LAPIN AMK
Lapland University of Applied Sciences

TUTKAN LÄHETTIMEN TEHOLÄHTEEN VALVONTAKORTIN TESTAUSLAITE

Juha-Matti Rensujeff

Opinnäytetyö
Tekniikan ja liikenteen ala
Tieto- ja viestintätekniikan koulutus
Tietotekniikan insinööri (AMK)

2016

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tekniikan ja liikenteen ala
Tieto- ja viestintätekniikan koulutus
Tietotekniikan insinööri (AMK)

Tekijä Juha-Matti Rensujeff Vuosi 2016
Ohjaaja Kenneth Karlsson
Toimeksiantaja Millog Oy, valvojana insinööri Janne Linkosuonio
Työn nimi Tutkan lähettimen teholähteen valvontakortin testauslaite
Sivu- ja liitesivumäärä 33 + 16

Opinnäytetyön toimeksiantona oli suunnitella ja toteuttaa tutkan teholähteen valvontakortin testauslaite Millog Oy Riihimäen toimipaikan käyttöön. Testauslaitteen avulla kyetään kehittämään piirikorttitason korjauskykyä ja lyhentämään käytettävyyden palauttamiseen kuluvaa aikaa.

Testauslaitteen vaatimusten mukaisesti sen avulla voidaan testata kaikki testattavan kortin toiminnot. Testattaviin toimintoihin kuuluu erilaisten tilatietojen ja synkronointipulssien testaus ja MINIBUS-sarjaväylän lukeminen. Testauksessa tarvittavien signaalien muodostamiseen tarvittiin analogi-, digitaali- ja mikrokontrolleritekniikkaa. Laitteen avulla testauksen suorittaja voi paikallistaa vian, vialliseen komponentin tai komponenttiryhmän tarkkuudella.

Tässä opinnäytetyön raportissa käydään läpi testauslaitteen suunnittelu, valmistus ja käyttöönottotestaus. Koska työ oli laaja-alainen, on painopiste kohdistettu elektroniikka ja ohjelmistosuunnitteluun, mutta myös käyttöliittymä ja mekaniikka suunnittelun ratkaisut käydään läpi. Lisäksi testaus ja käyttöönottovaiheen aikaiset ongelmat ja niiden ratkaisut on kuvattu raportissa.

Opinnäytetyön tuloksena valmistui toimiva piirikortintestauslaite, joka on luovutettu toimeksiantajalle.

Avainsanat piirikortintestauslaite, mikrokontrolleri, MINIBUS-sarjaväylä

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Technology, Communication and transport
Degree Programme in information Technology

Author Juha-Matti Rensujeff Year 2016
Supervisor(s) Kenneth Karlsson
Commissioned by Millog Oy, supervisor engineer Janne Linkosuonio
Subject of thesis Testing device for radar transmitter power supply supervision PCB
Number of pages 33 + 16

The purpose of this thesis was to design and produce a PCB testing device for Millog Oy Riihimäki unit. A PCB to be tested is a supervision PCB from a radar transmitter's power supply. Purpose of the testing device was to enhance PCB repair ability and shorten time of repair.

Requirement for the testing device specified that all functions of tested PCB needed to be tested. Tested functions included logic inputs, synchronization signals and reading of the MINIBUS-interface. To be able to generate needed signals it was necessary to use analog, digital and microcontroller technologies. Tester can find and locate a faulty component or component group from malfunctioning PCB with help of the testing device.

This report introduces design, manufacture and deployment testing of the testing device. Main focus is on electronic and software development but also mechanical designing is described. Also problems that were found in testing and deployment phases and solution for fixing them are bring out.

As a result of the thesis was working PCB testing device that is delivered to the client.

Key words PCB testing device, microcontroller, MINIBUS-serial bus

SISÄLLYS

KÄYTETYT MERKIT JA LYHENTEET

MINIBUS	Testattavan kortin valmistajan oma differentiaalinen, vuorosuuntainen ja synkroninen sarjaväylä.

1 JOHDANTO

Opinnäytetyön toimeksiantona oli suunnitella ja toteuttaa tutkan teholähteen valvontakortin testauslaite Millog Oy Riihimäen toimipaikan käyttöön. Testauslaitteen käyttötarkoitus on helpottaa piirikortin komponenttitason vianhakua ja korjausta. Testauslaitteen avulla helpotetaan piirikorttien viankorjausta ja lyhennetään käytettävyyden palauttamiseen kuluvaa aikaa.

Työ jakaantui tutkimus, suunnittelu, ohjelmointi, kokoonpano ja testausvaiheisiin. Tutkimusvaiheessa tutustuttiin testattavaan piirikorttiin ja sen ominaisuuksiin testauslaitteen vaatimusmäärittelyn näkökohdista. Suunnitteluvaiheessa suunniteltiin testauslaitteen käyttöliittymä, piirikortit ja ohjelmistonrakenne. Ohjelmointivaiheessa ohjelmoitiin testauslaitteen mikrokontrollerin ohjelmisto. Kokoonpanovaiheessa tehtiin mekaniikkatyöt koteloinnin osalta, koottiin piirilevyt ja tehtiin laitteen kokoonpano. Testausvaiheessa laitteen toiminta tarkastettiin ja tehtiin tarvittavat muutokset.

Työn kirjallisessa osuudessa käydään läpi työn eri vaiheet ja niissä tehdyt ratkaisut. Koska työ on laaja-alainen, painopisteenä ovat elektroniikka ja ohjelmistosuunnittelun osa-alueet.

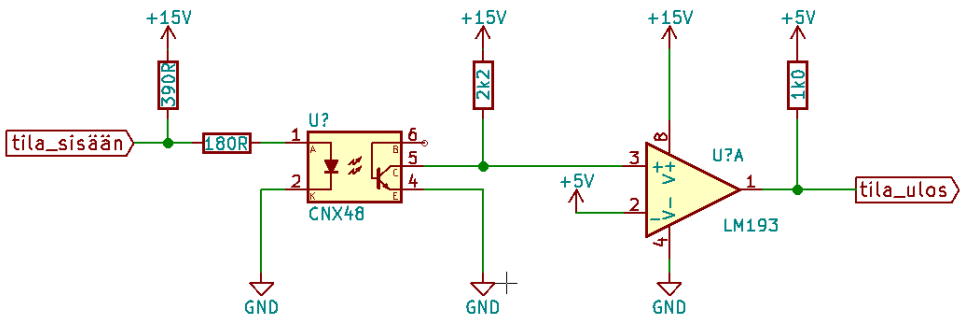
Millog Oy on Puolustusvoimien strateginen kumppani ja vastaa maa- ja merivoimien materiaalin kunnossapidosta. Sen 22 toimipaikalla on noin 1000 työntekijää. Millog on osa Patria konsernia ja sen pääomistajat ovat Patria Oyj ja Insta Group Oy.

2 TESTATTAVA PIIRIKORTTI

Valvontakortin tehtävänä on tarkkailla tutkan eri osien tilatietoja ja sallia lähettimen tarvitsemien jännitteiden muodostus. Lisäksi kortin vastaanottamat tilatiedot siirretään MINIBUS sarjaväylän kautta tutkan tiedonkäsittely-yksikölle. Kortin tärkeimpiä tehtäviä on suojella lähettimen kulkuaaltoputkea mahdollisilta vikatilanteilta.

Valvontakortti kytketään lähettimen teholähteen emolevyyn HE810 sarjan liittimellä jossa on 126 tavallista pinniä ja 6 koaksiaalista liityntää. Valvontakortti käyttää kolmea eri käyttöjännitettä 5 VDC, 15 VDC ja -15 VDC jotka muodostetaan tutkan teholähteessä olevilla virtalähteillä.

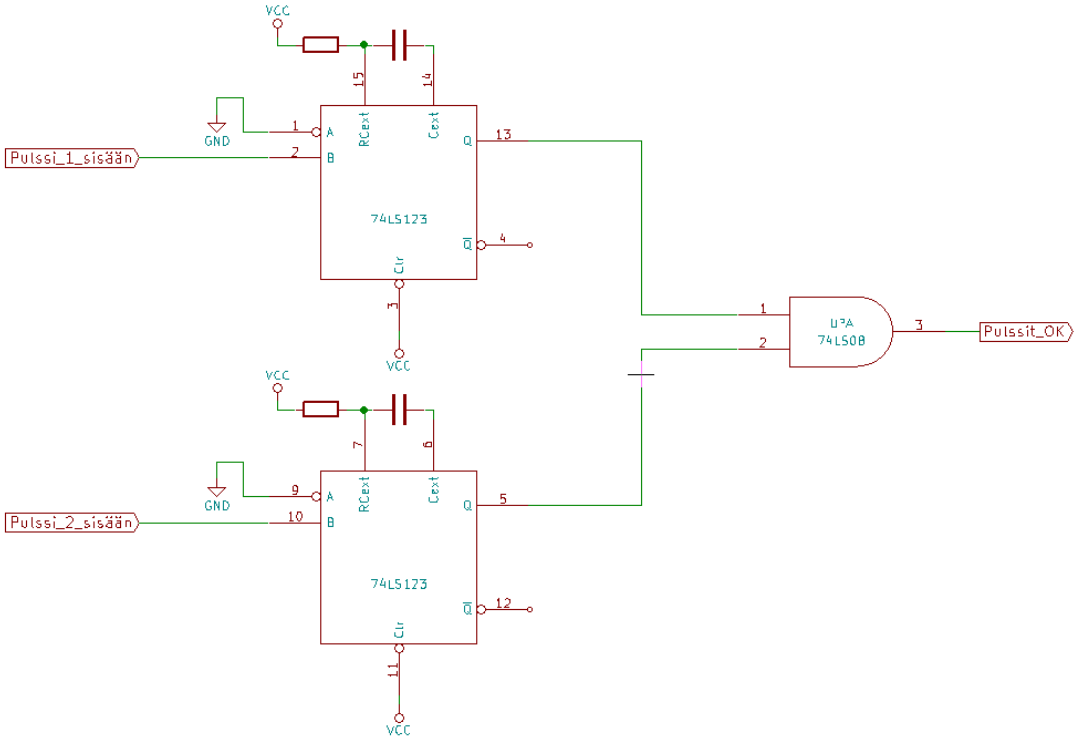
Valvontakortille tulee useita tilatietoja jotka muutetaan logiikkatasoisiksi komparaattoreilla. Osa signaaleista on galvaanisesti erotettu käyttämällä optoerottimia tulon ja komparaattorin välissä. Kuviossa 1 on esitetty optoerottimella galvaanisesti erotettu tulo ja sen kytkentä komparaattoriin tuloon. Pääsääntöisesti tulosignaalit on ylösvedetty 15 V jännitteeseen vastuksen läpi, jolloin tulon tila vaihdetaan kytkemällä se maahan esimerkiksi releellä, kytkimellä tai transistorilla.



Kuvio 1. Optoerottimella erotettu tulo ja sen kytkentä komparaattoriin

Tutkan synkronointiin liittyvät signaalit menevät komparaattorin läpi uudelleen liipaistavaan monostabiiliin multivibraattoriin tuloihin. Multivibraattoreiden lähdöt yhdistetään AND-logiikkapiirillä ja näin saadaan logiikkatasoinen tieto molempi-

en synkronointisignaalien olemassaolosta. Kuviossa 2 on yksinkertaistettu esimerkki kahden pulssimuotoisen signaalin tarkkailukytkennästä.



Kuvio 2. Pulssimuotoisten signaalien tarkkailukytkentä

Lähetyspulssin liipaisusta ja ilmaisusta saatavat signaalit käsitellään siten että liipaisusta tietyn ajan kuluessa saapuva ilmaisupulssi vaaditaan lähetyspulssi ok tilatiedon saamiseksi. Käytännössä tämä tehdään siten että Ilmaistu lähetyspulssi tuodaan D-kiikun datatuloon. Liipaisupulssia viivästetään monostabiililla multivibraattorilla ja tuodaan D-kiikun kellotuloon. Näin saadaan lukittua lähetyspulssin tila halutulla hetkellä kiikun ulostuloon.

3 TESTAUSLAITTEEN VAATIMUKSET

Vaatimusmäärittelyn mukaan testauslaitteella tulee voida muuttaa kortin kaikkien tulosignaalien tila. Lisäksi testauslaitteen pitää kyetä muodostamaan tarvittavat synkronointisignaalit, sekä lukemaan MINIBUS-väylää. Testauslaitteen tulee indikoida tiettyjen kortilta lähtevien tilatietojen tila ja näyttää tiettyjen MINIBUS-väylän osoitteiden arvot näytöllä. Lisäksi siinä täytyy olla tarvittavat liitynnät tiettyjen signaaleiden mittaamisen mahdollistamiseksi. Testauslaitteen alkuperäinen vaatimusmäärittely on liitteessä 1.

Testauslaitteen tulee täyttää myös yleiset vaatimukset esimerkiksi ettei se aiheuta lisävaurioita testattavaan korttiin ja ettei se rikkoonnu testattavan kortin vioista johtuen. Myös laitteen käyttöliittymän tulee olla selkeä ja tehdä testaamisesta mahdollisimman joustavaa.

