global Write
MODBUS (56)	Global Read
MODBUS (67)	Global Write
MODBUS (70)	Global Write
MODBUS (75)	Global Read
MODBUS (105)	Global Write
MODBUS (107)	Global Write
MODBUS (113)	Global Read
MODBUS (139)	Global Write
MODBUS (145)	Global Read
MODBUS (173)	Global Write
MODBUS (193)	Global Write
MODBUS (193)	Global Read

_global_init (1)	Global Write
sStatus	
_global_init (3)	Local Write
SQL (1)	Local Write
ton1	
_global_init (7)	Local Read
SQL (2)	Local Write
SQL (2)	Local Read
SOCKET_STREAM	
MYSQL_LOGIN (77)	Global Read
_global_init (214)	Global Write
SOCKET_TCP_NODELAY	
MYSQL_LOGIN (86)	Global Read
_global_init (212)	Global Write
SOCKET_SO_USELOOPBACK	
_global_init (199)	Global Write
SOCKET_SO_TYPE	
_global_init (210)	Global Write
SOCKET_SO_REUSEPORT	
_global_init (202)	Global Write
SOCKET_SO_RCVBUF	
_global_init (204)	Global Write
sStatusexecute	
_global_init (13)	Local Write
SQL (4)	Local Write
SOCKET_SO_KEEPALIVE	
_global_init (196)	Global Write
startexecute	
_global_init (12)	Local Write
SQL (4)	Local Write
SQL (4)	Local Read
SQL (5)	Local Read
SQL (5)	Local Write
SOCKET_INVALID	
_global_init (160)	Global Write
sqlstatement	
SQL (3)	Local Read
SQLSTATEMENT (2)	Local Write
SQLSTATEMENT (3)	Local Write
SQLSTATEMENT (4)	Local Write
SQLSTATEMENT (3)	Local Write
SQLSTATEMENT (4)	Local Write
SQLSTATEMENT (5)	Local Write
SQLSTATEMENT (6)	Local Write
SQLSTATEMENT (7)	Local Write
SQLSTATEMENT (8)	Local Write
SQLSTATEMENT (9)	Local Write
SQLSTATEMENT (10)	Local Write
SQLSTATEMENT (11)	Local Write
SQLSTATEMENT (12)	Local Write
SQLSTATEMENT (13)	Local Write
SQLSTATEMENT (14)	Local Write
SQLSTATEMENT (15)	Local Write
SQLSTATEMENT (16)	Local Write
SQLSTATEMENT (17)	Local Write
SQLSTATEMENT (18)	Local Write
SQLSTATEMENT (19)	Local Write
SQLSTATEMENT (20)	Local Write
SQLSTATEMENT (21)	Local Write
SQLSTATEMENT (22)	Local Write
SQLSTATEMENT (23)	Local Write
SQLSTATEMENT (24)	Local Write
SQLSTATEMENT (25)	Local Write
SQLSTATEMENT (26)	Local Write
SQLSTATEMENT (27)	Local Write
SOCKET_TCP_MAXSEG	
_global_init (213)	Global Write
SOCKET_SO_DEBUG	
_global_init (193)	Global Write
SOCKET_SOL	
_global_init (192)	Global Write
SOCKET_SO_SNDTIMEO	
_global_init (207)	Global Write
SOCKET_SO_SNDLOWAT	
_global_init (205)	Global Write

