# ECIO-40P -LAAJENNUSKORTTI

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietokone-elektroniikka
Opinnäytetyö
Syksy 2009
Heikki Poskiparta

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

POSKIPARTA, HEIKKI: ECIO-40P laajennuskortti

Tietokone-elektroniikan opinnäytetyö, 24 sivua, 20 liitesivua

Syksy 2009

TIIVISTELMÄ

---

Tämän päättötyön aiheena oli suunnitella ja toteuttaa monitoiminen laajennuskortti ECIO-40P- mikro-ohjaimelle sekä suunnitella valmistettavalle kortille sovellus elektronisena termostaattina. Työ tehtiin Koulutuskeskus Salpaukseen, jossa laajennuskorttia ja suunniteltua sovellusta käytetään matkajääkaapin termostaattina sekä muissa opetussovelluksissa.

Elektroninen termostaatti koostuu Ecio40P -mikro-ohjaimesta, laajennuskortista ja LM45 lämpötila anturista. Työn toteutus aloitettiin suunnittelemalla ECIO40P -mikro-ohjaimelle laajennuskortti, jolla mikro-ohjaimella olevia toimintoja voidaan laajentaa. Työn vaatimuksina oli että, Ohjainkortti ja siihen liitettävät oheislaitteet tulisivat toimimaan itsenäisenä ohjainyksikkönä.

Työn toisessa vaiheessa ohjelmoitiin ECIO-40P -mikro-ohjaimelle sovellus, joka mahdollistaisi suunnitellun laitteisto kokoonpanon käytön elektronisena termostaattina. ECIO-40 P -mikro-ohjaimen ohjelmointiin käytettiin Matrix Multimedian Flowcode V4 graafista ohjelmointiympäristöä. Valmistunut ohjain sovellus siirrettiin mikro-piirille USB -kaapelilla.

Valmistuneen laajennuskortin piirilevyn suunnitteluun käytettiin Cadsoft EAGLE 5.6.0 -piirilevyn suunnitteluohjelmaa.

Avainsanat: ECIO 40P, laajennuskortti, Flowcode

Lahti University of Applied Sciences
Degree Programme in Information Technology

POSKIPARTA, HEIKKI: Extension board for ECIO-40P

Bachelor's Thesis in Information Technology 24 pages, 20 appendices

Autumn 2009

ABSTRACT

---

The aim of this thesis was to design and implement a multipurpose extension board for the ECIO-40P microcontroller andto design an application for the board in electronic temperature control. The extension board was made for Salpaus Further Education where it was planned to be used as a fridge thermostat and for other educational purposes.

The electronically thermostat consists of these components: ECIO-40P microcontroller, extension board and LM45 temperature sensor. The project began by designing an extension board which expands the functions of the microcontroller. The requirement of the project was that the microcontroller and the devices attached to it can function on independent mode when the controller is taken off the coding mode.

The next step in the project was coding the application for the microcontroller. The code enables the devices to act as a temperature controler. Software for the microcontroller was developed by using Flowcode V4 graphical integrated development environment at Matrix multimedia. Completed software application was ported to the microcontroller by using a USB cable.

Eagle 5.6.0 layout editor was used in designing the PCB board for the extension card

Key words: ECIO-40P, multipurpose extension card, Flowcode

SISÄLLYS

LYHENNELUETTELO

LCD = Liquid Crystal Display

USB = Universal Serial Bus

LED = Light-Emitting Diode

RISC= Reduced Instruction Set Computer

# 1 JOHDANTO

Työn tavoitteena oli suunnitella ja toteuttaa laajennus- ja ohjainkortti, jolla ohjataan ja tarkkaillaan jääkaapin lämpötilaa termostaatin sijasta. Opinnäytetyön aiheen antoi Koulutuskeskus Salpauksen opettaja Jarmo Salo. Valmistuvaa laajennus korttia tullaan hyödyntämään lopputyön ohella Koulutuskeskus Salpauksen muissakin opetustöissä.

Opinnäytetyön sisältö jakaantuu kahteen erilliseen osioon: ECIO40P kehityskortin mikrokontrollerin ohjelmoimiseen sekä laajennuskortin suunnitteluun ja toteutukseen. Työn ensimmäisessä osiossa suunniteltiin laajennuskortti ECIO40P kehityskortin mikrokontrollerille. Suunnittelu ohjelmana käytettiin Cadsoft EAGLE 5.60 piirilevyn suunnittelu-ohjelmaa.

Työn ensimmäisessä osiossa valmistunutta laajennuskorttia hyödynnettiin asentamalla se ECIO40P – kehityskortin ja ohjattavan laitteen väliin. Kehityskortin ytimenä toimii Microchipin PIC 18f4455 mikro-ohjain. Tässä työssä kehityskortin mikro-ohjain ohjelmoitiin Matrix Multimedian suunnittelemalla Flowcode v4-ohjelmalla. Työn tavoitteena oli valmistaa itsenäisesti toimiva ohjainyksikkö ohjattavan laitteen yhteyteen. Tarvittaessa ohjainyksikön pystyi uudelleen ohjelmoimaan vaivattomasti ohjainyksikköön liitetyn ohjelmointi kaapelin avulla. Ohjelmointi tapahtui tietokoneen ja mikrokontrollerin välillä USB-kaapelilla.

# 2 LAITTEEN MÄÄRITTELY

Laajennuskortin perusajatuksena oli suunnitella piirikortti, joka pitäisi sisällään ECIO40P -mikro-ohjaimen vaatimat peruskomponentit. Peruskomponenttien lisäksi perusajatuksena oli mikro-ohjaimella olevien signaalipinnien vaivaton liittäminen ohjattavaan laitteeseen ruuviliittimien avulla. Kortin ulkomittojen tuli olla yhteensopiva DIN- kiskoon liitettävien moduuli standardien kanssa. Cadsoft EAGLE piirilevyn suunnitteluohjelman rajoituksista johtuen valmistettavan piirilevyn maksimimitoiksi määräytyi 100mm x 80mm.

Vaatimuksena oli myös, että mahdollisimman moni mikro-kontrollerin signaali voitaisiin ohjata laajennuskortissa sijaitseville ruuviliittimille, jotta niitä on mahdollisimman vaivatonta liittää laajennuskorttiin kytkettävään laitteeseen. Ruuviliittimet myös helpottaisivat osaltaan mahdollisten laajennuskortilla ilmenevien vikojen etsimisessä.

Laitteistolle tuleva sähkönsyöttö asetti suunniteltavalle laitteelle omat vaatimuksensa. Laitteen oli pystyttävä toimimaan myös itsenäisenä ohjainyksikkönä. Tämä edellytti että laajennuskortille oli suunniteltava tarvittavat liittimet ulkoista sähkönsyöttöä varten sekä regulaattorikytkentä mikrokontrolleria ja muita korttiin liitettäviä +5V käyttöjännitettä vaativia laitteita varten.

Laajennuskortin käyttöympäristö ja mahdolliset käyttäjät asettivat valittaville komponenteille tiettyjä vaatimuksia. Käyttäjäryhmä sekä laitteen kasauksessa käytettävä laitteisto pois sulkivat mahdollisuuden käyttää työssä pintaliitoskomponentteja. Komponenttien sijoittelu omille paikoilleen piirilevyllä oli tarkoin määriteltyä koska valmis piirikortti tultaisiin asentamaan DIN-kiskomoduuliin. Tämä osaltaan määräsi kortille asennettavien ruuviliittimien sijoittamisen piirilevyn pidemmille sivuille sekä läpi juotettavien komponenttien sijoittelun ainoastaan piirilevyn toiselle puolelle.

Mikro-ohjaimelta ruuviliittimille vietävien I/O-nastojen lukumäärä jäi vielä laitteistomäärittelyssä avoimeksi, koska mikro-ohjaimella olevien I/O-pinnien tarkasta lukumäärästä ei ollut tarkkaa tietoa. Myöskään piirilevyn reunoille tulevien ruuviliittimien aiheuttamasta tilan käytöstä ei määrittely tilanteessa ollut tarkempaa käsitystä.

Edellisten määrittelyjen lisäksi laajennuskortin tuli sisältää liittimet virransyötölle. Liittimiä käytettäisiin ohjaamaan suurempaa virtaa ja jännitettä vaativia laitteita. Kyseinen määrittely tuli toteuttaa releen ja releen ohjaimen avulla.