Suunnitteluprosessin aikana havaittiin tarve joillekin muutoksille laitteen vaatimuksiin. Muutoksilla yksinkertaistettiin laitteen käyttöä ja mahdollistetaan laitteeseen kuuluvan korttisovittimen jatkokäyttö myös mahdollisissa tulevissa testauslaitteissa. Seuraavassa listauksessa on vaatimusmäärittelyn muutokset:

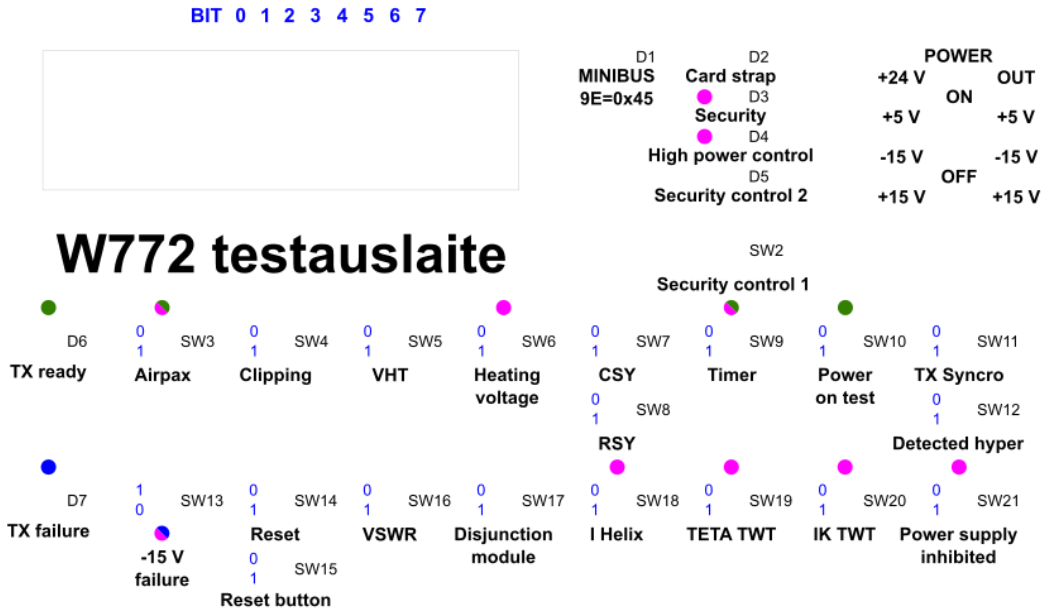
- Testauslaite muodostaa itse tarvittavat jännitteet yhdestä käyttöjännitteestä.
- Testauslaitteen korttisovittimen kaikki pinnit on käytettävissä lattakaapeleiden kautta.
- Testauslaiteella voidaan generoida ”TX failure” tila katkaisemalla -15 V syötön testattavalle kortille.

4 TESTAUSLAITTEEN SUUNNITTELU

4.1 Käyttöliittymä

Käyttöliittymän suunnittelu kulki yhdessä mekaniikan kotelosuunnittelun kanssa. Valitulla koteloratkaisulla käyttöliittymää varten jäi käytettäväksi 138 x 240 mm alue kotelon etupinnassa. Tälle alueelle tuli sijoittaa näyttö, 21 kpl kytkimiä ja 14 kpl ledejä. Suunnittelussa lähtökohta oli sijoittaa kytkimet siten, että niiden fyysinen sijoitus vastaa näytöllä näkyvän bitin sijaintia. Ledien sijoittelussa tavoiteltiin loogista järjestystä ja ryhmittelyä toimintojen mukaan.

Etupaneelin tarran avulla selkeytettiin testauslaitteen käyttöä ja siihen merkittiin eri toimintojen vaatimat kytkinten asennot. Tällöin käyttäjän ei tarvitse erikseen etsiä tietoa mitkä kytkimet vaikuttavat mihinkin toimintoon. Kuviossa 3 on lopullinen etupaneelin tarra josta ilmenee laitteen käyttöliittymän asettelu.



Kuvio 3. Testauslaitteen etupaneelin tarra

4.2 Mekaniikka

4.2.1 Korttisovitin

Testauslaitteen suunnittelussa lähtökohtana oli että itse testauslaite on omassa kotelossaan ja se kytketään kaapeleilla korttisovittimeen, mihin testattava kortti asennetaan testauksen ajaksi. Korttisovittimen runko on vanhasta testauslaitteista irrotettu ja siitä hyödynnettiin kotelo, korttiliitin ja kortintukimuovit.

Vaatimuksena oli että projektin käyttöön annettua korttisovitinta voitaisiin käyttää myös mahdollisissa tulevissa testauslaitteissa. Tämän vuoksi päätettiin korttipohjan kaikki 126 nastaa ja 6 koaksiaali-liitintä johdottaa sovitinkotelon kylkeen vaikkei niitä kaikkia tämän kortin testaamisessa tarvitakaan. Korttipohjan ja sovitinkotelon liittimen välinen johdotus tehtiin kiertoliitos eli wire-wrap-tekniikalla. Kiertoliitos mahdollistaa luotettavan ja nopean tavan liittää suuren määrän johtimia, lisäksi muutosten tekeminen johdotukseen on suhteellisen helppoa. Korttisovittimen kylkeen tehtiin reiät kuudelle BNC-runkoliittimelle ja kahdelle 64-napaiselle lattakaapeliliittimelle. Lisäksi korttisovittimen kotelon avonainen kylki suljettiin peltilevyllä.

4.2.2 Testauslaite

Testauslaitteen koteloksi valittiin pulpettimallinen alumiiniprofiilikotelon. Aluksi tarkoitus oli käyttää kotelon piirikortin kiinnitysuria, mutta hyvin nopeasti kävi selväksi, ettei se ole mahdollista. Kiinnitysuria käytettäessä tulisi piirikortti liu'uttaa sivultapäin paikoilleen. UI-piirikortin osalta kuitenkin etupaneelin läpi tulevat vipukytkimet ja IO-piirikortin osalta takalaidan lattakaapeliliittimet jotka tulevat takareunan läpi estävät sen. Näin ollen piirilevyjen kiinnitys tehtiin ruuveilla.

Kotelon etupaneeliin koneistettiin reiät kytkimille ja merkkivaloille sekä aukko näytölle. Takaosaan tehtiin aukot lattakaapeliliittimille, kahdelle BNC-runkoliittimelle ja kuudelle 4mm banaanirunkoliittimelle.

4.3 Elektroniikka

4.3.1 Virtalähde

Testauslaitteen vaatimusmäärittelyn mukaan laitteen tuli itse muodostaa tarvittavat jännitteet sille syötettävästä käyttöjännitteestä. Lisäksi kaikki sähköt testattavalle kortille pitää pystyä katkaisemaan yhdellä kytkimellä. Testauslaitteen käyttöjännitteeksi valittiin 24 V tasajännite.

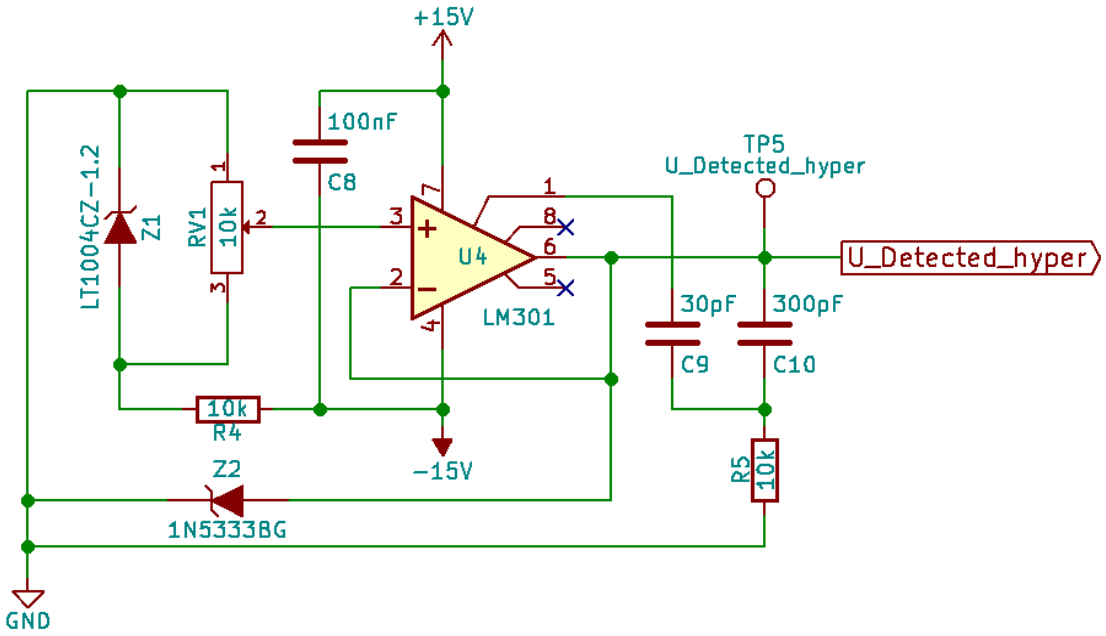
Määrittelyn mukaan syöttöön käytetään virranrajoituksella varustettua virtalähdettä, joten testauslaitteeseen ei suunniteltu erillistä ylivirtasuojausta. Myöskään ylijännitesuojausta ei tehty. Suojaus käyttöjännitteen väärää napaisuutta varten tehtiin kytkemällä normaalisti estosuunnassa oleva diodi käyttöjännitelinjojen väliin.

Testauslaitteen tarvitsemat jännitteet päätettiin muodostaa DC-DC muuntimilla. Näitä laitteeseen tuli yhteensä kolme kappaletta. Kaksi muunninta joiden ulostulojännite on 5 V ja yksi jossa on 15 V sekä -15 V ulostulot. Yksi 5 V muunnin syöttää testauslaitteen mikrokontrolleria, muunnin- ja logiikkapiirejä sekä näyttöä. Loput ovat testattavan kortin syöttöä varten. DC-DC muuntimien syöttöön asennettiin yhteismuotoinen kuristin (common mode choke) hakkurien aiheuttamien EMC-häiriöiden vähentämiseksi muuntimien datalehtien mukaisesti (THN 15-WI Series Application Note, 2012).

Testattavan kortin syötön kytkentään käytetään kahta relettä. Ensimmäistä relettä ohjataan suoraan POWER-kytkimellä ja se kytkee 5 V ja 15 V jännitteet testattavalle kortille. Toinen rele ohjaa -15 V linjaa ja sen ohjaus otetaan myös POWER-kytkimeltä, mutta se kiertää vielä -15 V FAILURE kytkimen kautta ja näin mahdollistaa -15 V vian muodostamisen.

Virtalähdeosalla muodostetaan myös ”HYPER DETECTED” signaalin tarvitsema noin -0,6 V jännite kuvion 4 mukaisesti. Jännite muodostetaan operaatiovahvistimella U4 joka on kytketty jänniteseuraajaksi (Horowitz & Winfield

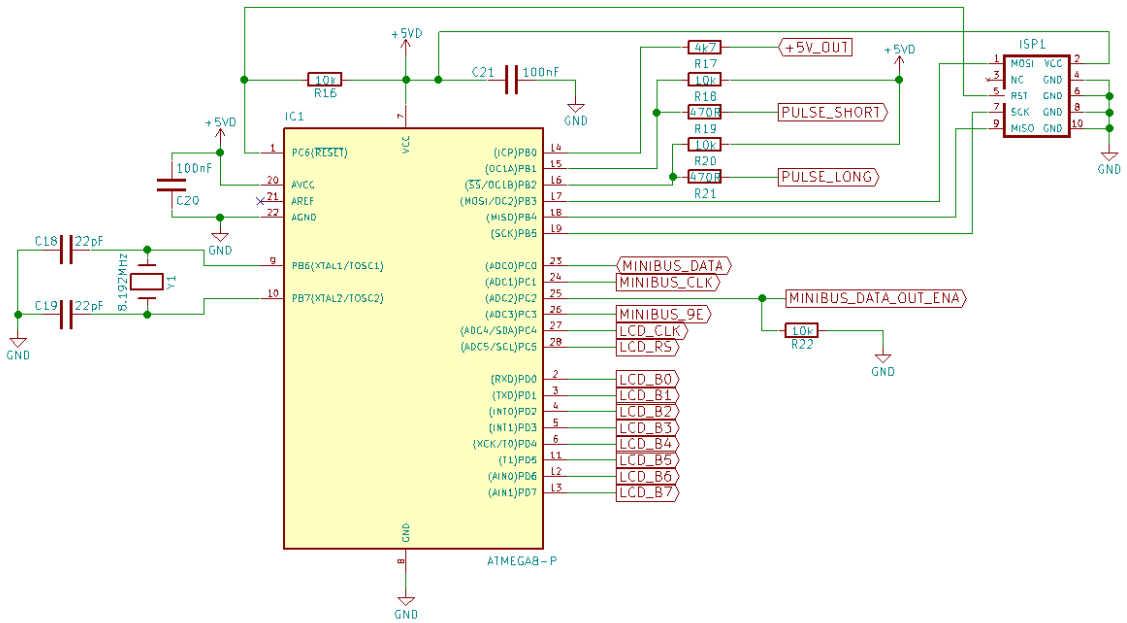
2015, 227). Jänniteseuraajan tulojännite tehdään 1,25V jännitereferenssipiirillä Z1, mistä otetaan jännitejako potentiometrillä RV1. Ulostulossa on estosuuntaan kytketty Zener-diodi estämässä vikatilanteiden aiheuttamat poikkeavat jännitteet.