SOCKET_SO_RCVLOWAT	
_global_init (206)	Global Write
SOCKET_SO_SNDBUF	
_global_init (203)	Global Write
SOCKET_SO_RCVTIMEO	
_global_init (208)	Global Write
SOCKET_SO_PROTOTYPE	
_global_init (211)	Global Write
SOCKET_SO_DONTROUTE	
_global_init (197)	Global Write
SOCKET_SO_BROADCAST	
_global_init (198)	Global Write
SOCKET_SO_REUSEADDR	
_global_init (195)	Global Write
SOCKET_SO_OOBINLINE	
_global_init (201)	Global Write
SOCKET_SO_LINGER	
_global_init (200)	Global Write
SOCKET_SO_ERROR	
_global_init (209)	Global Write
SOCKET_RAW	
_global_init (216)	Global Write
SOCKET_SO_ACCEPTCONN	
_global_init (194)	Global Write
SOCKET_IPPROTO_RAW	
_global_init (232)	Global Write
SOCKET_SEQPACKET	
_global_init (218)	Global Write
SOCKET_IPPROTO_PUP	
_global_init (228)	Global Write
SOCKET_IPPROTO_ND	
_global_init (231)	Global Write
SOCKET_RDM	
_global_init (217)	Global Write
SOCKET_INADDR_NONE	
MYSQL_LOGIN (47)	Global Read
_global_init (222)	Global Write
SOCKET_INADDR_LOOPBACK	
_global_init (220)	Global Write
SOCKET_pseudo_AF_XTP	
_global_init (182)	Global Write
SOCKET_INADDR_ANY	
_global_init (219)	Global Write
SOCKET_pseudo_AF_RTIP	
_global_init (185)	Global Write
SOCKET_MSG_OOB	
_global_init (234)	Global Write
SOCKET_IPPROTO_UDP	
_global_init (229)	Global Write
SOCKET_IPPROTO_TCP	
MYSQL_LOGIN (77)	Global Read
_global_init (227)	Global Write
SOCKET_INADDR_BROADCAST	
_global_init (221)	Global Write
SOCKET_DGRAM	
_global_init (215)	Global Write
SOCKET_pseudo_AF_PIP	
_global_init (188)	Global Write
SOCKET_MSG_PEEK	
_global_init (235)	Global Write
SOCKET_MSG_DONTROUTE	
_global_init (236)	Global Write
SOCKET_IPPROTO_MAX	
_global_init (233)	Global Write
SOCKET_IPPROTO_IP	
_global_init (223)	Global Write
SOCKET_IPPROTO_IDP	
_global_init (230)	Global Write
SOCKET_IPPROTO_IGMP	
_global_init (225)	Global Write
SOCKET_IPPROTO_GGP	
_global_init (226)	Global Write
SOCKET_IPPROTO_ICMP	

_global_init (224)	Global Write
SOCKET_FIONREAD	
_global_init (237)	Global Write
SOCKET_FIONBIO	
MYSQL_LOGIN (98)	Global Read
_global_init (238)	Global Write
SOCKET_FD_SETSIZE	
_global_init (239)	Global Write
SOCKET_AF_UNSPEC	
_global_init (161)	Global Write
SOCKET_AF_SNA	
_global_init (174)	Global Write
SOCKET_AF_ROUTE	
_global_init (180)	Global Write
SOCKET_AF_PUP	
_global_init (166)	Global Write
SOCKET_AF_NS	
_global_init (168)	Global Write
SOCKET_AF_UNIX	
_global_init (163)	Global Write
SOCKET_AF_SIP	
_global_init (187)	Global Write
SOCKET_AF_OSI	
_global_init (170)	Global Write
SOCKET_AF_MAX	
_global_init (189)	Global Write
SOCKET_AF_LINK	
_global_init (181)	Global Write
SOCKET_AF_LOCAL	
_global_init (162)	Global Write
SOCKET_AF_LAT	
_global_init (177)	Global Write
SOCKET_AF_IPX	
_global_init (186)	Global Write
SOCKET_AF_INET_BSD	
_global_init (190)	Global Write
SOCKET_AF_ISO	
_global_init (169)	Global Write
SOCKET_AF_INET_STREAMS	
_global_init (191)	Global Write
SOCKET_AF_IMPLINK	
_global_init (165)	Global Write
SOCKET_AF_HYLINK	
_global_init (178)	Global Write
SOCKET_AF_INET	
MYSQL_LOGIN (77)	Global Read
MYSQL_LOGIN (100)	Global Read
_global_init (164)	Global Write
SOCKET_AF_ECMA	
_global_init (171)	Global Write
SOCKET_AF_DLI	
_global_init (176)	Global Write
SOCKET_AF_DECnet	
_global_init (175)	Global Write
SOCKET_AF_DATAKIT	
_global_init (172)	Global Write
SOCKET_AF_COIP	
_global_init (183)	Global Write
SOCKET_AF_CNT	
_global_init (184)	Global Write
SOCKET_AF_CHAOS	
_global_init (167)	Global Write
SOCKET_AF_CCITT	
_global_init (173)	Global Write
SOCKET_AF_APPLETALK	
_global_init (179)	Global Write
SET_POINT	
UUSIPIDinit (9)	Local Write
UUSIPID (2)	Local Read
OHJAUS (3)	Local Write
savu%IW0	
MUUNNOKSET (16)	Global Read
_global_init (78)	Global Write