Seuraavat komponentit päätettiin sijoittaa laajennuskortille:

- kanta ECIO40P mikro-ohjaimelle
- ruuviliittimet mahdollisimman monelle I/O-pinnille
- 5 V:n regulaattoripiiri & ulkoinen sähkönsyöttö
- 1 kpl relelähtöjä sekä kanta releelle
- reguloitu 5V sähkönsyöttö lämpötila-anturille tai muille 5V lisälaitteille
- 100nF suodatinkondensaattori mikro-ohjaimelle.

Oheiskomponentit

Myöhemmin levylle lisättävien komponenttien määrää kasvatettiin. Piirille lisättiin releelle menevään ohjaus signaaliin jumpperi, jolla voidaan valita käytetäänkö signaalia releen ohjaukseen vai normaalina signaalilähtönä. Jumpperin lisäksi piirilevylle lisättiin kaksi kappaletta led valoja, joiden avulla voidaan seurata sen signaalijohtimen toimintaa johon kyseiset ledit on kytketty. Näiden lisäksi piirilevylle suunniteltiin lisättäväksi kaksi kappaletta painonappeja, joiden avulla voi kontrolloida tai muuttaa mikrokontrollerille menevien signaalien tilaa.

Mikro-ohjaimeksi kehitettävälle laajennuslevylle valittiin Matrix Multimedian suunnittelema USB:n kautta ohjelmoitava 40-pinninen ECIO-40P, jonka ytimenä toimii Microchipin valmistama PIC18F4455.

Mikrokontrollerin ohjelmointiympäristöksi sovittiin käytettäväksi myös Matrix Multimedian kehittelemää graafista Flowcode v4 -ohjelmointi ja simulointiympäristöä

# 3 ECIO-40P LAAJENNUSKORTIN SUUNNITTELU

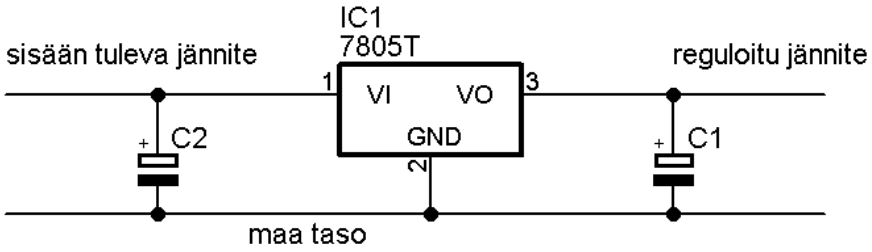
Laajennuskortin suunnittelussa päämääränä oli suunnitella kortti, joka mahdollistaisi ECIO 28P- ja ECIO 40P- mikro-ohjainten vaivattoman liittämisen oheislaitteisiin. Tämän lisäksi valmistettavan kortin tulisi laajentaa siihen kytkettävien kehityskorttien ominaisuuksia. Valmistettavan laajennuskortin käyttö opetustyössä rajasi kortille valittavien komponenttien ominaisuuksia. Valittavien osien vaatimuksina oli vaivaton komponenttien vaihtaminen sekä vian etsintää helpottava ulkoasu. Vaatimukset rajasivat valittavat osat käsittämään ainoastaan jalallisia läpijuotettavia komponentteja.

## 3.1 Komponenttien luonti

Osa piirilevylle suunniteltavista ja sijoitettavista komponenteista oli sellaisia joita ei EAGLE:n kirjastosta tai Cadsoftin sivuilta löytyvistä komponenttikirjaston täydennyspaketeista löytynyt. Matrix Multimedian ECIO-40P:n ja Omronin G2RL-1-E:n komponentteja ei täydennys paketeista huolimatta löytynyt EAGLE:n kirjastoista. Puuttuvien komponentti tiedostojen luontiin käytin Cadsoft EAGLE:n kirjastoeditoria, jolla pystyy luomaan tai muokkaamaan tarvittavien komponenttien ulkoasua ja piirrosmerkkiä. Editorilla komponentille luotiin piirikaavio editorissa käytettävä piirrosmerkki sekä ulkoasu editorissa.

## 3.2 Jänniteregulaattori

Piirilevylle tulevien komponenttien virransyötöksi valittiin määrittelyvaiheessa jänniteregulaattori. Jänniteregulaattoriksi valitsin Fairchild Semiconductorin valmistaman viiden voltin positiivisen jännitealueen regulaattorin KA7805:n. Sisään menon jännitealueeksi valmistaja on määritellyt 7-20 volttia. Ulostulojännitteeksi valmistaja lupaa tyypillisesti +5 V ja ulostulo virta-alueeksi 5mA- 1A. Virhemarginaaliksi ulostulo jännitteelle on valmistaja määritellyt 4 %:n marginaalin. Toimintalämpötila alueeksi on määritelty 0 ºC - +125 ºC. Maksimilämpötilaksi ilman lisäjäähdytystä valmistaja lupaa 65 ºC. (Fairchild 2008. 3) Regulaattorin kytkennässä käytettiin korjaavaa ulostulo kytkentää (KUVIO 1.) . Kytkennässä regulaattorin molempiin virransyöttöihin on liitetty elektrolyytti kondensaattori. Sisään tulevan jännitteen puolella oleva C1 -kondensaattori auttaa ulkoa tulevien häiriöiden poistamisessa. Ulostulon puolella oleva elektrolyyttikondensaattori tasaa äkillisiä virran notkahduksia.

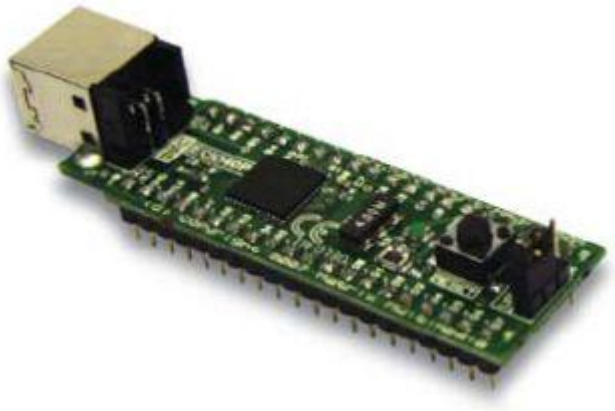



KUVIO 1. Jänniteregulaattorin kytkentä(Fairchild 2008, 23).

## 3.3 ECIO-40P mikro-kontrolleri

Matrix Multimedian kehittelemä ECIO-40P (KUVIO 2.). Mikrokontrollerialusta jota tässä opinnäytetyössä on käytetty, sisältää Microchipin valmistaman PIC 18F4455 -mikro-ohjaimen. Varsinainen mikro-ohjain on juotettu kiinni Matrix Multimedian kehittelemään alustaan, joka laajentaa mikro-ohjaimen toimintaa.

ECIO-40P alustaan on varsinaisen mikro-ohjaimen lisäksi lisätty ulkoinen 4Mhz kello-oskillaattori ja USB -liitin. Näiden ominaisuuksien lisäksi piirilevylle on lisätty reset -painonappi ja kontrolleri, joka sisältää piirille valmiiksi ohjelmoidun boot loaderin. Piirillä sijaitsee jumpperi, jolla valitaan onko laite ohjelmointi vai toiminta tilassa. Piirillä sijaitsevan jumpperin asento määrittelee mikro-ohjaimelle tulevan sähkönsyötön lähteen, eli saako mikro-ohjain tarvitsemansa sähkönsä USB:n kautta vai ulkoisesta virtalähteestä. Piirillä sijaitseva ulkoinen oskillaattori toimii mikro-ohjaimen sisäisen 48 MHz kellon esijakajana. Tämä on edellytyksenä, että piirillä oleva USB-väylä toimii täydellä siirtonopeudella. Lisäksi Mikrokontrollerialustaan on lisätty piikkirimat, jotka on mitoitettu sopimaan 0.6 tuuman DIL- piirikantaan. (Matrix multimedia 2008, 2-3.)



KUVIO 2. ECIO-40P (Microcontrollershop, 2009 ).