Kuvio 4 ”HYPER DETECTED” jännitteen muodostus

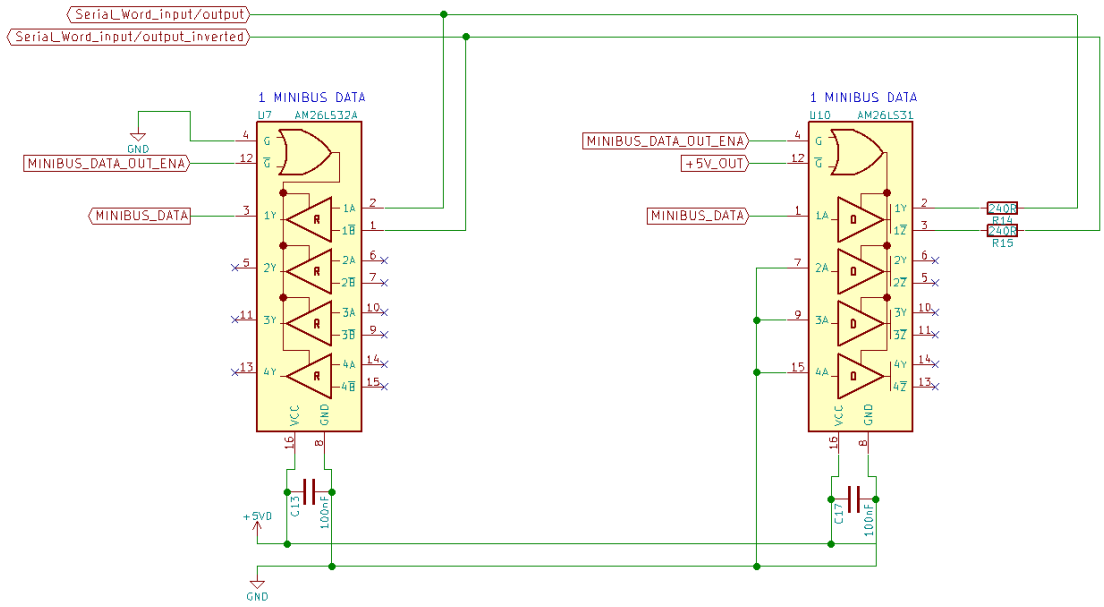
4.3.2 Mikrokontrolleri ja Minibus-väylä

Mikrokontrolleri muodostaa synkronointisignaalit, keskustelee MINIBUS-väylän kanssa sekä ohjaa etupaneelin näyttöä ja D1 merkkivaloa. Mikrokontrollerille tuodaan myös tieto onko testauslaitteen virta päällä ”+5V_OUT” jännitetietona. Lisäksi mikrokontrolleriin liittyy 10-napainen ISP-liitin jonka kautta sen ohjelmisto on mahdollista päivittää, ilman että piiriä tarvitsee irrottaa piirilevystä. Kuviossa 5 on esitetty mikrokontrollerin kytkennät.



Kuvio 5. Mikrokontrollerin kytkennät

MINIBUS-väylä on differentiaalinen, joten mikrokontrollerin signaalit pitää muuttaa oikeaan muotoon ja se tehdään RS-422 muunninpiireillä. MINIBUS-datasignaali on välillä lähtevä ja välillä tuleva signaali. Tästä syystä mikrokontrolleri ohjaa lähetin- ja vastaanotinpiirien tilaa. Ohjaus on kytketty lähetinpiirissä aktivointituloon ja vastaanotinpiirissä käänteiseen aktivointituloon. Näin ollen lähetin ja vastaanotin eivät ole yhtä aikaa aktiivisina. Mikrokontrollerin MINIBUS-datalinja on kytketty lähetin ja vastaanotinpiireihin. Datalinjan kytkentä on esitetty kuviossa 6. MINIBUS-kellosignaali muodostetaan omalla lähetinpiirillä joka on aina aktiivinen, koska kellosignaaleja lähetetään niin datan lähetyksen, kuin myös vastaanoton aikana.

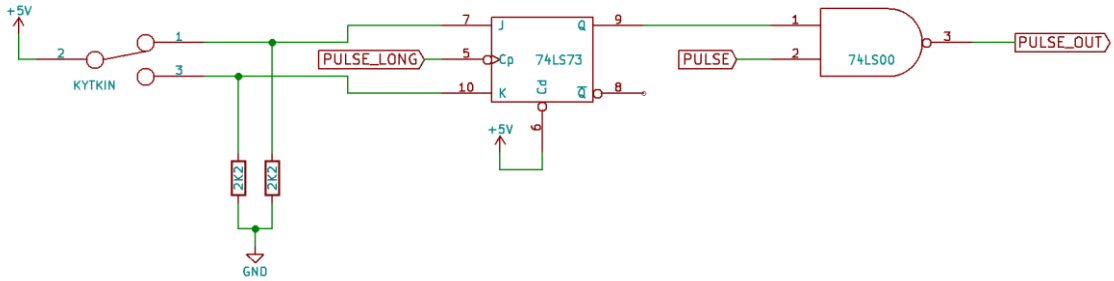


Kuvio 6. MINIBUS-datasignaalin lähetin ja vastaanotinpiirien kytkentä

4.3.3 Signaalien generointi

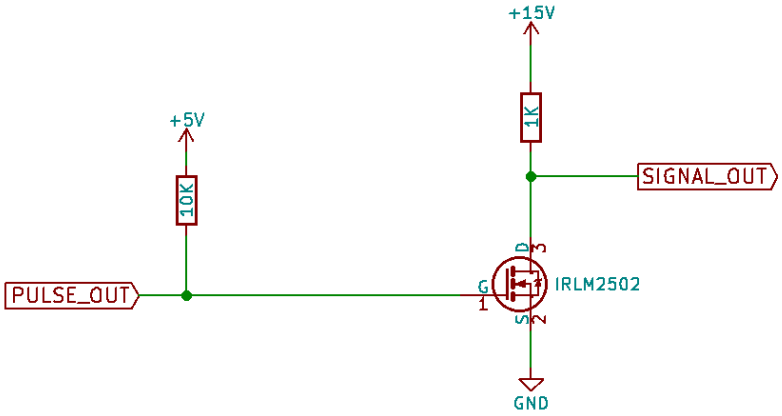
Komparaattoreille menevät signaalit muodostetaan kytkimien avulla. Osa tuloista on ylösvedetty +15 V testattavalla kortilla joten tilan vaihtaminen onnistuu kytkemällä linja kytkimellä maahan. Osa linjoista vaatii kuitenkin jännitesyötön testauslaitteelta. Näissä tapauksissa on testauslaitteessa virranrajoitusvastus mahdollisia vikatilanteita varten suojaamassa testattavaa korttia ja itse testauslaitetta.

Synkronointisignaalien kytkentä tehdään logiikkapiirien avulla joita ohjataan etupaneelin kytkimillä. Vaihtokytkimeltä tilatieto viedään JK-kiikun J ja K tuloihin. JK-kiikun kellotuloon tuodaan mikroprosessorin tuottama ”PULSE_LONG” signaali. Tällä kytkennällä estetään kytkinvärähtelyn aiheuttamat häiriöt ja lisäksi estetään tilanvaihtuminen kesken pulssin. Tämä helpottaa mahdollisten mittausten tekemistä, koska pulssit ovat aina kokonaisia. JK-kiikun ulostulosta kytkintieto viedään NAND-portin tuloon. Portin toiseen tuloon tuodaan mikrokontrollerilta tarvittava pulssimuotoinen signaali. Kuviossa 7 on esitetty esimerkki synkronointisignaalin ohjauskytkennästä.



Kuvio 7. Synkronointisignaalin ohjaus kytkimellä

Synkronointisignaalien CSY ja RSY muunnos logiikkatasoisesta signaalista +15 V tasoon tehdään kytkin FET:ien avulla. Kytkennässä N-kanavaisen FET:in hilalle tuodaan NAND-portin ulostulosta saatu signaali. Lähde on kytketty maahan ja nielu on ylösvedetty vastuksen kautta +15 V. Nyt FET:n johtaessa on ulostulossa 0 V ja FET:n ollessa johtamattomana +15 V. Kuviossa 8 on esitetty esimerkki signaalin tasomuunnoksesta FET:n avulla.

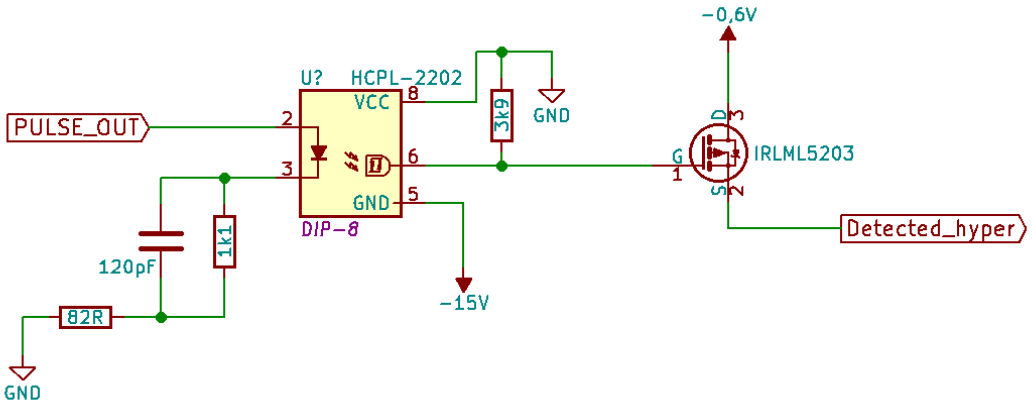


Kuvio 8. Signaalin tasomuunnos MOSFET:n avulla.

”TX synchro” signaali on RS422 tasoinen differentiaalinen signaali. Se muodostetaan RS422 lähetinpiirin avulla. Signaalin lähdössä on suojavastukset estämässä mahdolliset vauriot vikatilanteissa niin testattavalle kortille, kuin testauslaitteelle.

”Detected hyper” signaalin taso on noin -0,6 V ja sen kytkentä on hieman monimutkaisempi. Signaali tuodaan nopealle optoerottimelle, jonka käyttöjännit-

teeksi on kytketty -15 V. Optoerottimen jälkeen signaalin jännitetaso vaihtelee 0 ja -15 V välillä. Tämän jälkeen signaali viedään P-kanavaisen FET:n hilalle. FET:n nielu on kytketty ”U_DETECTED_HYPER” jännitteeseen joka on noin -0,6 V. Tämän jälkeen FET:n lähteestä on saatavissa haluttu signaali oikealla jännitetasolla. Kuviossa 9 on esitetty kyseinen jännitteenmuunnos.



Kuvio 9. Hyper detected signaalin jännitteenmuunnos.

4.3.4 Signaalien indikointi

Signaalien indikointi tehdään yksinkertaisesti johtamalla kyseinen signaali etuvastuksen kautta ledille. Signaaleille jotka lähtevät testattavalta kortilta optoerottimen kautta syötetään tarvittava jännite testauslaitteelta.

4.4 Piirilevyt

4.4.1 UI-piirilevy

Piirilevyjen suunnitteluun käytettiin KiCad nimistä avoimenlähdekoodin elektroniikkasuunnittelu ohjelmistoa. Kicad on monipuolinen ja nopeasti kehittyvä suunnitteluohjelmisto, josta löytyvät kaikki piirilevyn suunnitteluun tarvittavat ominaisuudet, sekä muun muassa mahdollisuus katsella piirilevyä 3D-näkymästä. Tämä helpottaa joissain tilanteissa mahdollisten komponenttien sijoitteluvirheiden havaitsemista. (KiCad EDA 2016)

UI-piirilevylle tuli käyttöliittymän kytkimet, ledit ja näyttö. Se asennettiin etupaneelin taakse siten että vipukytkimet ovat käytettävissä etupaneelista. UI- ja IO-piirilevyjen väliin tuli 64-napainen lattakaapeli. Piirilevyllä on vain passiivisia komponentteja, näyttöä lukuun ottamatta, joten sen suunnittelussa painopiste oli kytkimien ja ledien sijoittelulla käyttöliittymäsuunnitelman mukaisesti. Piirilevyn suunnittelussa käytin apuna tulostettua versiota piirilevystä, jonka avulla pystyin varmistamaan että komponenttien sijoittelu on järkevä ja piirilevyn kiinnitys koteloon onnistuu. Piirilevyn piirikaavio on liitteessä 2.

4.4.2 IO-piirilevy

IO-piirilevy sisältää virtalähteen, mikrokontrollerin, logiikka- ja muunnospiirit sekä liitynnät korttisovittimen lattakaapeleille ja muille liittimille. Piirilevyn suunnittelussa tuli ottaa huomioon korkeimpien komponenttien sijoittelu, siten että piirilevyt sopivat koteloon. Koska piirilevyllä on hakkurivirtalähteitä, analogisia sekä digitaalisia piirejä tuli suunnittelussa ottaa huomioon häiriöiden vähentäminen. Erityistä tarkkuutta vaati maatasojen jatkuvuus ja bypass eli ohituskondensaattoreiden asennus mahdollisimman lähelle mikropiirejä. Ohituskondensaattoreilla pienennetään virtasilmukan kokoa, jotteivät digitaalisten mikropiirien nopeat virranmuutokset aiheuta häiriöitä ympäristöön (AVR042: AVR Hardware Design Considerations 2016, 4–5).

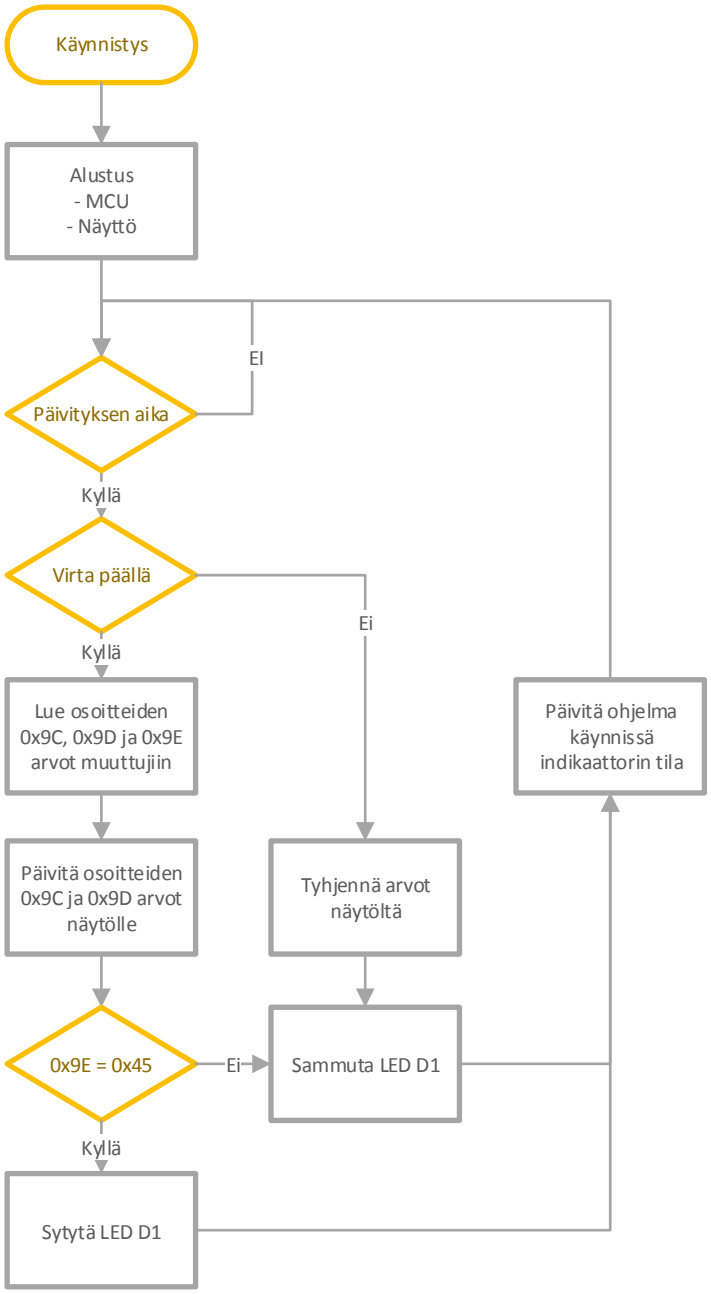
4.5 Ohjelmisto

4.5.1 Ohjelmiston rakenne

Testauslaitteen mikrokontrollerin tehtävänä on MINIBUS-väylän lukeminen, tulosten esittäminen näytöllä sekä tahdistuspulssien luominen. Testauslaitteen ohjelmiston rakenne on esitelty kuviossa 10.