VALVONTA (1)	Global Read
VALVONTA (3)	Global Read
OHJAUS (2)	Global Read
savu%IW0	
MUUNNOKSET (16)	Global Read
_global_init (78)	Global Write
VALVONTA (1)	Global Read
VALVONTA (3)	Global Read
OHJAUS (2)	Global Read
SANA%MW0	
MUUNNOKSET (37)	Local Write
MUUNNOKSET (40)	Local Write
MUUNNOKSET (43)	Local Write
_global_init (3)	Local Write
sammunut	
_global_init (40)	Global Write
VALVONTA (4)	Global Write
OHJAUS (1)	Global Read
rVersion	
_global_init (255)	Global Write
run	
MOOTTORIT (1)	Global Read
MOOTTORIT (2)	Global Read
MOOTTORIT (3)	Global Read
MOOTTORIT (4)	Global Read
_global_init (5)	Global Write
ruuvi	
MOOTTORIT (1)	Global Write
MOOTTORIT (5)	Global Read
_global_init (30)	Global Write
RING_BUFFER_SIZE	
_global_init (254)	Global Write
Response	
MODBUS (39)	Global Read
MODBUS (40)	Global Read
MODBUS (59)	Global Read
MODBUS (60)	Global Read
MODBUS (76)	Global Read
MODBUS (78)	Global Read
MODBUS (79)	Global Read
MODBUS (78)	Global Read
MODBUS (79)	Global Read
MODBUS (193)	Global Write
MODBUS (193)	Global Read
_global_init (3)	Global Read
rs1	
_global_init (8)	Local Read
OHJAUS (6)	Local Write
OHJAUS (6)	Local Read
RESET	
DERIVAATTORI (1)	Local Read
DERIVAATTORIinit (12)	Local Write
UUSIPID (6)	Local Write
UUSIPID (30)	Local Write
pid2	
MUUNNOKSET (21)	Local Read
MUUNNOKSET (26)	Local Read
_global_init (3)	Local Read
OHJAUS (3)	Local Write
OHJAUS (3)	Local Read
RESET	
UUSIPIDinit (18)	Local Write
UUSIPID (1)	Local Read
UUSIPID (4)	Local Read
UUSIPID (9)	Local Read
OHJAUS (3)	Local Write
pumppuvika	
MOOTTORIT (4)	Global Read
_global_init (70)	Global Write
VALVONTA (6)	Global Read
VALVONTA (6)	Global Write
pumppu%QX0.1	
MOOTTORIT (4)	Global Write
_global_init (71)	Global Write
SQLSTATEMENT (23)	Global Read

VALVONTA (6)	Global Read
pumppu%QX0.1	
MOOTTORIT (4)	Global Write
_global_init (71)	Global Write
SQLSTATEMENT (23)	Global Read
VALVONTA (6)	Global Read
Query	
MODBUS (26)	Global Write
MODBUS (27)	Global Write
MODBUS (28)	Global Write
MODBUS (29)	Global Write
MODBUS (38)	Global Read
MODBUS (43)	Global Write
MODBUS (44)	Global Write
MODBUS (45)	Global Write
MODBUS (46)	Global Write
MODBUS (57)	Global Read
MODBUS (63)	Global Write
MODBUS (64)	Global Write
MODBUS (65)	Global Write
MODBUS (66)	Global Write
MODBUS (76)	Global Read
MODBUS (82)	Global Write
MODBUS (83)	Global Write
MODBUS (84)	Global Write
MODBUS (85)	Global Write
MODBUS (86)	Global Write
MODBUS (87)	Global Write
MODBUS (115)	Global Write
MODBUS (116)	Global Write
MODBUS (117)	Global Write
MODBUS (118)	Global Write
MODBUS (119)	Global Write
MODBUS (120)	Global Write
MODBUS (148)	Global Write
MODBUS (149)	Global Write
MODBUS (150)	Global Write
MODBUS (151)	Global Write
MODBUS (152)	Global Write
MODBUS (153)	Global Write
MODBUS (193)	Global Write
MODBUS (193)	Global Read
_global_init (2)	Global Read
puhallin	
MOOTTORIT (2)	Global Write
MOOTTORIT (5)	Global Read
_global_init (32)	Global Write
pid1	
_global_init (1)	Local Read
OHJAUS (2)	Local Write
OHJAUS (2)	Local Read
OVERFLOW	
UUSIPIDinit (22)	Local Write
UUSIPID (1)	Local Read
UUSIPID (7)	Local Write
UUSIPID (33)	Local Write
UUSIPID (34)	Local Read
UUSIPID (37)	Local Write
OHJAUS (3)	Local Read
oMySql	
_global_init (5)	Local Read
SQL (1)	Local Write
SQL (1)	Local Read
SQL (4)	Local Write
SQL (4)	Local Read
MB_START_ADDRESS_RANGE	
MODBUS_EXTENDED_SL... (17)	Global Read
MODBUS_EXTENDED_SL... (18)	Global Read
MODBUS_EXTENDED_SL... (15)	Global Read
MODBUS_EXTENDED_SL... (16)	Global Read
MODBUS_EXTENDED_SL... (21)	Global Read
MODBUS_EXTENDED_SL... (22)	Global Read
MODBUS_EXTENDED_SL... (42)	Global Read