## 3.4 PIC18F4455 mikro-ohjain

PIC18F4455 kuuluu microchipin PIC18 -perheeseen, joilla on korkea suoritusteho ja integroitu A/D muunnin. Kaikki PIC18 -perheeseen kuuluvat mikro-ohjaimet toimivat edistyksellisellä RISC -arkkitehtuurilla. Mikro-ohjaimen Harvard-arkkitehtuurin erilliset data- ja ohjelmaväylät mahdollistavat 16 -bittisen käskyjen

ja 8 -bittisen datan käytön. Kaksitasoinen käskyjono mahdollistaa näissä mikrokontrollereissa käskyjen suorittamisen yhden jakson aikana. PIC 18 perheen sisältämät erikoisominaisuudet vähentävät mikro-ohjaimen ulkoisten komponenttien lukumäärää sekä vähentävät laitteessa tarvittavan virran kulutusta. Ohjelmointiväylänä opinnäytetyössä käytetty mikro-ohjain käyttää USB -väylää. Mikro-ohjaimen käyttämää ohjelmamuistia voidaan uudelleen ohjelmoida tyypillisesti noin 100000 kertaa, ja käyttömuistina toimiva EEPROM -muisti kestää 1000000 kirjoitusjaksoa. (microchip, 2009.)

Alla on listattuna opinnäytetyössä käytetyn PIC18F4455 mikro-ohjaimen tärkeimpiä ominaisuuksia:

- 24kb Flash-ohjelmamuistia
- suorittimen nopeus 12MIPS
- 2048B SRAM-muistia
- 256B EEPROM -muistia
- digitaalisia liitäntöjä: 1-A/E/USART, 1-MSSP(SPI/I2C)
- laskureita: 1 x 8-bit, 3 x 16-bit
- analogia-digitaali- muunnin: 13 kanavaa, 10-bit resoluutio
- komparaattoreita: 2kpl
- I/O-pinnejä: 35kpl
- PWM-lähtöjä 1kpl
- USB-väylä: 1kpl, USB 2.0
- toiminta jännitealueella 2.0-5.5V

(Microchip 2009, 1)

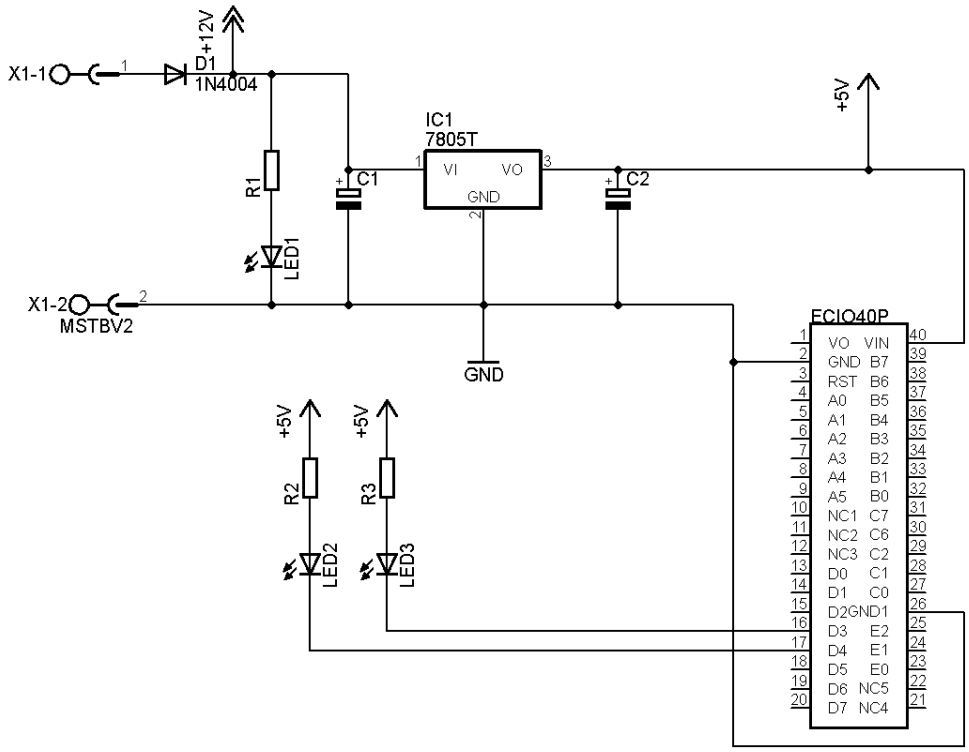
# 4 PIIRILEVYJEN SUUNNITTELU

Lopputyön aikana suunniteltiin ja tehtiin valmistettavasta piirilevystä kaksi eri versiota. Ensimmäinen versio sisälsi mikro-ohjaimen toiminnan testaukseen tar-

vittavat komponentit. Toiseen versioon tullaan korjaamaan ensimmäisen levyn valmistuksen aikana huomatut ongelmat ja virheet.

## 4.1 Protolevy

Laajennuskortin suunnittelun ensimmäinen versio sisälsi ainoastaan pakolliset komponentit mikrokontrollerin ohjelmointia ja testausta varten. Ensimmäinen versio sisälsi näin ollen jännite regulaattorikytkennän mikropiirin tarvitsemaa käyttöjännitettä varten. Regulaattorin sisään tulevan sähkön syötön puolelle oli kytketty vastus-diodi estämään ja ilmaisemaan virransyötön vääränapaisuus. Tämän lisäksi laajennuskortille oli asennettu kaksi kappaletta ledejä ja piirikanta, johon mikrokontrolleri tultaisiin lopullisessa laitteessa kytkemään. Kyseisellä kokoonpanolla pystyttiin todentamaan levylle suunnitellun regulaattorin kytkennän toiminta sekä ohjelman ja input- output-pinnien toimivuus. (KUVIO 3.)




KUVIO 3. ECIO-40P testikytkentä

### 4.2 Havaitut suunnitteluvirheet

Laajennuskortin kokoamisvaiheessa kävi ilmi suunnittelun ja valmistuksen aikana tapahtuneita virheitä jotka aiheuttaisivat työ edistyessä ongelmia. Havaitut ongelmat pyrittiin havaitsemaan ja korjaamaan ennen piirilevyn lopullisen version tekemistä. Ensimmäinen ongelma ilmeni jo ennen piirilevyn syövytystä. Ongelman aiheuttajana oli valotuskalvon puutteellinen tulostusjälki. Tulostusjäljessä oli tulostuksen aikana tullut naarmu jossa ei ollut ollenkaan johdinalueita peittävää väriä. Värin puute olisi aiheuttanut syövytystilanteen jälkeen muutaman piirilevylle syövytetyn signaalin johtamattomuuden. Ongelma korjattiin ennen syövytystä kynällä, joka on tarkoitettu johtimien piirtämiseen piirilevylle.

Toinen ongelma ilmeni piirilevyn porauksen yhteydessä. Osa asennettavien komponenttien juotoskohdista oli asennettaville komponenteille liian pieniä. Tästä johtuen piirilevyä poratessa juotoskohdan tinaus hävisi lähes kokonaan ja asennettavan komponentin juottaminen piirilevylle hankaloitui. Ongelman korjaus tapahtui EAGLE:n komponenttikirjastoja muokkaamalla. Ongelmallisten komponenttikirjastojen sisältämien komponenttien juotoskohtien kokoa suurennettiin.

Kolmas ongelma ilmeni piirilevyn kasausvaiheessa. Suunnitteluvaiheessa valittujen ruuviliittimien kotelointi ja tilan tarve levyltä poikkesi levylle asennettavien liittimien koteloinnista. Levylle tulevat ruuviliittimen kotelointi tarvitsi enemmän tilaa levyltä, kuin mitä suunnitteluohjelman vastaava kotelointi. Tämä aiheutti muutamien komponenttien päällekkäisyyksiä piirilevyä kasatessa. Ongelma ratkesi protolevyä kasatessa sijoittamalla päällekkäiset komponentit eri puolille piirilevyä. Lopullisen piirilevyn valmistukseen kyseinen ongelma ratkaistaan siirtämällä ongelmalliset komponentit kauemmaksi toisistaan. Tällöin päällekkäisyyksiä ei esiinny ja kaikki komponentit voidaan sijoittaa piirilevyn samalle puolelle.