Testauslaitteen ohjelma on suhteellisen yksinkertainen ja suoraviivainen rakenteeltaan. Testauslaitteen pääohjelman lähdekoodi on liitteessä 4. Ohjelmassa

käytetään AVR Libc kirjaston moduuleja ja näytön ohjaamiseen muokattua versiota Scienceprog.comin LCD kirjastosta.



Kuvio 10. Testauslaitteen ohjelman rakenne

4.5.2 Alustus

Mikrokontrollerin ohjelman käynnistyessä alustetaan käytettävät muuttujat ja vakiotekstit jotka näytetään laitteen käynnistyessä. Normaalin käynnistyksen

tapauksessa näytetään teksti ”W722 testauslaite” ja toisessa ruudussa ohjelman käännöspäivä ja -aika. Jos mikrokontrollerin uudelleenkäynnistyksen syy on jokin muu kuin ”Power-on reset”, tulostetaan näytölle uudelleenkäynnistyksen aiheuttaja. Heti ohjelman alussa luetaan mikrokontrollerin ”MCU Control and status register” (MCUCSR) arvo talteen ja nollataan se. Tästä MCUCSR rekisterin arvosta tiedetään mikä aiheutti ohjelmiston uudelleenkäynnistyksen ja se voidaan tulostaa näytölle (ATMEGA8(L) datasheet 2013, 41).

Testauslaitteessa käytetään kahta mikrokontrollerin ajastinta. Ajastin 1:n avulla luodaan ”Pulse short” ja ”Pulse long” pulssit. Koska luotavien pulssien pituus on suhteellisen lyhyt ja molempien tulee laskea alas yhtä aikaa, käytetään ajastinta tilassa 14. Tässä tilassa lähdöt on aluksi alhaalla ja kun laskuri saavuttaa halutun arvon kyseinen lähtö nousee ylös. Laskurin saavuttaessa määritetyn ”TOP” arvon se nollataan ja lähdöt ovat jälleen alhaalla. Pulssien toistoväli on 1,000 ms (1,000 kHz) ja näin ollen saadaan laskurin TOP arvoksi 0x1FFF (8191) (kaava 1). (ATMEGA8(L) datasheet 2013, 75–101)

$$TOP = \frac{f_{\mathrm{clk_I/O}}}{f_{\mathrm{prf}} \times N} - 1 \quad (1)$$

Missä

TOP	on	laskurin TOP arvo
$f_{\mathrm{clk_I/O}}$	on	Mikrokontrollerin taajuus (8,192 MHz)
f_{prf}	on	Haluttu pulssintoistotaajuus (1,000 kHz)
N	on	Laskurin “prescaler” arvo (1)

Haluttu pulssin pituus määritetään ”Output Compare Register 1 A” (OCR1A) ja ”Output Compare Register 1 B” (OCR1B) avulla. ”Pulse short” pulssi luodaan OCR1A:n avulla ja sen pituus tulee olla 1 µs, jolloin OCR1A:n arvoksi saadaan 0x0008 (8) (kaava 2). ”Pulse Long” signaali luodaan OCR1B:n avulla ja sen pituus tulee olla 9 µs, jolloin OCR1B:n arvoksi saadaan 0x004A (74) (kaava 2). (ATMEGA8(L) datasheet 2013, 75–101)

$$OCR1 = \frac{t_{pulse}}{1/f_{\mathrm{clk_I/O}}/N} \quad (2)$$

Missä

$OCR1$	on	OCR1 rekisterin arvo
$f_{\mathrm{clk_I/O}}$	on	Mikrokontrollerin taajuus (8,192 MHz)
t_{pulse}	on	Haluttu pulssin pituus (1 tai 9 µs)
N	on	Laskurin “prescaler” arvo (1)

Ajastin 2:sta käytetään päivitysvälin laskurin arvon kasvattamiseen. Ajastinta käytetään tilassa 2, jolloin ajastin nollataan kun se saavuttaa ”Output Compare Register” (OCR2) rekisteriin tallennetun arvon. Samalla tapahtuu keskeytys jossa päivitysvälin laskurin arvoa kasvatetaan yhdellä. Koska päivitysvälin laskurin arvoa halutaan kasvattaa 10 ms välein, tulee OCR2 rekisterin arvoksi 0x4F (79) (kaava 3). (ATMEGA8(L) datasheet 2013, 102–120)

$$OCR2 = \frac{f_{\mathrm{clk_I/O}}}{1/t_{\mathrm{ocr}} \times N} - 1 \tag{3}$$

Missä

$OCR2$	on	OCR2 rekisterin arvo
$f_{\mathrm{clk_I/O}}$	on	Mikrokontrollerin taajuus (8,192 MHz)
t_{ocr}	on	Haluttu keskeytyksen väli (10 ms)
N	on	Laskurin “prescaler” arvo (1024)

Seuraavaksi ohjelma alustaa MINIBUS väylän tarvitsemat mikrokontrollerin pinnit siten, että väylä on lepotilassa. Sen jälkeen alustetaan näytön tarvitsemat pinnit oikeisiin tiloihin ja tehdään näytön alustus kirjaston funktiolla. Alustuksen yhteydessä näytön muistiin siirretään omat vapaasti määriteltävät merkit. Ohjelma käyttää kahta ohjelmoitua merkkiä, joilla ilmaistaan ohjelman käynnissä olo. Indikointi tapahtuu vaihtamalla näytön ensimmäisen merkin ensimmäistä pikseliä jokaisen päivityskerran jälkeen, eli 1 sekunnin välein.

Tämän jälkeen ohjelma tulostaa näytölle tervetulotekstin riippuen onko kyseessä normaali ”Power-on reset” tai jostain muusta syystä tapahtunut uudelleenkäynnistys. Normaalin käynnistyksen yhteydessä tulostetaan ensiksi ”W772

testauslaite” teksti ja seuraavaan näyttöön ohjelman käännös päivä ja aika. Jos uudelleenkäynnistyksen syy on jokin muu, tulostetaan pelkkä uudelleenkäynnistyksen syy näytölle. Tämän jälkeen näytölle tulostetaan osoitteet ”9C:” ja ”9D:” jotka ovat näkyvissä koko ohjelman suorituksen ajan.

Alustuksen lopuksi aktivoidaan mikrokontrollerin vahtikoira toiminto, joka huolehtii mikrokontrollerin uudelleenkäynnistyksestä jos sitä ei nollata 2 sekuntiin. Lisäksi aktivoidaan keskeytykset jolloin päivitysvälilaskuri alkaa käydä.

4.5.3 Pääohjelma

Pääohjelman alussa tarkistetaan onko päivitysväli muuttuja ”timing” saavuttanut määritetyn UPDATE_INTERVAL arvon 100. Jos arvo on saavutettu, aloitetaan päivitys. Aluksi nollataan ”timing” muuttuja ja vahtikoira. Tämän jälkeen tarkistetaan onko testauslaitteen virta päällä kutsumalla funktiota getPowerStatus.

Jos getPowerStatus funktio palautti arvon tosi, aloitetaan MINIBUS-väylän osoitteiden arvojen lukeminen. Osoitteiden lukemisen jälkeen päivitetään näytölle osoitteiden 9C ja 9D arvot funktiolla printChar2Bin, joka tulostaa 8-bittisen luvun bitteinä näytölle. Lisäksi tarkistetaan onko osoitteen 9E arvo yhtä suuri kuin 0x45 (69). Jos arvo on yhtä suuri, sytytetään merkki ledi D1 ja jos ei ole sammutetaan ledi D1.

Jos getPowerStatus funktio palautti arvon epätosi, ei väylää lueta ja näytölle kirjoitetaan osoitteiden arvoksi pelkkää ”-” funktion printDash avulla, sekä ledi D1 sammutetaan. Pääohjelman lopuksi vaihdetaan vielä ”ohjelma käynnissä” indikaattorin tila.

4.5.4 MINIBUS-väylän lukeminen

MINIBUS-väylä on synkronoitu simplex sarjaväylä, eli siinä on oma linja kellosignaalille ja toinen linja joka toimii sekä datan ulos- että sisääntulona. Väylä on rakenteeltaan yksi isäntälaite ja monta orjalaitetta. Isäntälaite lähettää 9 bitti-

sen sanan, joka muodostuu aloitusbitistä ja 8 bittisestä osoitteesta jonka se haluaa lukea. Orjalaite josta kysytty osoite löytyy, siirtää kyseisen osoitteen arvon siirtorekisteriin ja valmistautuu lähettämään sen. Tämän jälkeen isäntälaitteella on rajattu aika lukea kahdeksan bittiä kellottamalla kellosignaalia ja lukemalla bitit datalinjasta. MINIBUS-väylän kirjaston lähdekoodi on liitteessä 5.

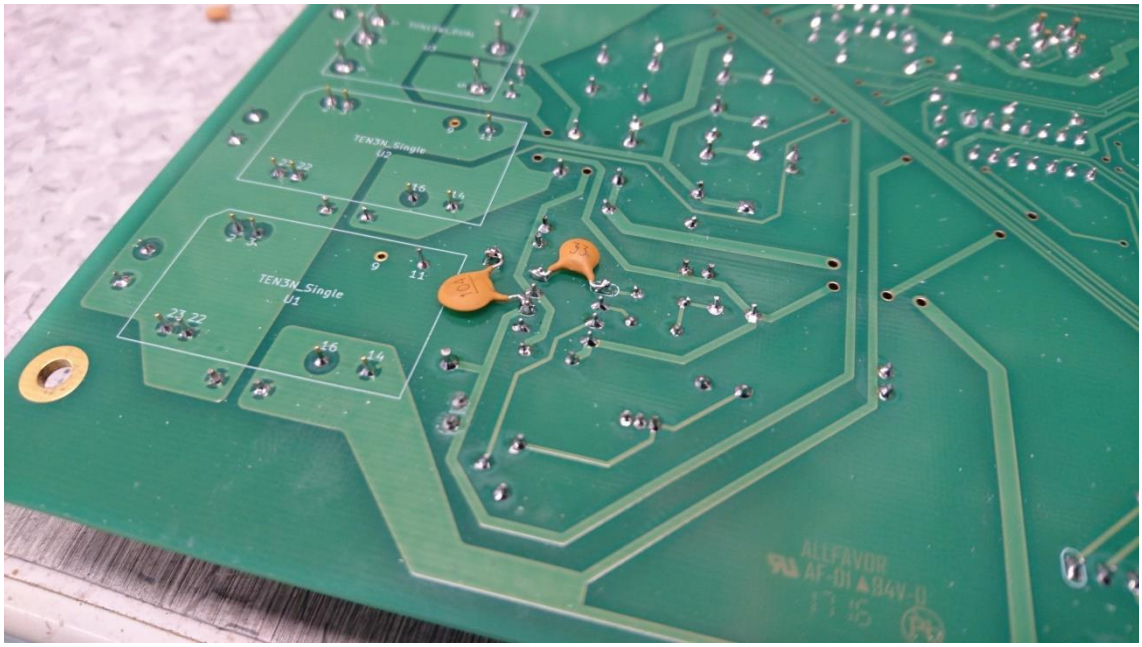
Koska MINIBUS-väylän ajoitukset ovat erittäin tiukat ja nopeus kohtuullisen suuri, ei osoitteen luonti onnistu lennosta mikrokontrollerilla. Tämän ongelman kiertämiseksi osoitteen lähetyksen dataväylän tilat muodostetaan ennen lähetystä taulukkoon. Tällöin osoitteen lähetyksessä tarvitsee lukea vain taulukosta onko bitti 1 vai 0 ja näin ollen ajoitus ei muutu osoitteen muuttuessa. Osoitteen lähetyksen jälkeen dataväylä muutetaan sisääntuloksi ja odotetaan hetki, jotta orjalaite ehtii valmistautua datan lähetykseen.

Seuraavaksi isäntälaite kellottaa 8 kellopulssia ja lukee samalla dataväylän tilan muuttujaan. Tämän jälkeen isäntälaite voi aloittaa seuraavan osoitteen lähetyksen.