MODBUS_EXTENDED_SL... (43)	Global Read
MODBUS_EXTENDED_SL... (17)	Global Read
MODBUS_EXTENDED_SL... (18)	Global Read
MODBUS_EXTENDED_SL... (19)	Global Read
MODBUS_EXTENDED_SL... (15)	Global Read
MODBUS_EXTENDED_SL... (16)	Global Read
MODBUS_EXTENDED_SL... (15)	Global Read
MODBUS_EXTENDED_SL... (16)	Global Read
_global_init (258)	Global Write
OUT	
DERIVAATTORI (5)	Local Write
DERIVAATTORI (15)	Local Write
DERIVAATTORIinit (13)	Local Write
UUSIPID (15)	Local Read
UUSIPID (35)	Local Read
MB_SIZE_SLAVEBUFFER	
_global_init (257)	Global Write
m4vika	
MOOTTORIT (5)	Global Write
_global_init (65)	Global Write
OHJAUS (1)	Global Read
m4valmis	
MOOTTORIT (5)	Global Write
_global_init (59)	Global Write
m4ramppi	
MOOTTORIT (5)	Global Write
_global_init (63)	Global Write
m4suunta	
MOOTTORIT (5)	Global Write
_global_init (61)	Global Write
m4hal	
MOOTTORIT (5)	Global Write
_global_init (62)	Global Write
m3vika	
MOOTTORIT (5)	Global Write
_global_init (57)	Global Write
OHJAUS (1)	Global Read
m3valmis	
MOOTTORIT (5)	Global Write
_global_init (51)	Global Write
m3ramppi	
MOOTTORIT (5)	Global Write
_global_init (55)	Global Write
m3suunta	
MOOTTORIT (5)	Global Write
_global_init (53)	Global Write
m3hal	
MOOTTORIT (5)	Global Write
_global_init (54)	Global Write
MB_MAX_BUFFER_SIZE	
MODBUS_EXTENDED_MA... (266)	Global Read
MODBUS_EXTENDED_SLAVE (20)	Global Read
_global_init (256)	Global Write
m2vika	
MOOTTORIT (5)	Global Write
_global_init (49)	Global Write
VALVONTA (7)	Global Read
o3	
_global_init (11)	Local Write
VALVONTA (3)	Local Write
VALVONTA (5)	Local Read
m4nollanopeus	
MOOTTORIT (5)	Global Write
_global_init (66)	Global Write
m2ramppi	
MOOTTORIT (5)	Global Write
_global_init (47)	Global Write
m2nollanopeus	
MOOTTORIT (5)	Global Write
_global_init (50)	Global Write
m2suunta	
MOOTTORIT (5)	Global Write
_global_init (45)	Global Write