Käynnistysvaiheen ongelma ilmeni vaiheessa, jossa oli tarkoitus testata laitteen toimintaa itsenäisenä ohjainyksikkönä. Vika ilmeni maapotentiaalin puutteena osassa piirilevyä. Regulaattorilta eteenpäin lähtevä kytkentä haarautui komponentin jalan kohdalta kahteen eri suuntaan. Toisessa johtimen haarassa maataso toimi

ilman ongelmia ja toisesta johtimesta maataso puuttui täysin. Vika korjaantui juottamalla haara kohdassa olevan komponentin jalka uudelleen piirilevylle, jolloin ilmennyt vikakin korjaantui.

Testausvaiheen alussa ilmeni toinenkin maatason etenemisongelma. Protolevyssä käytetty piirilevy oli suunniteltu täysin valmiin laajennuslevyn pohjaratkaisuksi. Osa piirillä olevista käyttöjännitevedoista oli suunniteltu kulkevaksi suoraan piirilevylle sijoitettavien komponenttien kahden jalan kautta seuraavalle komponentille. Koska kyseisiä komponentteja ei protolevylle asennettu, estyi käyttöjännitesignaalien kulku puuttuvia komponentteja seuraaville osille. Väliaikainen ongelma ratkaistiin asentamalla puuttuvien komponenttien tilalle hyppylangat, jotka korjasivat signaalivedot johtaviksi.

## 4.3 Lopullinen versio

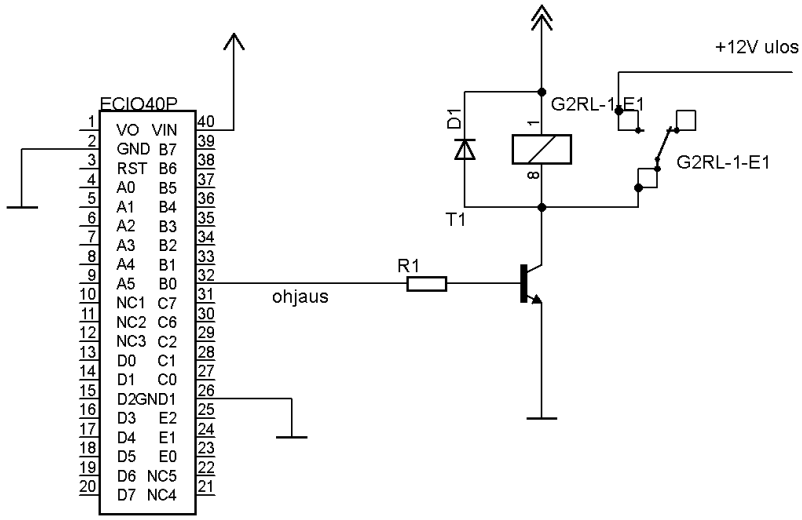
Kytkennän toiseen sekä samalla lopulliseen version kytkentään lisäsin aiemmin suunniteltujen komponenttien lisäksi kaksi kappaletta painokytkimiä joilla voitaisiin vaihtaa kahden inputsignaalin tilaa tarvittaessa. Saadakseni laajennuskortin käytettävistä ominaisuuksista monipuolisempia, lisäsin painonappikytkentöihin jumpperit, joilla painonapit voitiin irrottaa +5V käyttöjännitteestä. Tällöin kyseisten signaalipinnit voivat toimia I/O pinneinä. Tämän lisäksi tuli suunnitella kytkentä releohjaimelle ja releelle. Ensimmäisen version piirikaavion pohjalta kävi selville, että kaikille mikro-ohjaimesta löytyville signaaleille ei olisi mahdollista kytkeä ruuviliittimiä piirilevyn käytettävästä tilasta johtuen. Tilan puutteesta johtuen oli toisen version kytkentäkaavioon tehtävä muutoksia. Muutokset koskivat lähinnä parin portin ruuviliittimien vaihtamista piikkirimoiksi. Muutoksen kohteiksi joutuivat portit B ja C. Osa piikkirimoille menevistä porttien B ja C pinneistä sisältää useamman toiminnon. Nämä signaalijohtimet on kytketty piikkirimojen lisäksi ruuviliittimiin. Kytkentä kaavion suunnitelmiin lisättiin myös lisälaitteiden tarvitseman + 5V ja GND virransyötön ruuviliittimet.

### 4.3.1 Releohjain

Releen ohjaus kytkentä perustuu transistorin kykyyn toimia kytkimenä. Kytkennässä pienellä ohjausjännitteellä voidaan ohjata transistorin yli vaikuttavaa kollektorivirtaa, jolla transistoriin kytkettyä laitetta ohjataan. Kytkentä jossa transistoria käytetään kytkimenä voi olla esimerkiksi (KUVIO 4.):n mukainen. Esimerkin kytkennässä transistorin T1 kannalle tuodaan mikrokontrollerin I/O pinnistä B0 signaali, joka ylittää kannan ja emiterin vaatiman kynnysjännitteen transistorin tila muuttuu johtavaksi. Kynnysjännitteen ylityksestä johtuen myös kollektori emitteri virran tila muuttuu johtavaksi. Muutos saa aikaan että releen solenoidiin vaikuttava käämi tulee jännitteelliseksi ja rele vaihtaa tilaansa.

> *Mikäli releen käämin rinnalla on suojadiodi, on se kytkettävä estosuuntaan eli positiivinen jännite tulee sille puolelle, johon kolmion tai nuolen kärki osoittaa eli sille puolelle, jossa on poikkiviiva. Itse komponenttiin on yleensä piirretty vain pelkkä poikkiviiva. Suojadiodin tarkoitus on oikosulkea käämin aiheuttama jännitepiikki, joka syntyy, kun käämiltä nopeasti katkaistaan virta. Käämin aiheuttama jännitepiikki saattaa aiheuttaa ongelmia relettä ohjaavilla transistoriasteilla. Mikäli jännite kytketään rinnakkaisdiodilla suojatulle käämille väärin päin se rikkoutuu hyvin suurella todennäköisyydellä, ellei napaisuutta ole suojattu toisella sarjadiodilla.* (Halytin, 2009.)

Näin ollen Kytkennässä oleva diodi estää induktion purkaantumisesta aiheutuvan vääräsuuntaisen virrankulun transistorille T1 ja siitä takaisin mikropiirille. Kytkennässä oleva vastus on yleensä arvoltaan kohtalaisen suuri, jolloin transistorille menevä virta ei hajota transistoria mutta antaa tarvittavan virtamäärän transistorin ohjaukselle.




KUVIO 4. Releohjaimen kytkentä.

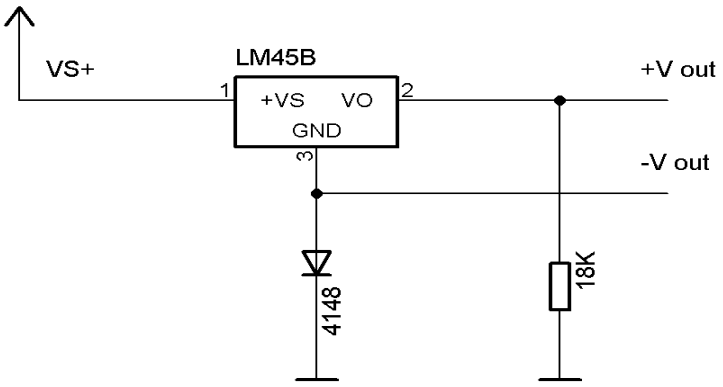
### 4.3.2 Rele

Releeksi valittiin Omronin valmistama G2RL-1-E. Releen valintaan vaikuttivat komponentin pieni koko ja hyvät kytkentäominaisuudet sekä suuri virranhallinta-kapasiteetti. Releen kytkentäjännitteeksi on määritelty 5V. Valmistajan mukaan releen käämi kytkeytyy ja rele vaihtaa tilaansa, jos jännite on maksimissaan 70 % nimetystä toimintajännitteestä. Vastaavasti käämin varaus vapautuu kun jännite on minimissään 10 % nimellisjännitteestä. Toiminta ajaksi releelle luvataan 30000 työminuuttia, kun releellä on kuormana 24V tasajännite ja virtana 16 ampeeria. (Omron 2008, 132–135.)