5 TESTAUSLAITTEEN KOKOONPANO

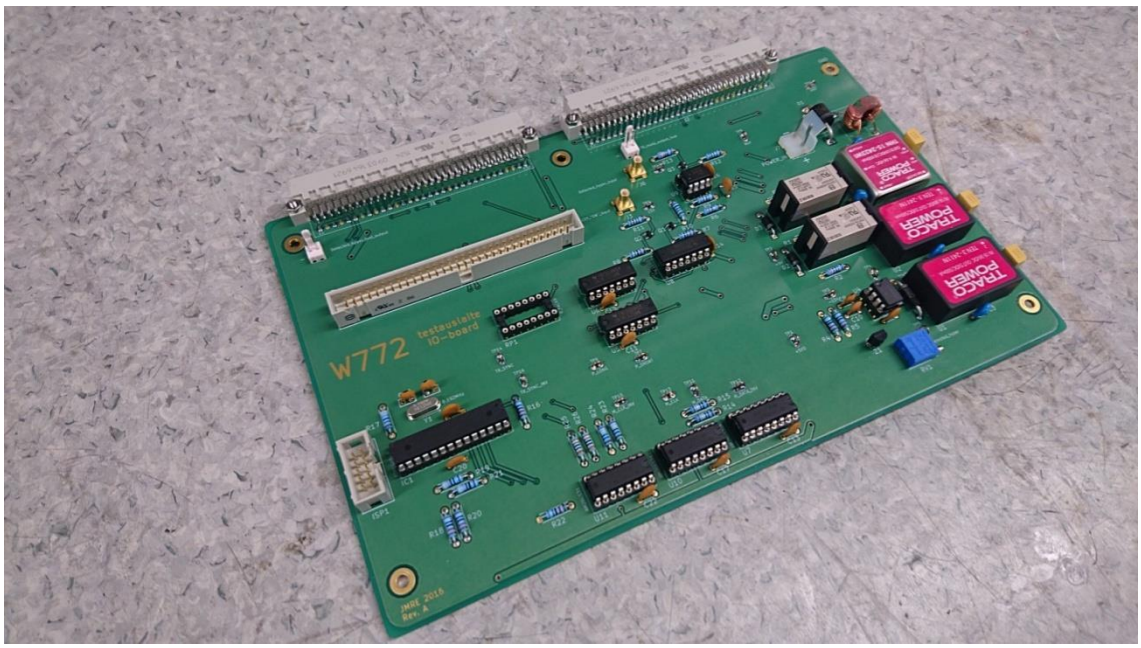
5.1 Piirilevyjen valmistus

Piirilevyt teetettiin alihankkijalla, mutta niiden kalustaminen tapahtui omin toimin. Koska projektissa ei ollut tarkoitus tehdä kuin yksi versio piirilevyistä, oli odotettavissa että niissä on joitain suunnitteluvirheitä. Kokoonpanovaiheessa havaittiin joitain suunnitteluvirheitä ja ne korjattiin erilaisin ratkaisuin. IO-levyn kokoonpanon jälkeisessä testauksessa havaittiin ongelmia ”Hyper Detected” jännitteen muodostuksessa. Syyksi paljastuivat puuttuvat kondensaattorit jännitteen muodostuksesta vastaavassa operaatiovahvistinkytkennässä. Kondensaattoreiden lisääminen onnistui kuitenkin helposti piirilevyn alapuolelle (kuva 1).



Kuva 1. IO-levyn operaatiovahvistinkytkennän korjaus

Tässä vaiheessa kokeiltiin myös mikrokontrollerin ohjelmiston lataamista ja siinä havaittiinkin seuraava ongelma. Mikrokontrollerin sulakkeiden ohjelmoinnin jälkeen se ei enää käynnistynyt. Hyvin nopeasti huomattiin että mikrokontrollerin kiteen kondensaattorit ovat 22 nF, eikä 22 pF kuten pitäisi olla. Kondensaattoreiden vaihdon jälkeen IO-levy toimi normaalisti ja oli valmiina testauslaitteen kokoonpanoa varten (kuva 2).



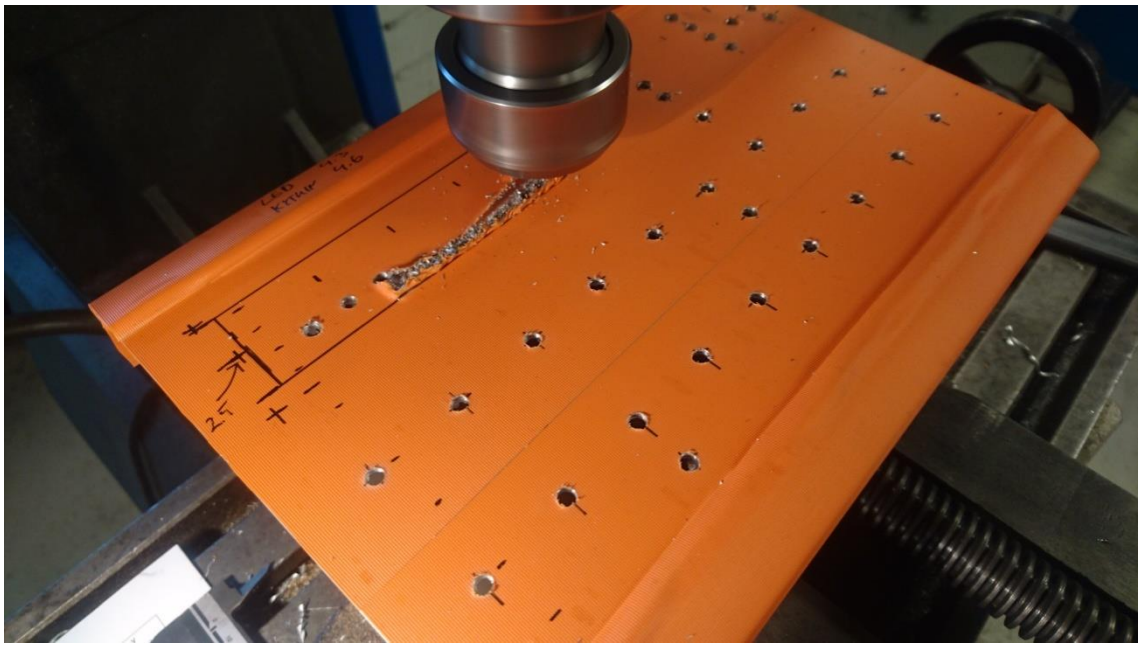
Kuva 2. Valmis IO-levy

UI-levyn kokoonpano oli huomattavasti yksinkertaisempi tehtävä kuin IO-levyn kokoonpano. UI-levyn testauksen yhteydessä ei ilmennyt ongelmia ja näin UI-levykin oli valmiina odottamassa testauslaitteen kokoonpanoa (kuva 3).

Kuva 3. Valmis UI-levy

5.2 Koteloinnin koneistus

Testauslaitteen koteloon koneistettiin reiät kytkimiä, merkkivaloja, näyttöä ja liittimiä varten. Koneistus suoritettiin manuaalijyrsinkoneella. Etupaneelin reikien sijainnit saatiin piirilevysuunnitteluohjelmistosta, joten niiden asemointi onnistui helposti jyrsinkoneella (kuva 4). Myös korttisovittimen koteloon tuli koneistaa reiät lattakaapeliliittimiä varten.



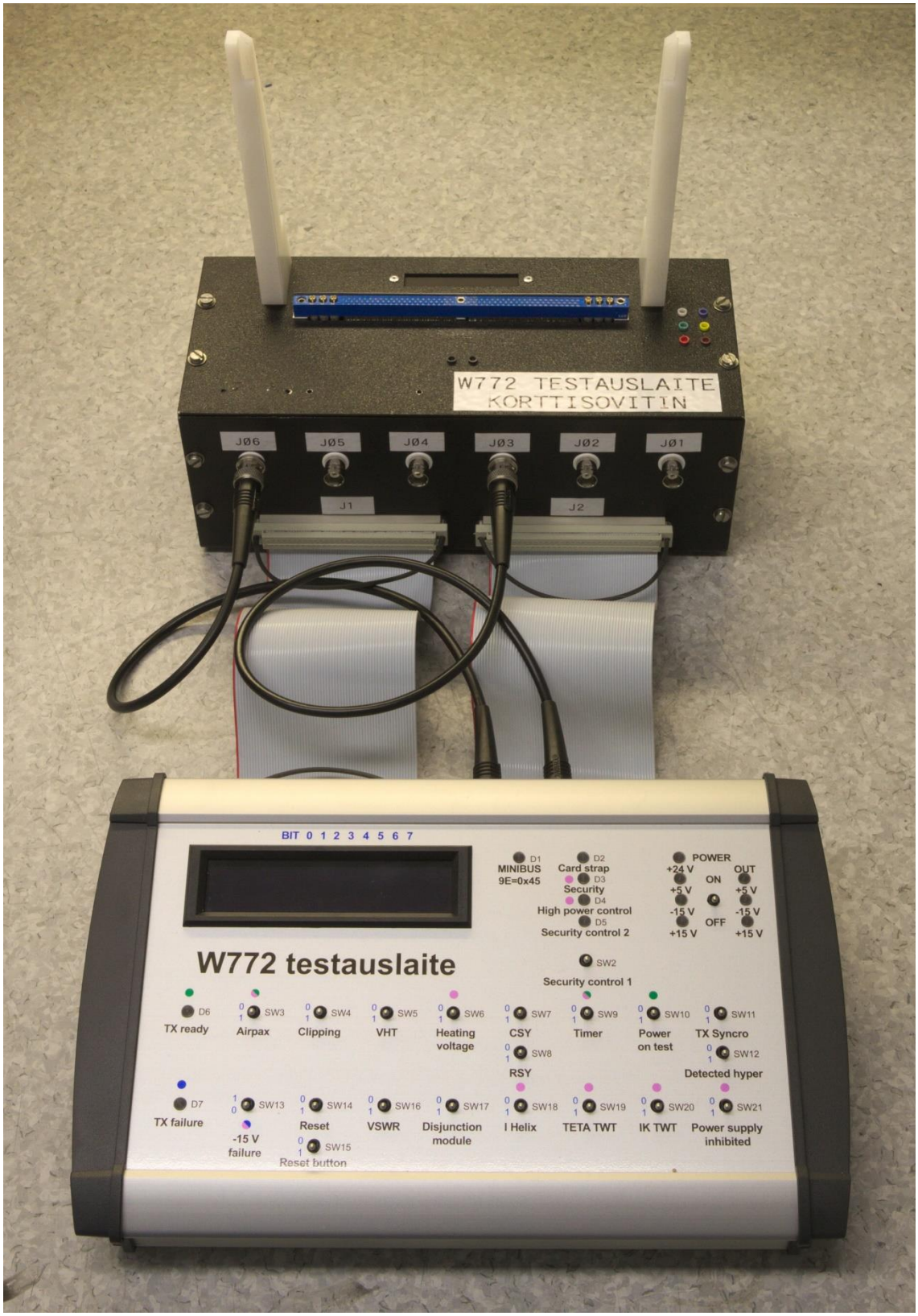
Kuva 4. Testauslaitteen etupaneelin koneistus

5.3 Testauslaitteen kokoonpano

Testauslaitteen kokoonpanovaiheessa asennettiin piirikortit koteloon, valmistettiin tarvittavat lattakaapelit piirikorttien sekä testauslaitteen ja korttisovittimen väliin, tehtiin tarvittavat kaapelit IO-kortin ja testauslaitteen liittimien välille. Lisäksi liitettiin kiertoliitostekniikalla korttisovittimen korttiliittimen ja lattakaapeliliittimien väliset johtimet (kuva 5).

Kuva 5. Korttisovittimen korttiliittimen ja lattakaapeliliittimien välinen johdotus

Tässä vaiheessa asennettiin myös etupaneelin tarra ja tehtiin loput merkinnät liittimiin ja kaapeleihin. Testauslaite oli ensimmäistä kertaa valmiina ja sen testaus voitiin aloittaa (kuva 6).



Kuva 6. Testauslaite kokoonpantuna

5.4 Testauslaitteen testaus

Testauslaitteen testaus aloitettiin varmistamalla että kaikki johdotukset ovat oikein. Tämä tehtiin mittaamalla piirikortin liittimestä, että ohjattavat signaalit tulevat oikeisiin nastoihin ja erityisesti käyttöjännitteet ovat oikein. Tässä vaiheessa ei havaittu ongelmia, joten testattava kortti voitiin kytkeä testauslaitteeseen ja aloittaa toiminnan testaus.

Testauslaitteen toiminnan testauksessa, virran päälle kytkennän jälkeen voitiin havaita MINIBUS-väylän lukemisen toimivan. Merkkivalo D1 syttyi ja näytölle tuli kortilta luetut sanat näkyviin. Seuraavaksi testattiin kaikkien kytkimien toiminta ja havaittiin että kytkimet SW6 ja SW9 toimivat väärinpäin, eli kytkimen ollessa yläasennossa väylästä luettiin 1 vaikka vastaus olisi pitänyt olla 0. Dokumentaation tarkastuksen jälkeen havaittiin virhe suunnittelussa ja kytkimien toiminta olikin väärinpäin. Nämä korjattiin katkaisemalla kytkimelle menevät piirilevyn johtimet ja kytkemällä ne kytkentälangalla kytkimien vastakkaisiin jalkoihin.

Seuraava havaittu ongelma oli, ettei merkkivalo D7 syttynyt missään vaiheessa. Ensimmäiseksi varmistettiin ledin toiminta kytkemällä piirikortin liittimen nasta P1-40 maihin ja näin ollen tiedettiin, ettei vika ole testauslaitteen kytkennässä. Seuraavaksi mitattiin testattavan kortin puolelta lähtevän signaalin toimivuus ja yllätykseksi optoerottimen lähtö toimi normaalisti. Seuraavaksi tutkittiin piirikorttia tarkemmin ja havaittiin, että testattavan kortin dokumentaatiossa on virhe ja kyseinen signaali onkin liittimen nastassa P1-48. Koska virheellistä nastaa P1-40 ei ollut kytketty piirikortilla mihinkään, voitiin korjaus tehdä kytkemällä P1-40 ja P1-48 linjat yhteen testauslaitteessa kytkentälangan avulla.