m2hal	
MOOTTORIT (5)	Global Write
_global_init (46)	Global Write
MANUAL	
UUSIPIDinit (17)	Local Write
UUSIPID (1)	Local Read
UUSIPID (4)	Local Read
OHJAUS (3)	Local Write
LIMITS_ACTIVE	
UUSIPIDinit (21)	Local Write
UUSIPID (8)	Local Write
UUSIPID (40)	Local Write
UUSIPID (43)	Local Write
UUSIPID (49)	Local Write
SANA%MW0	
MUUNNOKSET (37)	Local Write
MUUNNOKSET (40)	Local Write
MUUNNOKSET (43)	Local Write
_global_init (3)	Local Write
q	
_global_init (7)	Local Write
OHJAUS (6)	Local Read
OHJAUS (7)	Local Write
overflow2	
_global_init (4)	Local Write
OHJAUS (1)	Local Read
OHJAUS (3)	Local Write
overflow1	
_global_init (2)	Local Write
OHJAUS (1)	Local Read
OHJAUS (2)	Local Write
m2valmis	
MOOTTORIT (5)	Global Write
_global_init (43)	Global Write
MB_MAX_JOBS	
_global_init (259)	Global Write
M1	
MODBUS (38)	Local Read
MODBUS (57)	Local Read
MODBUS (76)	Local Read
MODBUS (193)	Local Write
_global_init (1)	Local Read
o	
_global_init (11)	Local Write
OHJAUS (5)	Local Write
OHJAUS (6)	Local Read
m3nollanopeus	
MOOTTORIT (5)	Global Write
_global_init (58)	Global Write
MB_Error	
MODBUS (38)	Global Read
MODBUS (57)	Global Read
MODBUS (76)	Global Read
MODBUS (193)	Global Write
_global_init (4)	Global Write
m2run	
MOOTTORIT (5)	Global Write
_global_init (44)	Global Write
MAX_SOCKET_FD_SETSIZE	
_global_init (240)	Global Write
m4ohjauspaikka	
MOOTTORIT (5)	Global Write
_global_init (64)	Global Write
m4run	
MOOTTORIT (5)	Global Write
_global_init (60)	Global Write
m3ohjauspaikka	
MOOTTORIT (5)	Global Write
_global_init (56)	Global Write
m3run	
MOOTTORIT (5)	Global Write
_global_init (52)	Global Write
m2ohjauspaikka	

MOOTTORIT (5)	Global Write
_global_init (48)	Global Write
login	
_global_init (1)	Local Read
SQL (1)	Local Write
SQL (1)	Local Read
laatikko4%IX4.3	
MOOTTORIT (4)	Global Read
_global_init (75)	Global Write
limit2	
_global_init (2)	Local Read
VALVONTA (2)	Local Write
VALVONTA (2)	Local Read
laatikko3%IX4.2	
MOOTTORIT (4)	Global Read
_global_init (74)	Global Write
LO%MB1	
_global_init (2)	Local Write
LO%MB1	
_global_init (2)	Local Write
laatikko2%IX4.1	
MOOTTORIT (4)	Global Read
_global_init (73)	Global Write
laatikko2%IX4.1	
MOOTTORIT (4)	Global Read
_global_init (73)	Global Write
limit3	
_global_init (3)	Local Read
VALVONTA (3)	Local Write
VALVONTA (3)	Local Read
kuittaus	
MOOTTORIT (5)	Global Read
_global_init (39)	Global Write
VALVONTA (4)	Global Read
VALVONTA (6)	Global Read
VALVONTA (7)	Global Read
OHJAUS (1)	Global Read
OHJAUS (2)	Global Read
OHJAUS (3)	Global Read
limit1	
_global_init (1)	Local Read
VALVONTA (1)	Local Write
VALVONTA (1)	Local Read
laatikko1%IX4.0	
MOOTTORIT (4)	Global Read
_global_init (72)	Global Write
laatikko1%IX4.0	
MOOTTORIT (4)	Global Read
_global_init (72)	Global Write
KP	
UUSIPIDinit (10)	Local Write
UUSIPID (1)	Local Read
UUSIPID (15)	Local Read
UUSIPID (35)	Local Read
OHJAUS (3)	Local Write
INIT	
DERIVAATTORI (1)	Local Read
DERIVAATTORI (6)	Local Write
DERIVAATTORIinit (6)	Local Write
IN	
DERIVAATTORI (2)	Local Read
DERIVAATTORI (3)	Local Read
DERIVAATTORI (4)	Local Read
DERIVAATTORI (15)	Local Read
DERIVAATTORI (18)	Local Read
DERIVAATTORIinit (10)	Local Write
UUSIPID (30)	Local Write
il2	
_global_init (4)	Local Write
VALVONTA (2)	Local Write
VALVONTA (5)	Local Read
limit1	
_global_init (9)	Local Read