### 4.3.3 LM45

Lämpötila-anturiksi työhön valittiin aiemmin suunniteltu ja toteutettu National Semiconductorin LM45:n mallikytkentään perustuva lämpötila-anturi. Kyseisellä kytkennällä anturin mittaus alueeksi saadaan -20 ºC - + 100 ºC (KUVIO 5.). Lämpötila-anturin lähtö jännitteeksi valmistaja on määritellyt 10mV / 1 ºC lineaarisen

skaalauksen. Kyseinen anturi kytkentä soveltuu hyvin opinnäytetyön sovelluskohteeseen, jossa kohteen käyttölämpötila ei laske alle 0 ºC asteen. (National 2005, 3)




KUVIO 5. Lämpötila-anturin kytkentä (National 2005,7.)

## 4.4 Kytkennän toiminta

Virransyöttö ECIO- 40P:n laajennuskortille tapahtuu ruuviliitin PWR kautta. Sisään tuleva virransyöttö on suojattu diodilla ja oikean kytkennän napaisuuden varmistaa ruuviliittimen napojen väliin kytketty vastus- led-yhdistelmä.

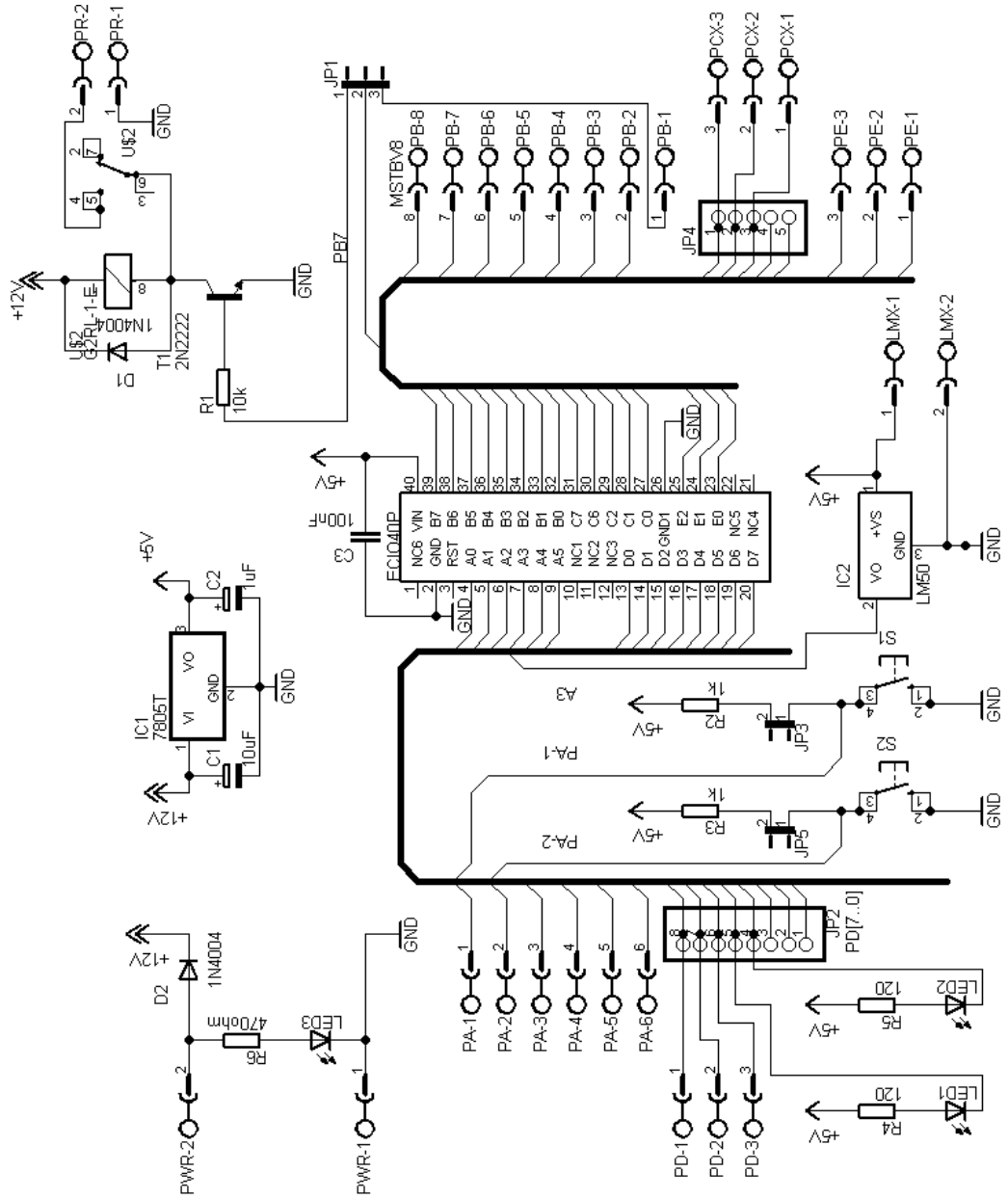
Regulaattorin molemmilla puolilla sijaitsevat kondensaattorit auttavat regulaattorin toiminnassa. Sisään syötettävän virran puolella sijaitseva 10uF -kondensaattori C1 ehkäisee syntyneitä häiriöitä, jos virransyötön suodin sijaitsee kaukana regulaattorista. Regulaattorin ulostulevan virran puolella oleva kondensaattori taas tasaa jännitteen syöttöä nopeiden syöttöjännitteen alenemisien aikana.

Ecio-40P:ltä suurin osa signaalipinneistä on kytketty suoraan ruuviliittimille meneviin kuparijohtimiin. Poikkeuksena ovat mikro kontrollerin portit D ja C, jonka kaikki pinnit on kytketty piirilevyllä oleviin piikkirimoihin. Portin D signaalit on yhdistetty piikkirimaan JP2 ja portin C taas piikkirimaan JP4. Osa piikkirimoille viedyistä signaaleista on kuitenkin kytketty myös ruuviliittimille. Tämä johtuu siitä, että kyseisissä signaalipinneissä on useampia toimintoja, joita pienenkin toiminnon muutoksen jälkeen voidaan tarvita.

Piirilevyllä sijaitsevat ledien LED1 ja LED2 katodit on kytketty mikrokontrollerin signaalijohtimiin D4 ja D3. Ledien anodit on suojattu ylijännitettä vastaan 120 ohmin vastuksella, jotka on kytketty +5V käyttöjännitteeseen. Ledit aktivoituvat, kun mikro-ohjaimelta saapuva signaali saa arvon 0.

Portin A pinneihin A0 ja A1 on kytketty piirilevyllä sijaitsevat painonapit. Kytkentä on toteutettu haaroitusmenetelmällä, jossa pinneille menevät signaalit on kummatkin kytketty jumpperin ja yhden kilo-ohmin vastuksen kautta +5V käyttöjännitteeseen. Ennen vastusta kytkentä haarautuu painokytkimille, joiden toinen pää on vuorostaan kytketty maa tasoon. Kytkintä painettaessa kytkentä sulkeutuu ja +5 voltin jännite ohjautuu suoraan maatasoon, minkä takia piirille menevän signaalin tila muuttuu +5V:sta 0V:iin. Piirin monipuolisuuden takia voidaan painonappi kytkennöissä olevat jumpperit irroittaa, jolloin kyseiset portit toimivat normaaleina I/O pinneinä.

Portin B pinnistä 7 lähtevän signaalin kulkua pystytään vaihtamaan jumpperilla JP1. Jumpperin Ensimmäinen vaihtoehto ohjaa signaalin kulkemaan suoraan ruuviliittimille. Toinen vaihtoehto ohjaa signaalin 10 kilo-ohmin suojavastuksen läpi transistorin kannalle. Signaalin saadessa + 5V:n loogisen ykköstilan. Laitteen kytkentäkaavio on esitetty KUVIOSSA 6.




KUVIO 6. Lopullisen laitteen kytkentä

## 5 PIIRILEVYN VALMISTUS

Piirilevyn valmistus tapahtui Koulutuskeskus Salpauksen tiloissa. Tekniikkana käytettiin yksinkertaista valotus-syövytysmenetelmää. Aluksi valmistettava piirilevy leikattiin DIN -kiskoon asennettavan piirilevykehikon kokoiseksi. Tämän jälkeen piirilevystä poistettiin UV-kalvo mikä estää haitallisen valon aallonpituuk-

sien pääsyn piirilevyn pinnalle valmiiksi suihkutettuun valoherkkään pinnoitteeseen. Seuraavaksi asemoidaan kalvolle tulostettu piirikaaviokuva valmistettavalle piirilevylle ja kalvo sekä piirilevy sijoitetaan valotuskoneeseen, jossa UV-valo aikaan saa kalvolle tulostetun johdinkaavion siirtymisen piirilevylle. Valotettu piirilevy siirretään lämmitetyllä ferrikloridilla täytettyyn syövytyslaitteeseen, jossa ylimääräinen kupari syövytetään pois levyltä. Kun levyn kuparivedot ovat valmiita, piirilevy huudellaan vedellä. Tämän jälkeen piirilevy on valmis porattavaksi.