Seuraavaksi havaittiin, ettei lähetyspulssin ilmaisua tapahdu vaikka kytkimet SW11 ja SW12 ovat yläasennossa. Vikaa alettiin selvittää varmistamalla, että testattavalla kortilla tapahtuu oikeat tilamuutokset kytkimiä käännettäessä. Ensimmäiseksi havaittiin, ettei TX-synchro signaali toiminut oikein ja syyksi paljastui signaalin väärä polariteetti. Tämä korjattiin kytkemällä testauslaitteen vas-

tukset R25 ja R26 siten, että käänteinen ja ei-käänteinen signaali menevät ristiin. Tämäkään ei vielä riittänyt korjaamaan vikaa ja vianetsintää jatkettiin ”Detected_hyper” signaalin toiminnan tutkimisella. Oskilloskoopilla signaalia tutkittaessa havaittiin sen yliampuvan kytkentä ja katkaisuhetkellä. Ongelma korjaantui kun jännitelinjaan, mistä signaali muodostetaan, lisättiin vielä yksi kondensaattori heti kytkin-FETin läheisyyteen. Sekä toinen kondensaattori ”Detected_hyper” signaalin ja maan välille. Tämänkään jälkeen tilatieto ei muuttunut, kuten pitäisi ja lisämittausten jälkeen havaittiin, että 8 µs pituinen pulssi ei ollut riittävän pitkä. Pulssin pituus kasvatettiin mikrokontrollerin ohjelmaa muuttamalla 9 µs ja tilatieto alkoi muuttua suunnitellusti.

Viimeinen poikkeavuus havaittiin testauslaitteen käyttökoulutuksen yhteydessä. Virallisessa testipenkissä vialliseksi todettu kortti toimi normaalisti testauslaitteella. Dokumentaatiota tutkimalla havaittiin, ettei testauslaitteessa ollut päätevastusta MINIBUS-väylän datalinjassa ja näin ollen se ei kuormittanut testattavan kortin lähetinpiiriä riittävästi. Kun datalinjaan lisättiin 150 Ω vastus testipisteiden TP11 ja TP12 välille tuli viallisen kortin vika esille myös testauslaitteella.

6 POHDINTA

Opinnäytetyön tavoitteena oli kehittää ja toteuttaa piirikortin testauslaite. Työ oli erittäin monipuolinen sisältäen useita eri kehittämisprojektin osa-alueita. Työssä jouduin ratkaisemaan elektroniikka, mekaniikka kuin myös ohjelmistopuolen ongelmia. Osaan asioista olin saanut tietoa nykyisistä opinnoista, osaan aikaisemmista opinnoista, osaan omista elektroniikkaharrastuksista ja paljon tietoa oli kertynyt myös työelämästä. Työ oli kuitenkin ensimmäinen näin laaja-alainen elektroniikkaprojekti ja moni asia suunnitteluvaiheessa tuli tehtyä vaikeimman kautta. Projekti oli kuitenkin läpivietävissä ja lopputuloksena olen kehittynyt laaja-alaisesti elektroniikkaprojektin suunnittelutyössä.

Haasteita työssä aiheutti testattavan piirikortin dokumentaation laatu, osa merkinnöistä oli epäselviä ja niitä joutui varmistamaan eri lähteistä ja suoraan piirikortista tarkistamalla. Lisäksi signaalien nimitykset vaihtelivat dokumentista toiseen mikä toi lisähaasteita suunnitteluun.

Työn aikataulu venyi työkiireiden vuoksi, mutta lopputuloksena valmistui toimiva testauslaite piirikorttien vikakorjauksen avuksi. Testauslaitteen avulla on tehty jo ensimmäiset vianpaikannukset ja piirikortti korjaukset.

LÄHTEET

ATMEGA8(L) datasheet. 2013. Viitattu 1.11.2016. http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf.

AVR042: AVR Hardware Design Considerations. 2016. Viitattu 1.11.2016. http://www.atmel.com/Images/Atmel-2521-AVR-Hardware-Design-Considerations_ApplicationNote_AVR042.pdf

Horowitz, P. & Winfield, H. 2015. The Art of Electronics third edition. 6. Painos. New York: Cambridge University Press.

KiCad EDA, 2016, Viitattu 1.11.2016. http://kicad-pcb.org/.

THN 15-WI Series Application Note, 2012. Viitattu 1.11.2016. http://www.tracopower.com/products/thn15wi-application.pdf.

LIITTEET

Liite 1. Vaatimusmäärittely
Liite 2. UI-piirilevyn piirikaavio
Liite 3. IO-piirilevyn piirikaaviot
Liite 4. Testauslaitteen ohjelman pääohjelman lähdekoodi
Liite 5. Testauslaitteen ohjelman MINIBUS kirjaston lähdekoodi
Liite 6. Testauslaitteen ohjelman LCD-näytön kirjaston lähdekoodi

Liite 1 1(2): Vaatimusmäärittely

PCB W772 (PN 45091314) Maintenance And Security Card- kortin testauslaite (versio 1.0/7.8.2014JaLi)

Kortin tekninen spesifikaatio löytyy ohjekirjasta ITOHJ90 PCB TFH W772 THOMS-45091314 (nimike 1427-407-7471).

Sähköiset vaatimukset:

1. Testauslaitteessa on liittimet, joihin teholähteet kytketään. Testauslaite kytkee liittimiltä käyttösähköt kortille. Oletus on, että testauslaitteeseen kytketään teholähteet, joissa on virranrajoitusmahdollisuus.
2. Testauslaitteessa on pääkytkin, josta voidaan kytkeä päälle ja pois kaikki kortille tulevat käyttösähköt.
3. Testauslaite muodostaa jokaiselle alla luetellulle signaalille jännitearvot niin, että kortilla olevat komparaattorit voidaan testata. Tämä tarkoittaa, että komparaattorin lähtö saadaan sekä ylä- että alatilaan käyttäjän toimesta (esim. kytkimestä vaihtaen tai potentiometriä kääntämällä).

    a. P1_6 Security Control 1
    b. P1_7 TWT temp (teta)
    c. P1_9 Heating voltage
    d. P1_12 Power supply inhibited
    e. P1_23 Timer “ON” input (TWT preheat time)
    f. P1_28 Airpax (CB on/off)
    g. P1_30 Reset button
    h. P1_62 Power “ON” test input
    i. P1_78 VHT “ON”
    j. P1_87 Clipping “ON” test
    k. P1_95 I helix “ON” test
    l. P1_98 Reset input
    m. P1_100 VSWR “ON” test
    n. P1_102 Ik TWT “ON”
    o. P1_104 Disjunction module

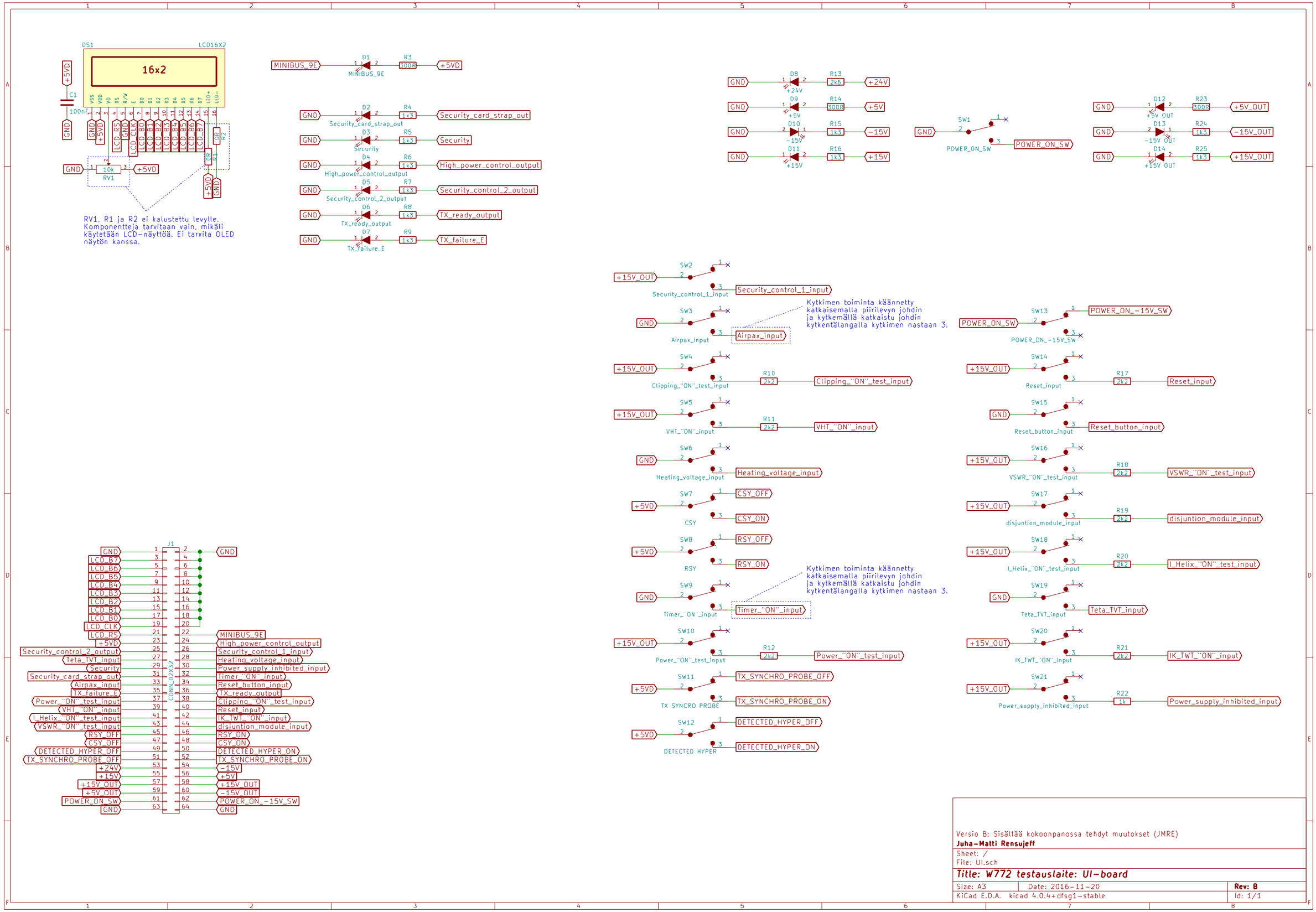
4. Testauslaite muodostaa TX-Synchro signaalin kortille (RS422), pinnit P1_50 ja P1_54.

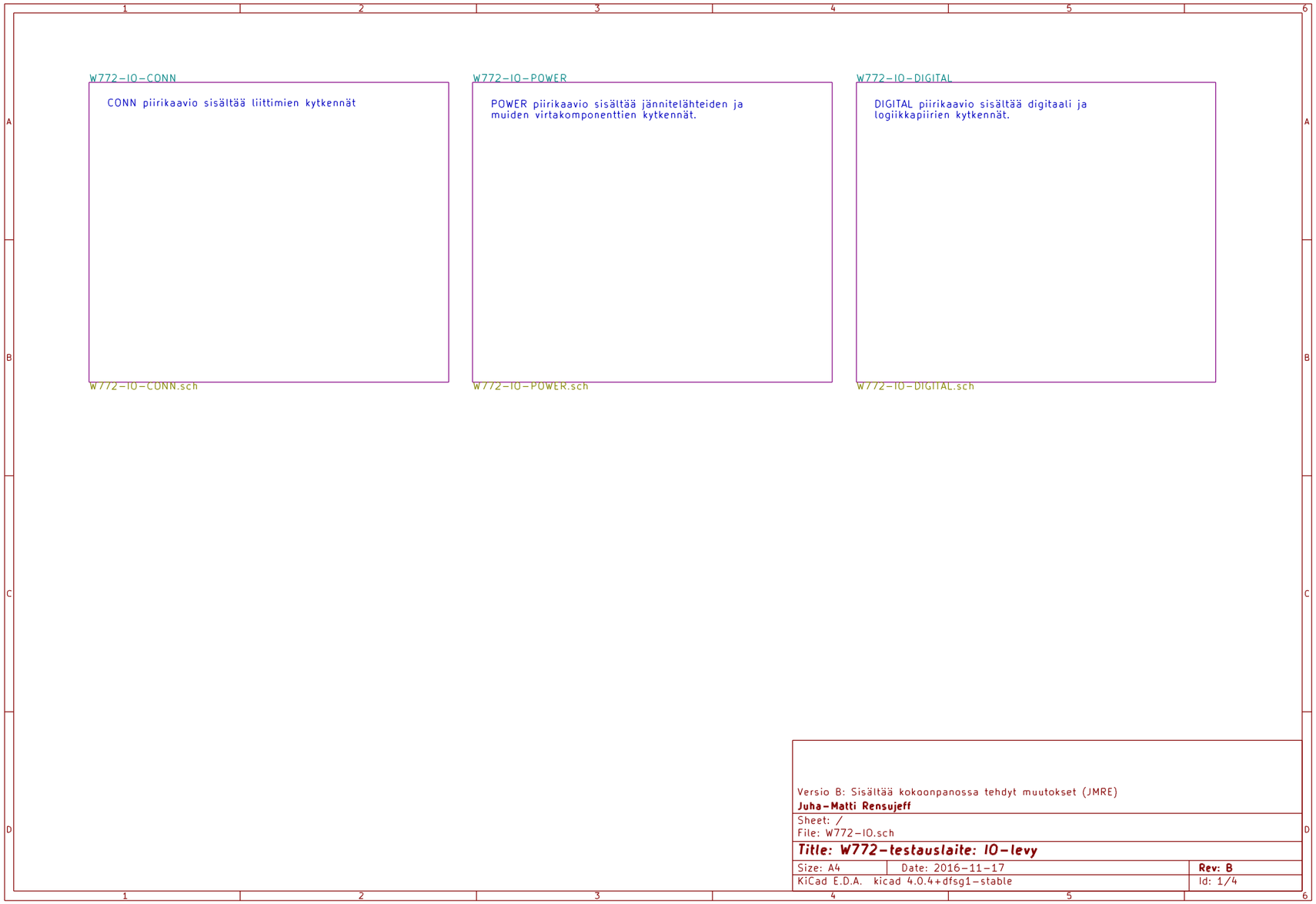
5. Testauslaite muodostaa RSY-signaalin, pinnit P1_124 ja P1_125
6. Pinniin P1_6 (Security control 2 input) testilaite syöttää käyttäjän ohjaamana +15V tai 0V.
7. Testauslaite muodostaa Presence SYC-signaalin P1_*03
8. Testauslaite muodostaa ”Detected Hyper”-signaalin P1_*06
9. Testauslaite pystyy kommunikoimaan W772 kortin kanssa Minibus-väylän kautta.
10. Seuraaviin kortilta lähteviin signaaleihin pitää pystyä liittymään seuraavasti:
    a. Bnc-liitin P1_01 (CSY TEST)
    b. Bnc-liitin P1_02 (RSY TEST)
    c. Bnc-liitin P1_04 (FR transmitter output)
    d. Naparuuvi (banaani-liitin naaras) P1_1 (Detected Hyper Test)
    e. Naparuuvi (banaani-liitin naaras) P1_10 (Security)
    f. Naparuuvi (banaani-liitin naaras) P1_75 (TX ready output)