OHJAUS (5)	Local Write
OHJAUS (5)	Local Read
laatikko4%IX4.3	
MOOTTORIT (4)	Global Read
_global_init (75)	Global Write
laatikko3%IX4.2	
MOOTTORIT (4)	Global Read
_global_init (74)	Global Write
HI%MB0	
MUUNNOKSET (38)	Local Read
MUUNNOKSET (41)	Local Read
MUUNNOKSET (44)	Local Read
_global_init (1)	Local Write
INIT	
UUSIPIDinit (6)	Local Write
UUSIPID (4)	Local Read
UUSIPID (9)	Local Read
UUSIPID (11)	Local Write
happi%IW2	
MUUNNOKSET (48)	Global Read
_global_init (79)	Global Write
SQLSTATEMENT (5)	Global Read
OHJAUS (3)	Global Read
il	
_global_init (13)	Local Write
OHJAUS (5)	Local Write
gcMySql_wExtClientFlags	
MYSQL_LOGIN (231)	Global Read
MYSQL_LOGIN (232)	Global Read
_global_init (253)	Global Write
gcMySql_wClientFlags	
MYSQL_LOGIN (228)	Global Read
MYSQL_LOGIN (229)	Global Read
_global_init (252)	Global Write
I	
MUUNNOKSET (26)	Local Read
UUSIPIDinit (2)	Local Read
UUSIPID (5)	Local Write
UUSIPID (15)	Local Read
UUSIPID (31)	Local Write
UUSIPID (33)	Local Read
UUSIPID (35)	Local Read
UUSIPID (44)	Local Write
UUSIPID (50)	Local Write
happi%IW2	
MUUNNOKSET (48)	Global Read
_global_init (79)	Global Write
SQLSTATEMENT (5)	Global Read
OHJAUS (3)	Global Read
gcMySql_tTimeOut	
MYSQL_LOGIN (481)	Global Read
MYSQL_LOGOUT (84)	Global Read
MYSQL_EXECUTE (191)	Global Read
MYSQL_QUERY (368)	Global Read
_global_init (247)	Global Write
gcMySql_xClearResultSetBeforeUse	
MYSQL_QUERY (6)	Global Read
_global_init (244)	Global Write
gcMySql_iSqlUpperBound	
MYSQL_EXECUTE (40)	Global Read
MYSQL_QUERY (61)	Global Read
_global_init (1)	Global Write
gcMySql_iSqlLength	
_global_init (2)	Global Write
gcMySql_dwRxBufferSize	
_global_init (249)	Global Write
gcMySql_tResponseCompliedlyReceived	
MYSQL_QUERY (139)	Global Read
_global_init (245)	Global Write
gcMySql_dwMaxRowSize	
MYSQL_QUERY (24)	Global Read
MYSQL_QUERY (324)	Global Read
_global_init (243)	Global Write

gcMySql_iSqlLength	
gcMySql_dwMaxPacketSize	
MYSQL_LOGIN (234)	Global Read
MYSQL_LOGIN (235)	Global Read
MYSQL_LOGIN (236)	Global Read
MYSQL_LOGIN (237)	Global Read
_global_init (250)	Global Write
gcMySql_dwMaxRowCount	
MYSQL_QUERY (23)	Global Read
MYSQL_QUERY (323)	Global Read
_global_init (242)	Global Write
gcMySql_dwMaxIdentifierLength	
MYSQL_QUERY (240)	Global Read
MYSQL_QUERY (250)	Global Read
_global_init (246)	Global Write
datawrite3	
MOOTTORIT (5)	Global Write
MODBUS (118)	Global Read
MODBUS (120)	Global Read
_global_init (24)	Global Write
SQLSTATEMENT (11)	Global Read
OHJAUS (4)	Global Write
datawrite2	
MOOTTORIT (5)	Global Write
MODBUS (85)	Global Read
MODBUS (87)	Global Read
_global_init (21)	Global Write
gcMySql_dwMaxColumnCount	
MYSQL_QUERY (15)	Global Read
MYSQL_QUERY (219)	Global Read
_global_init (241)	Global Write
datawrite1	
_global_init (18)	Global Write
dataread4	
MUUNNOKSET (43)	Global Read
MODBUS (79)	Global Write
_global_init (15)	Global Write
gcMySql_bCharsetNumber	
MYSQL_LOGIN (239)	Global Read
_global_init (251)	Global Write
datawrite4	
MOOTTORIT (5)	Global Write
MODBUS (151)	Global Read
MODBUS (153)	Global Read
_global_init (27)	Global Write
SQLSTATEMENT (13)	Global Read
OHJAUS (4)	Global Write
dataread3	
MUUNNOKSET (40)	Global Read
MODBUS (60)	Global Write
_global_init (12)	Global Write
SQLSTATEMENT (25)	Global Read
dataread2	
MUUNNOKSET (37)	Global Read
MODBUS (40)	Global Write
_global_init (9)	Global Write
DAIKA	
UUSIPIDinit (19)	Local Write
UUSIPID (30)	Local Read
OHJAUS (3)	Local Write
D	
UUSIPIDinit (3)	Local Read
UUSIPID (6)	Local Write
UUSIPID (15)	Local Read
UUSIPID (30)	Local Write
UUSIPID (35)	Local Read
il1	
_global_init (5)	Local Write
VALVONTA (1)	Local Write
VALVONTA (5)	Local Read
i	
MODBUS (39)	Local Write
MODBUS (39)	Local Read