# 6 LAITTEISTOLLE TEHDYT MITTAUKSET

Laitteiston ainoat mittaukset tapahtuivat piirilevyn ensimmäisestä versiosta. Mikro ohjaimeen oli testejä varten ohjelmoitu termostaattikokoonpanossa käytettävä ohjelma, jotta saaduista mittaustuloksista tulisi mahdollisimman lähellä totuutta olevia.

## 6.1 Jänniteregulaattorin mittaukset

Ensimmäisenä mitattiin piirilevylle asennetun jänniteregulaattorin piireille antaman jännitteen lukuarvo. Ennen testin aloittamista laajennuskortista poistettiin kaikki komponentit mitkä mahdollisesti vaikuttaisivat regulaattorin mittaustulokseen. Mittauksen testilaitteistona testauskytkennöissä käytettiin säädettävää virtalähdettä TL 3035, jonka jännite voitiin säätää välillä 0 – 30 V DC. Virtalähteelle määritelty virtamaksimi on 3 Ampeeria. Testi kytkennällä regulaattorin ulostulojännitteeksi saatiin 4,92 V, ja sisäänmenojännitteeksi oli säädetty 12V.

### 6.2 ECIO-40P mittaukset

Toisena mittauksen suoritettiin ECIO-40P:n I / O -pinnien ulostulo jännitteet, kun mikro-ohjain oli kytkettynä laajennuskorttiin. Testikytkennän jännitteellisissä arvoissa ei esiintynyt minkäänlaisia muutoksia regulaattorilta saataviin tuloksiin nähden, vaikka mikro-ohjain oli kytkettynä. I/O -porttien jännitteet olivat aktiivitilassa 4,92 V, ja passiivitilassa pinnien jännite laski 0 volttiin.

## 7 OHJELMOINTI

Lopputyön ohjelmointi työkaluna käytin englantilaisen Matrix Multimedian toimittamaa Flowcode -ohjelmointiympäristöä. Flowcode -ohjelmointiympäristö sisältää C- ja Assemply-kääntäjät sekä simulaatiotilan, jossa suunnitellun ohjelman toiminnan voi testata ennen piirille latausta.

### 7.1 Laitteiston ominaisuuksien määrittäminen

Ohjelman suunnittelun toteutus alkaa valitsemalla mikropiiri, jolle suunniteltava ohjelma tullaan asentamaan. Valinta määrittelee lopullisessa koodissa määritetyt muistiosoitteet ja piirin sisäiset määritykset, esimerkiksi piirissä käytettävän kellotaajuuden.

Ennen varsinaista ohjelman kirjoitusosuutta valitaan, millaisia sisäisiä ominaisuuksia tai ulkoisia komponentteja tullaan valmiissa laitekokonaisuudessa käyttämään. Tällaisia komponentteja voivat olla muun muassa laitteen PWM-lähtö tai laitteeseen kiinnitettävä graafinen LCD -paneeli. Komponenttien valinnan jälkeen jokaiselle komponentille määritellään laitteessa käytettävät portit ja signaalipinnit joihin kyseisen osan toiminta vaikuttaa tai joihin kyseinen laite tullaan kytkemään,

sekä osan omat toiminnot, esimerkiksi minkälaisessa signaalin tilassa LED on aktiivinen. Tämän lisäksi määritellään valitulle osalle simuloinnissa käytettävä ulkoasu.

## 7.2 Vuokaavion rakentaminen

Varsinaisessa ohjelmointi osuudessa suunnitellaan vuokaavio, jonka mukaan mikrokontrollerille tuleva ohjelma suorittaa toiminnot. Vuokaavion rakenne osat löytyvät ohjelman vasemmassa reunassa pystyssä sijaitsevasta valintapaneelista. Kuvakkeet tässä valinta paneelissa noudattavat vuokaaviolle tyypillisiä piirrosmerkkejä. Vuokaavion suunnittelu tapahtuu raahaamalla kyseisiä kuvakkeita ohjelmointi-ikkunaan aloitus- ja lopetus-kuvakkeiden väliin sekä asettamalla kuvake sille paikalle, jossa sen halutaan vaikuttavan ohjelman toimintaan. Tämän jälkeen kyseessä olevalle vuokaavion osalle määritellään minkä kytkentä paneelissa olevan laitteen ominaisuuksia valittu vuokaavion osa tulee toteuttamaan.

## 7.3 Simulointi

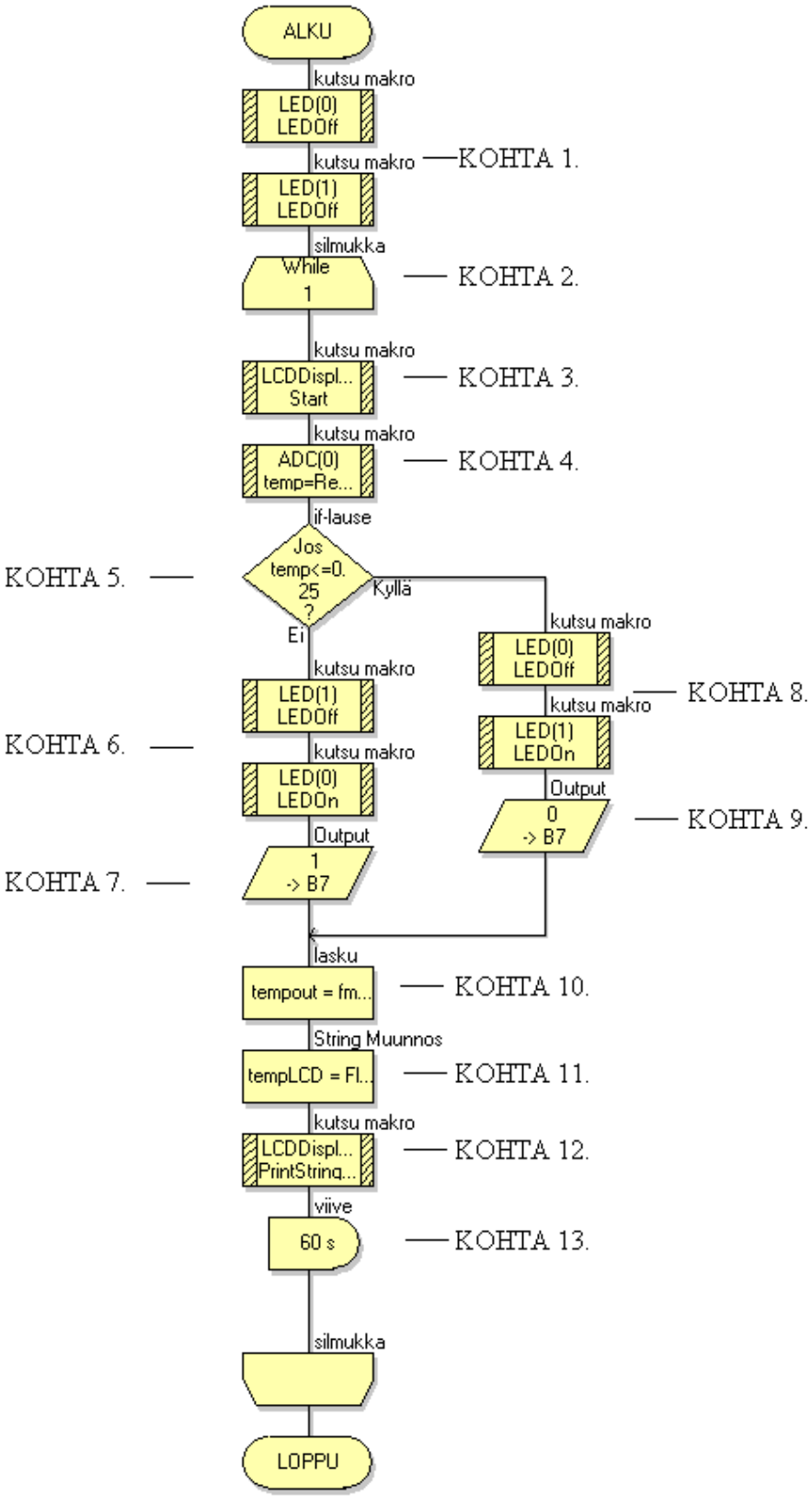
Kun varsinainen vuokaavio on valmis, testataan suunnitellun ohjelman toimintaa simulaatio tilassa. Simulaation aikana pystytään havaitsemaan valmistetun koodin toiminta. Simuloinnin aikana pystytään huomaamaan myös mahdolliset ei toivotut ominaisuudet. Myös ohjelman mahdolliset rakenteelliset virheet havaitaan. Simulointi tilan jälkeen voidaan muuttaa ohjelman parametreja, jos on tarve. Simulointi tilassa ohjelman suoritus tapahtuu lähes reaaliajassa, jolloin ohjelman todellinen suoritusaika saadaan selville.