Käyttäjävaatimukset:

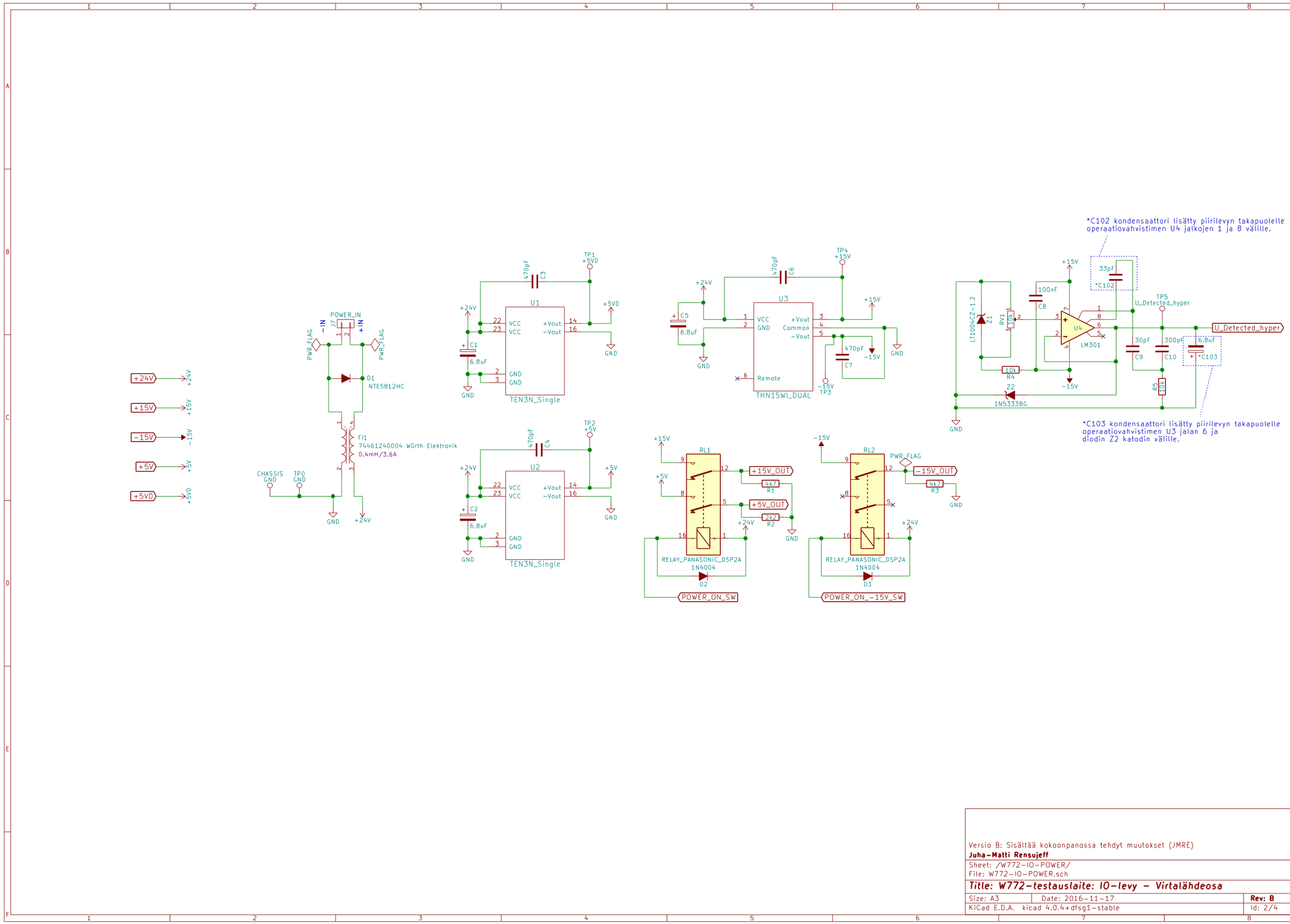
1. Laitteessa on näyttö, josta nähdään Minibus-väylän kautta luettu tieto ja mistä osoitteesta data on luettu. Laite lukee 8-bittiset sanat osoitteista 9E, 9C ja 9D. Vain osoitteiden 9C ja 9D datat näytetään bittimuodossa. Jos osoitteen 9E data on yhtä suuri kuin 45_{hex} (69_{des}) sytytetään vihreä merkkivalo laitteen käyttöpaneelissa. Testauslaite lukee osoitteita noin 1 sekunnin välein.
2. Laite valvoo onko lenkki P1_2 ja P1_3 oikosulussa vai ei (High power control input & output). Kortilla on rele, joka vetäneenä laittaa ko. signaalit yhteen. Käyttäjälle tieto lenkin tilasta esim. ledillä.
3. Testauslaitteessa on indikointi pinnien P1_21 ja P1_22 (Security card strap) välisen oikosulkulenkin toiminnan selvittämiseen (lenkki auki tai kiinni). Esim. ledi.
4. Indikointi toimiiko opto oikein signaalissa P1_69 ja P1_72 (TX ready input & output), esim ledi.
5. Indikointi toimiiko opto oikein signaalissa P1_40 ja P1_51 (Tx failure), esim. ledi.

Liite 2: UI-piirilevyn piirikaavio




W772-IO-CONN
CONN piirikaavio sisältää liittimien kytkennät
W772-IO-CONN.sch
W772-IO-POWER
POWER piirikaavio sisältää jännitelähteiden ja
muiden virtakomponenttien kytkennät.
W772-IO-POWER.sch
W772-IO-DIGITAL
DIGITAL piirikaavio sisältää digitaali ja
logiikkapiirien kytkennät.
W772-IO-DIGITAL.sch
Versio B: Sisältää kokoonpanossa tehdyt muutokset (JMRE)
Juha-Matti Rensujeff
Sheet: /
File: W772-IO.sch
Title: W772-testauslaite: IO-levy
Size: A4
Date: 2016-11-17
Rev: B
KiCad E.D.A. kicad 4.0.4+dfsg1-stable
Id: 1/4

Liite 3 2 (4): IO-piirilevyn piirikaaviot



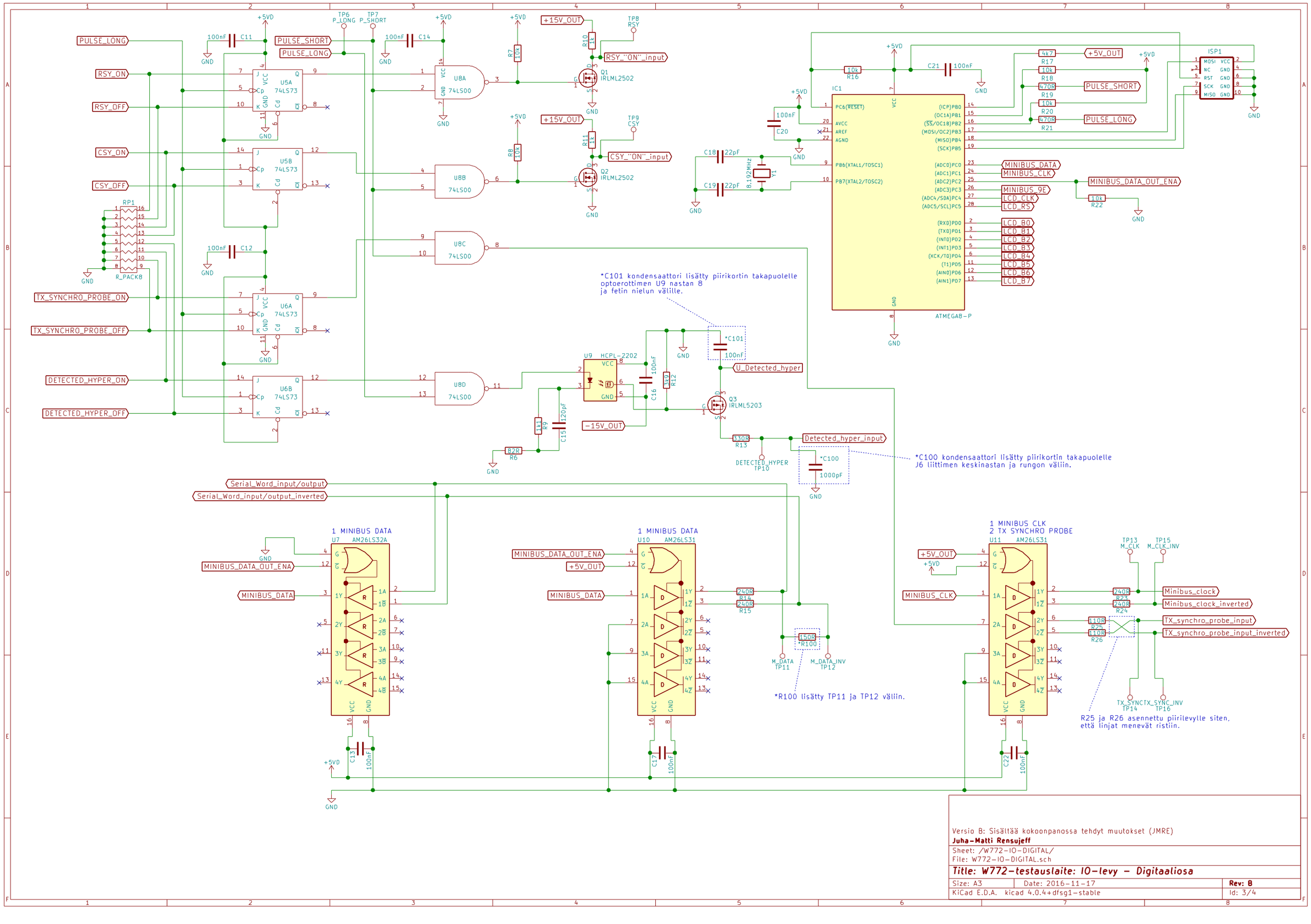
Versio B: Sisältää kokoonpanossa tehdyt muutokset (JMRE)
Juha-Matti Rensujeff

Sheet: /W772-IO-POWER/
File: W772-IO-POWER.sch

Title: W772-testauslaite: IO-levy - Virtalähdeosa

Size: A3	Date: 2016-11-17	**Rev: B**
KiCad E.D.A. kicad 4.0.4+dfsg1-stable		Id: 2/4

Liite 3 3 (4): IO-piirilevyn piirikaaviot



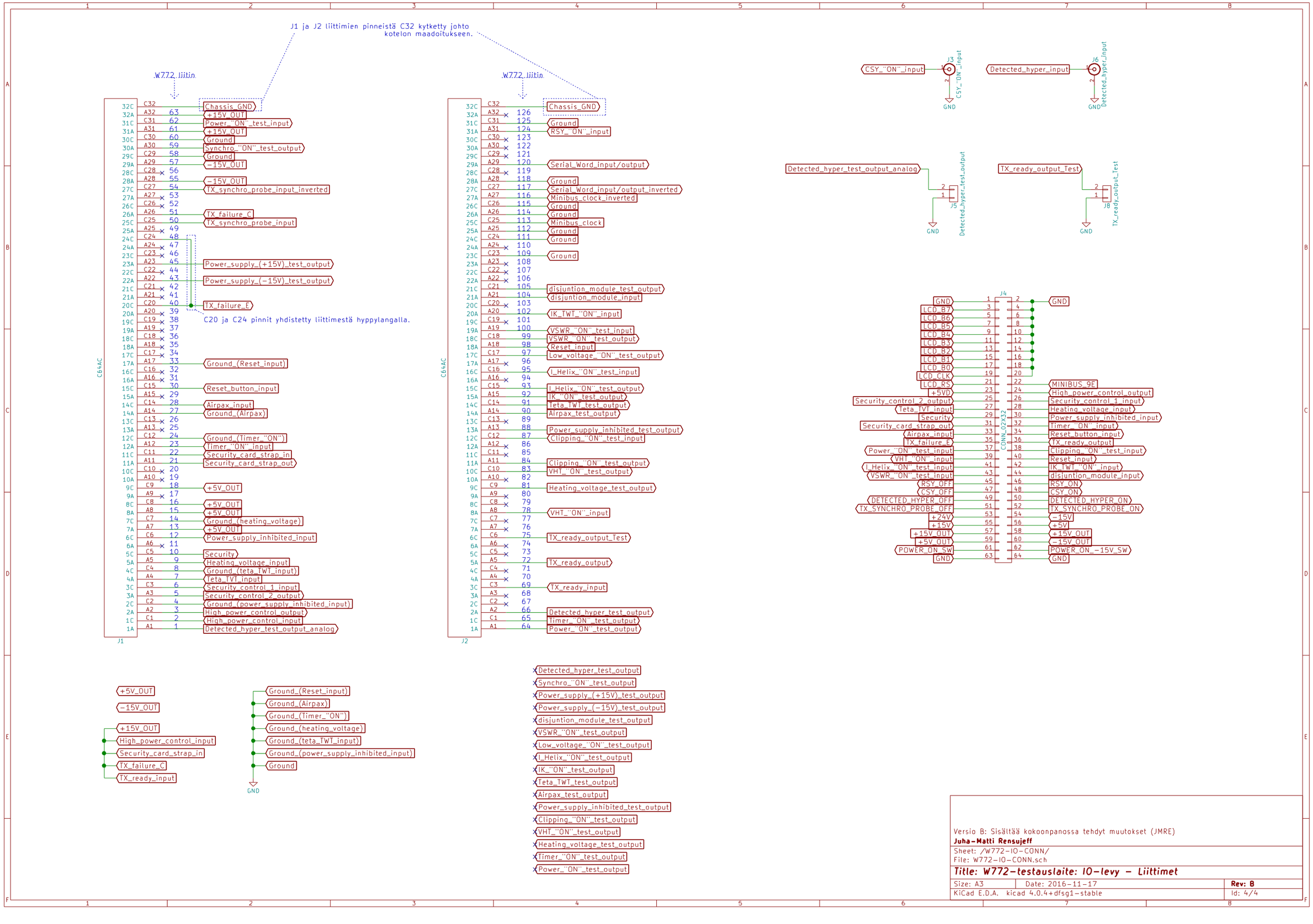
Versio B: Sisältää kokoonpanossa tehdyt muutokset (JMRE)
Juha-Matti Rensujeff