MODBUS (40)	Local Read
MODBUS (41)	Local Read
MODBUS (41)	Local Write
MODBUS (59)	Local Write
MODBUS (59)	Local Read
MODBUS (60)	Local Read
MODBUS (61)	Local Read
MODBUS (61)	Local Write
MODBUS (78)	Local Write
MODBUS (78)	Local Read
MODBUS (79)	Local Read
MODBUS (80)	Local Read
MODBUS (80)	Local Write
_global_init (3)	Local Write
CLOCK	
UUSIPIDinit (1)	Local Read
UUSIPID (18)	Local Write
UUSIPID (19)	Local Write
UUSIPID (21)	Local Read
UUSIPID (22)	Local Read
UUSIPID (26)	Local Write
UUSIPID (27)	Local Write
UUSIPID (53)	Local Write
HI%MB0	
MUUNNOKSET (38)	Local Read
MUUNNOKSET (41)	Local Read
MUUNNOKSET (44)	Local Read
_global_init (1)	Local Write
gcMySql_iSqlUpperBound	
gcMySql_dwTxBufferSize	
_global_init (248)	Global Write
executedierror	
_global_init (15)	Local Write
SQL (4)	Local Write
execute	
_global_init (8)	Local Read
SQL (4)	Local Write
SQL (4)	Local Read
ERROR	
MUUNNOKSET (21)	Local Read
UUSIPIDinit (5)	Local Write
UUSIPID (2)	Local Write
UUSIPID (15)	Local Read
UUSIPID (31)	Local Read
UUSIPID (35)	Local Read
UUSIPID (44)	Local Read
UUSIPID (50)	Local Read
dataread1	
_global_init (6)	Global Write
diError	
_global_init (4)	Local Write
SQL (1)	Local Write
aputime	
MUUNNOKSET (31)	Global Write
_global_init (37)	Global Write
asSqlCommand	
_global_init (10)	Local Write
SQL (3)	Local Write
SQL (4)	Local Read
asetukset	
MUUNNOKSET (3)	Global Read
MUUNNOKSET (4)	Global Read
MUUNNOKSET (31)	Global Read
_global_init (2)	Global Write
OHJAUS (2)	Global Read
OHJAUS (3)	Global Read
apumuuttujat	
MUUNNOKSET (3)	Global Write
MUUNNOKSET (4)	Global Write
MUUNNOKSET (9)	Global Write
MUUNNOKSET (16)	Global Write
MUUNNOKSET (21)	Global Write
MUUNNOKSET (26)	Global Write

MUUNNOKSET (48)	Global Write
_global_init (34)	Global Write
SQLSTATEMENT (7)	Global Read
SQLSTATEMENT (9)	Global Read
SQLSTATEMENT (15)	Global Read
SQLSTATEMENT (17)	Global Read
OHJAUS (5)	Global Read
anturivika	
_global_init (41)	Global Write
VALVONTA (5)	Global Write
OHJAUS (1)	Global Read
ajastus	
DERIVAATTORI (10)	Local Read
DERIVAATTORI (12)	Local Write
DERIVAATTORI (21)	Local Write
DERIVAATTORIinit (9)	Local Write
ACTUAL	
UUSIPIDinit (8)	Local Write
UUSIPID (2)	Local Read
UUSIPID (30)	Local Read
OHJAUS (3)	Local Write
%MW0	
MYSQL_CLOSE (3)	Local Read
MYSQL_LOGIN (57)	Local Read
MYSQL_LOGIN (59)	Local Read
MYSQL_LOGIN (81)	Local Read

Call Tree of PLC_PRG (PRG-LD)