Ennen ohjelman siirtoa piirille ohjelma tallennetaan sekä käännetään normaaliin rivi riviltä tulkittavaan muotoon. Käännösvaihtoehtoina ohjelmassa on joko C- tai assemply -kielinen käännös. Ennen ohjelman piirille latausta, ohjelma tekee valitusta ohjelmakielisestä käännöksestä vielä yhden käännöksen. Rivi riviltä tulkitta-

va ohjelmakäännös muutetaan muotoon, jota piiri ymmärtää. PIC -piireissä käännös tapahtuu hex-muotoon.

## 7.4 Ohjelma kaavion kulku

(KUVIO 7.):ssä. on esitetty ECIO-40P:lle tehty elektronisen termostaatin ohjelma, jota käytetään valmistuneen laitteen ensimmäisenä sovelluksena. Seuraavalla sivulla olevan vuokaavion kuvateksteistä olisi näkynyt vain kaksi ensimmäistä sanaa, joten ohjelman kulku on esitetty kuvan jälkeisillä sivuilla. Ohjelman C-kielinen toteutus löytyy liitteestä 3.




KUVIO 7. Ohjelman vuokaavio

Vuokaaviossa ohjelman kulku etenee seuraavasti:

KOHTA 1:
Aluksi järjestelmän porttiaktiivisuudesta ilmoittavat led-valot sammutetaan.

KOHTA 2:
Mennään ikuiseen silmukkaan jossa pää-ohjelma toimii.

KOHTA 3:
Järjestelmään liitettävän lcd-näytön toiminnot alustetaan ja mahdolliset kirjoitetut tekstit pyyhitään.

KOHTA 4:
Luetaan anturilta lämpötila-anturilta saatu jännitteenä ilmoitettu lämpötila-arvo ja muutetaan se ohjelmallisesti digitaaliseksi arvoksi, jota mikrokontrolleri ymmärtää, sekä tallennetaan arvo liukuluku muodossa ohjelmassa määritettyyn temp -muuttujaan.

KOHTA 5:
Vertaillaan temp -muuttujan arvoa ohjelmassa määritettyyn 0.25 raja-arvoon sekä tehdään valinta saadun vertailun tuloksena.

KOHTA 6:
Kohdan 5 vertailun tulos on ollut raja-arvoa pienempi. Tästä johtuen jatketaan ohjelman suorittamista kohdasta kuusi, jolloin sammutetaan piirilevylle sijoitettu D-portin bitin 3 ohjaama led sekä sytytetään piirilevyllä oleva d-portin bitin 4 ohjaama led.

KOHTA 7:
Kohdan kuusi lisäksi portin B seitsemäs bitti saa loogisen tilan 1. jolloin porttiin kytketty rele vaihtaa tilansa johtavaksi.

KOHTA 8:
Kohdan 5 vertailun tulos on ollut raja-arvoa suurempi. Tästä johtuen jatketaan ohjelman suorittamista kohdasta 8, jolloin sytytetään piirilevyllä oleva D-portin bitin 3 ohjaama led, sekä sammutetaan piirilevylle sijoitettu d-portin bitin 4 ohjaama led.

KOHTA 9:
Kohdan 8 lisäksi portin B seitsemäs bitti saa loogisen tilan 0. jolloin porttiin kytketty rele vaihtaa tilansa johtamattomaksi.

KOHTA10:
Kerrotaan temp -muuttujaan sijoitettu arvo sadalla, jolloin saadaan celsiusasteikkoa vastaavat luku-arvot sekä tallennetaan saadut lukuarvot tempout -muuttujaan.

KOHTA11:
Muunnetaan tempout muuttujan sisältö string parametriksi ja tallennetaan saatu merkkijono tempLCD -muuttujaan.

KOHTA12:
Luetaan tempLCD:n sisältämä lukuarvo ja tulostetaan se LCD-näytölle.

KOHTA13:
Asetetaan silmukan lopussa olevan viiveen pituudeksi 60 sekuntia, minkä jälkeen silmukka pyörähtää ympäri kohtaan 2.

# 8 YHTEENVETO

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa laajennuskortti ECIO40P -mikrokontrollerille. Työn tavoitteena ja vaatimuksena oli, että mikrokontrolleriin kytketty laajennuskortti yhdistelmä toimisi itsenäisenä laitekokonaisuutena ohjelmoinnin jälkeen ja ohjaisi lämpötila anturilta saatujen tietojen perusteella laajen-

nuskortin releohjaimeen kytkettyä jääkaappia termostaatin tapaan. Lopullinen laitteen kytkentä jääkaappiin ei kuulunut opinnäytetyön sisältöön. Lopullinen laitekokonaisuuden kytkentä toteutetaan Koulutuskeskus Salpauksessa lähitulevaisuudessa opetustyön muodossa.

Opinnäytetyön keskeisimpiä tavoitteita oli suunnitella yleiskäyttöinen laajennuskortti ECIO-40P mikro-ohjaimelle. Toisena keskeisenä tavoitteena oli tutustua saman tuottajan valmistamaan ohjelmointi ympäristöön ja soveltaa kyseisellä ohjelmistolla luotua ohjelmaa opinnäytetyössä.

Työn ensimmäisessä osiossa toteutettiin piirilevyn suunnittelu Cadsoft EAGLE 5.6.0 suunnitteluohjelmistolla ja valmistus perinteisellä syövytystekniikalla. Työ eteni vaiheittain, jolloin mahdolliset ongelmatilanteet havaittiin ajoissa ja ne voitiin korjata ennen seuraavaa vaihetta. Suunnittelun edetessä kytkentään tehtiin lisäyksiä, jotka muuttivat suunnittelun ja toteutuksen varsin mielenkiintoiseksi. Ohjainkortilla olevien komponenttien tilankäyttö piirilevyllä olisi voitu saada pienemmäksi käyttämällä jalallisten komponenttien tilalla pintaliitoskomponentteja. Valmistuvan laitteen sovellus kohteet ja käyttäjäkunta asettivat työssä käytettäville komponenteille omat rajoitteensa. Tästä syystä työssä päädyttiin käyttämään helposti vaihdettavia ja testattavia jalallisia komponentteja.

Työssä käytetty Cadsoft EAGLE -suunnitteluympäristön käytön uudelleenopettelu ja uusien komponenttien luonti komponenttikirjaston sovellusta käyttäen toi mukanaan omat lisähaasteensa.  Piirilevyn suunnittelutilanteessa tarkkojen tietojen puute valittavista komponenteista toi mukanaan tilankäyttö- ja sijoitteluongelmia, jotka kuitenkin voidaan ratkaista ennen laajennuslevyn viimeisen version valmistamista.

Työn toisessa osiossa oli päätarkoituksena tutustua Matrix Multimedian tuottamaan Flowcode -ohjelmistoon. Flowcode ohjelmistoa käytettiin ohjelmoitaessa ECIO-40P mikrokontrollerille asennettua mikropiiriä. Tämäkin lopputyön osio toi mukanaan omat haasteensa, koska ohjelmaan syventävä opas toimitetaan pakettiratkaisuna. Koulutuskeskus Salpaukseen mikro-ohjaimet oli tilattu ja toimitettu

kuitenkin bulk -versioina ilman syventävää ohjekirjaa. Myös kyseisen ohjelmiston ohjelmointitapa oli erikoinen. Ohjelma rakennettiin vuokaaviosta, jonka kääntäjä kääntää ensin rivi riviltä tulkittavaan muotoon ja sen jälkeen hex-muotoon jota PIC -mikrokontrollerit ymmärtävät.