Sheet: /W772-IO-DIGITAL/
File: W772-IO-DIGITAL.sch

Title: W772-testauslaite: IO-levy - Digitaaliosa

Size: A3	Date: 2016-11-17	**Rev: B**
KiCad E.D.A. kicad 4.0.4+dfsg1-stable		Id: 3/4

Liite 3 4 (4): IO-piirilevyn piirikaaviot



Versio B: Sisältää kokoonpanossa tehdyt muutokset (JMRE)
Juha-Matti Rensujeff

Sheet: /W772-IO-CONN/
File: W772-IO-CONN.sch

Title: W772-testauslaite: IO-levy - Liittimet

Size: A3	Date: 2016-11-17	**Rev: B**
KiCad E.D.A. kicad 4.0.4+dfsg1-stable		Id: 4/4

```
//*****************************************************************************
//
// File Name     : 'main.c'
// Title         : W772 testauslaiteen ohjelma
// Author        : Juha-Matti Rensujeff
// Created       : 2016-05-01
// Version       : 1.0
// Target MCU    : Atmel AVR series
//
//*****************************************************************************

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include "lcd_lib.h"
#include "minibus.h"


#define UPDATE_INTERVAL    100   // Päivitysväli * 10 ms

// Funktioiden esittely
void printChar2Bin(uint8_t);      // Tulostaa näytölle binäärilukuna annetun parametrin
void printDash(void);             // Tulostaa näytölle kahdeksan viivaa (-)
uint8_t getPowerStatus(void);     // Palauttaa 1 jos virta on päällä, 0 jos ei

// Normaali käynnistyksen käynnistys teksti (sivu 1)
const uint8_t LCDwelcomeln1[] PROGMEM="      W772\0";
const uint8_t LCDwelcomeln2[] PROGMEM="  testauslaite\0";

// Normaali käynnistyksen käynnistys teksti (sivu 2)
      // Ohjelman käännösaika
const uint8_t LCDverln1[] PROGMEM="SW ver  "__TIME__ "\0";
      // Ohjelman käännöspäivä
const uint8_t LCDverln2[] PROGMEM="     "__DATE__ "\0";

// Ei normaalin käynnistyksen käynnistysteksti
      // Ulkoinen reset
const uint8_t LCDresetExt[] PROGMEM="   EXT reset\0";

      // Mikrokontrollerin alijännitevahti
const uint8_t LCDresetBO[] PROGMEM="Brown-out reset\0";

       // Mikrokontrollerin vahtikoira
const uint8_t LCDresetWD[] PROGMEM=" Watchdog reset\0";


volatile uint8_t timing = 0;          // muuttujalla lasketaan 10 ms välein
                                      // tulevia keskeytyksiä
```

```
int main(void)
{
	// Haetaan resetin syy ja nollataan rekisteri
	uint8_t resetFlags = MCUCSR;
	MCUCSR = 0;

	// Alustetaan muuttujat
	uint8_t result9C = 0x00;
	uint8_t result9D = 0X00;
	uint8_t result9E = 0X00;
	uint8_t running = 0;


	// Porttien alustukset
	PORTC|= 1<<3;				// Minibus 9E led poispäältä
	DDRC|= 1<<3;				// Minibus 9E led portti ulostuloksi




	// Timer 1 alustukset (Pulssien generointi)
	DDRB = 0x06;				// Asetetaan PB1 and PB2 ulostuloiksi
						// (Pulse short ja Pulse long)

	ICR1 = 0x1FFF;				// PWM top value = 1 ms

	OCR1A = 0x0007;				// OC1A pulssin pituus = 1 us
	OCR1B = 0x004A;				// OC1B pulssin pituus = 9 us

	// WGM mode 14 (Fast PWM, TOP = ICR1)
	// OC1A and OC1B ei invertoivassa tilassa
	TCCR1A = (1<<WGM11)|(1<<COM1A1)|(1<<COM1B1);

	// (set on compare match, clear on BOTTOM)
	// WGM mode 14 (Fast PWM, TOP = ICR1)
	// Prescaler 1
	TCCR1B = (1<<WGM12)|(1<<WGM13)|(1<<CS10);


	// Timer 2 alustukset (Päivitysvälin laskuri)
	OCR2 = 0x4F;				// CTC = 10 ms
	TIMSK |= (1<<OCIE2);			// Output compare keskeytys päälle
	TCCR2 = (1<<WGM21)			// WGM mode 2 (CTC, TOP = OCR2)
		|(1<<CS22)|(1<<CS21)|(1<<CS20);	// Prescaler 1024


	// Minibus alustus
	MinibusInit();

	// Näytön alustus
	LCDinit();
```

```
// Jos normaali käynnistys
if (resetFlags&0x01){
		// Tulostetaan tervetuloteksti
		CopyStringtoLCD(LCDwelcomeln1, 0, 0);
		CopyStringtoLCD(LCDwelcomeln2, 0, 1);
		_delay_ms(2000);

		// Tulostetaan build date
		CopyStringtoLCD(LCDverln1, 0, 0);
		CopyStringtoLCD(LCDverln2, 0, 1);
		_delay_ms(2000);
}
else if (resetFlags&0x02){	// External reset
		CopyStringtoLCD(LCDresetExt, 0, 1);
		_delay_ms(2000);
}
else if (resetFlags&0x04){	// Brown-out reset
		CopyStringtoLCD(LCDresetBO, 0, 1);
		_delay_ms(2000);
}
else if (resetFlags&0x08){	// Watchdog reset
		CopyStringtoLCD(LCDresetWD, 0, 1);
		_delay_ms(2000);
}

LCDclr();

// Tulostetaan osoitteet näytölle
LCDGotoXY(2, 0);
LCDsendChar('9');
LCDsendChar('C');
LCDsendChar(':');

LCDGotoXY(2, 1);
LCDsendChar('9');
LCDsendChar('D');
LCDsendChar(':');

// Aktivoidaan vahtikoira (2s)
wdt_enable(WDTO_2S);

// Aktivoidaan keskeytykset
sei();


for(;;) {				// Main looppi
	// Jos timing yhtäsuuri kuin UPDATE_INTERVAL suoritettaan päivitys
	if (timing >= UPDATE_INTERVAL) {
		timing = 0;
		wdt_reset();		// Nollataan vahtikoira
```

```
            // Virta päällä, luetaan osoitteet väylästä ja tulostetaan tulokset.
            if (getPowerStatus()) {
                    result9C = MinibusRead(0x9C);
                    result9D = MinibusRead(0x9D);
                    result9E = MinibusRead(0x9E);

                    LCDGotoXY(6, 0);
                    printChar2Bin(result9C);

                    LCDGotoXY(6, 1);
                    printChar2Bin(result9D);

                    if (result9E == 0x45){
                        PORTC&=~(1<<3);
                    } else {
                        PORTC|= 1<<3;
                    }
                } else {                        // Testattavan kortin virta pois,
                                                // tulostetaan - ja
                                                // sammutetaan D1 merkkivalo.
                LCDGotoXY(6, 0);
                    printDash();
                    LCDGotoXY(6, 1);
                    printDash();
                PORTC|= 1<<3;
                }

                // Vaihdetaan ohjelmakäynnissä indikaattorin tila
                if (running) {
                    running = 0;
                } else {
                    running = 1;
                }
                LCDGotoXY(0, 0);
                LCDsendChar(running);

            }
        }
        return 0;                       // Tänne ei päästä koskaan
}

// Funtio tulostaa annetun merkin binääri muodossa (Vähiten merkitsevä bitti ensin).
void printChar2Bin(uint8_t value) {
        for (int8_t x = 0; x<=7; x++) {
            if(bit_is_set(value, x)) {
                LCDsendChar('1');
            }else {
                LCDsendChar('O');
            }
        }
        return;
}
```

Liite 4 5(5): Testauslaitteen ohjelman pääohjelman lähdekoodi

```
// Funktio tulostaa näytölle 8 "-" merkkiä
void printDash() {
	for (int8_t x = 0; x<8; x++) {
		LCDsendChar('-');
	}
	return;
}

// Funktio tarkistaa onko virta päällä ja palauttaa 1 jos on.
uint8_t getPowerStatus() {
	if (PINB&(1<<0)) {
		return 1;
	}
	return 0;
}

// Keskeytysrutiini TIMER2 COMPARE
ISR(TIMER2_COMP_vect){
	// Lisää timing arvoa yhdellä keskeytyksen tapahtuessa.
	timing++;
}
```

```
//*****************************************************************************
//
// File Name     : 'minibus.h'
// Title         : Minibus interface
// Author        : Juha-Matti Rensujeff
// Created       : 2016-05-01
// Version       : 1.0
// Target MCU    : Atmel AVR series
//
//*****************************************************************************


#ifndef MINIBUS
#define MINIBUS

    #include <inttypes.h>

    #define MINIBUS_DATA   0        // Määritetään minibus data pinni
    #define MINIBUS_CLK    1        // Määritetään minibus kello pinni
    #define MINIBUS_OUT_ENA  2      // Määritetään minibus out enable pinni

    #define MBP PORTC               // Määritetään minibus portti
    #define MBPIN PINC
    #define MBD DDRC


    #define MINIBUS_DELAY_RCV      5.7  // Odotus osoitteen lähetyksen
                                        // jälkeen

    // Esitellään funtiot
    void MinibusInit(void);           // Alustaa minibus väylän tarvitsemat pinnit
    uint8_t MinibusRead(uint8_t);     // Palauttaa parametrinä annetun minibus
                                      // väylän osoitteen arvon

#endif
```

```
//*****************************************************************************
//
// File Name     : 'minibus.c'
// Title         : Minibus interface
// Author        : Juha-Matti Rensujeff
// Created       : 2016-05-01
// Version       : 1.0
// Target MCU    : Atmel AVR series
//
//*****************************************************************************


#include "minibus.h"
#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>

static void MinibusSendAddr(uint8_t);
static uint8_t MinibusReceive(void);


// Funktio alustaa MINIBUS väylän tarvitsemat liitynnät oikeaan tilaan
void MinibusInit() {
	MBP&=~(1<<MINIBUS_DATA);	// Varmistetaan että kaikki MINIBUS
					// lähdöt alhaalla
	MBP&=~(1<<MINIBUS_CLK);
	MBP&=~(1<<MINIBUS_OUT_ENA);

	MBD|=1<<MINIBUS_DATA|1<<MINIBUS_CLK|1<<MINIBUS_OUT_ENA;
				// Kaikki MINIBUS pinnit ulostuloiksi
	return;
}

// Funtion lukee parametrinä annetun MINIBUS väylän osoitteen ja palauttaa
	// sen arvon
uint8_t MinibusRead(uint8_t addr) {
	uint8_t value = 0;
	MinibusSendAddr(addr);
	_delay_us(MINIBUS_DELAY_RCV);
	value = MinibusReceive();
	return value;
}

// Funktio lähettää MINIBUS väylään kysyttävän osoitteen
static void MinibusSendAddr(uint8_t addr) {
	uint8_t mask;
	uint8_t count = 0;
	uint8_t table[18];		// Taulukkoon tallennetaan minibus
					// portin tilat jokaisella osoitteen
					// lähetyksen vaiheella


	MBP&=~(1<<MINIBUS_CLK);	// Varmistetaan että kello on alhaalla
```

```
	MBD|=(1<<MINIBUS_DATA);		// Minibus data ulostuloksi


	// Luodaan lähetettävä bittikuviot
	table[count] =	MBP|1<<MINIBUS_DATA|
		1<<MINIBUS_CLK|1<<MINIBUS_OUT_ENA;	// ensimmäinen "1" bitti

	for (mask=0x80; mask!=0; mask>>=1) {
		count++;
		if (addr & mask) {
				// Osoite
			table[count] = table[count-1]|1<<MINIBUS_DATA;

		} else {
			table[count] = table[count-1]&~(1<<MINIBUS_DATA);
		}
		table[count]=table[count]&~(1<<MINIBUS_CLK);
			// Ja kello alas

		count++;
		table[count]=table[count-1]|1<<MINIBUS_CLK;
			// Kello ylös
	}

	count++;
	table[count]=table[0]&~(1<<MINIBUS_OUT_ENA|1<<MINIBUS_CLK);
		// Lopuksi Minibus vastaanottotilaan


	MBP|=1<<MINIBUS_OUT_ENA|1<<MINIBUS_DATA;	// Minibus lähetystilaan
							// ja data bitti ylös

	count = 0;
	while (count < 18) {				// Aloitetaan lähetys

		MBP = table[count];
		count++;
		asm volatile ("nop");
		MBP = table[count];
		count++;
		asm volatile ("nop");
		asm volatile ("nop");
		asm volatile ("nop");

	}


	MBD&=~(1<<MINIBUS_DATA);	// Minibus data portti sisääntuloksi.
	MBP&=~(1<<MINIBUS_DATA);	// Ylösveto pois
	return;
}
```

```
// Funktio lukee MINIBUS väylästä 8 bittiä
uint8_t MinibusReceive() {
	uint8_t value = 0;


	for(int8_t x=7; x>=0; x--) {
		asm volatile ("nop");
		asm volatile ("nop");
		if (MBPIN & 1<<MINIBUS_DATA){
			value|=1<<x;
		}
		MBP |= 1<<MINIBUS_CLK;
		asm volatile ("nop");
		MBP &=~(1<<MINIBUS_CLK);

	}

	return value;
}
```