Kaikki opinnäytetyölle asetetut tavoitteet saavutettiin odotusten mukaisesti. Ainoana esteenä valmiin laitteen rakentamiselle oli erikoisempien komponenttien tilaus, minkä vuoksi lopullisen piirilevyn valmistus siirtyy lähitulevaisuuteen.

Parannusehdotuksia:
Jatkossa laajennuskorttia toiminnan seuraamista jääkaapin ohjauksessa voisi parantaa suunnittelemalla ja toteuttamalla laajennuskorttiin soveltuvan LCD – näytön. Näyttö helpottaisi esimerkiksi tulkitsemaan jääkaapin sisällä olevan ilman lämpötilan. Toisena parannusehdotuksena voisi ajatella otettavan käyttöön laajennuslevyllä olevat painonapit, joilla voisi säätää termostaattiohjelmaan asetettua raja arvoa joko ylös- tai alaspäin.

LÄHTEET

Fairchild 2008. LM7805-datasheet[verkkojulkaisu]. viitattu [29.10.2009]. Saatavissa: http://www.fairchildsemi.com/ds/LM%2FLM7805.pdf

Matrix multimedia 2008. ECIO-40P-datasheet[verkkojulkaisu]. Viitattu [18.10.2009]. Saatavissa: http://www.matrixmultimedia.com/datasheets/ECIO-60-2.pdf

Microchip 2009. PIC18F4455-datasheet[verkkojulkaisu]. Viitattu [1.11.2009]. Saatavissa: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf

Omron 2008. G2RL-1-E-datasheet[verkkojulkaisu]. Viitattu [20.10.2009]. Saatavissa: http://downloads.omron.fi/OCB/Products/Relays/PCB%20Power%20Relays/Up%20to%2030A/G2RL/J117/J117-E2-04A-X.pdf

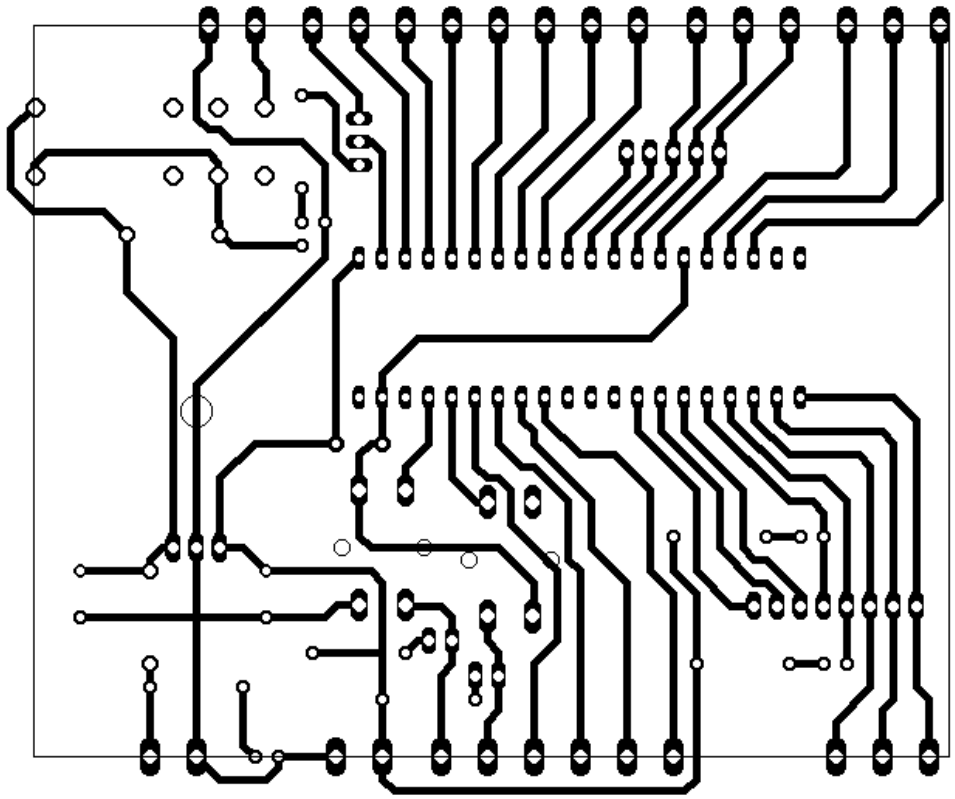
National 2005. LM45-datasheet[verkkojulkaisu]. Viitattu[20.11.2009]. Saatavissa: http://www.national.com/ds.cgi/LM/LM45.pdf

KUVIO 2. ECIO-40P[verkkojulkaisu]. viitattu[16.11.2009]. Saatavissa: http://microcontrollershop.com/product_info.php?products_id=2394

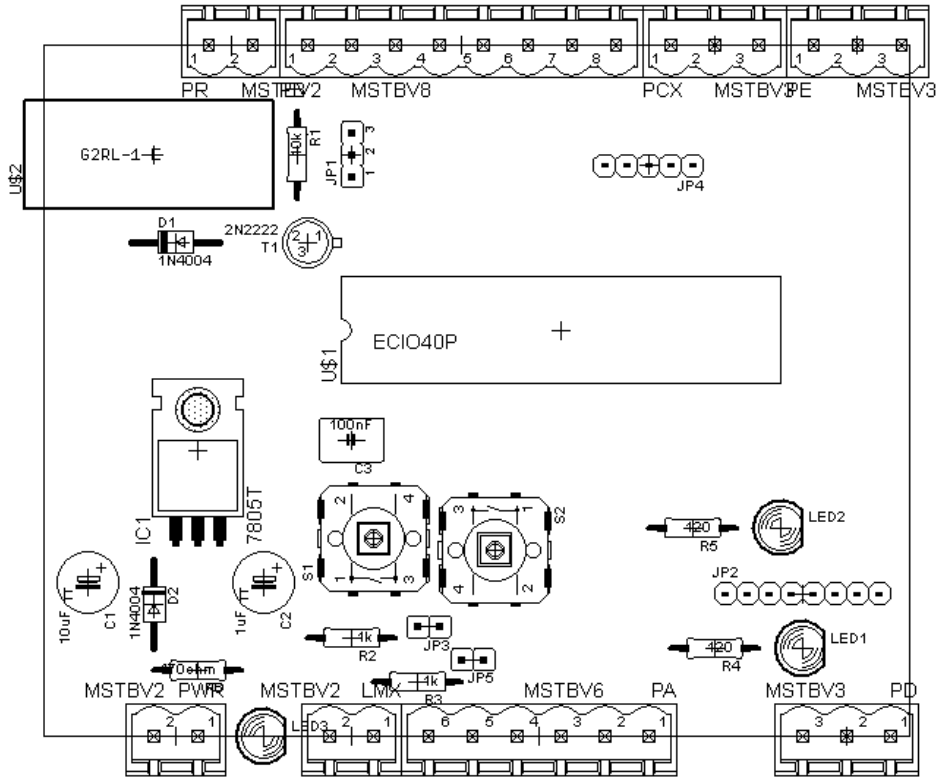
Halytin.com 2009. Releen suojadiodi[verkkojulkaisu]. viitattu[16.11.2009]. Saatavissa: www.halytin.com/catalog/asennusohjeet.php

# LIITTEET

## LIITE 1 laajennuskortin versio 2 syövytyskuva



LIITE 2 laajennuskortin 2 osasijoittelukuva



LIITE 3 C-kielinen ohjelma koodi

```
//*****************************************************************************************
//**
//**  File name:     termistori.c
//**  Generated by:  Flowcode v4.0.0.53
//**  Date:          Thursday, November 19, 2009 22:19:17
//**  Licence:       Demo
//**
//**        ***DEMO VERSION***
//**
//**
//**     EI KAUPALLISEEN KÄYTTÖÖN
//**
//**  http://www.matrixmultimedia.com
//*****************************************************************************************


#define MX_PIC

//Määrittää mikro-ohjaimen
#define P18F4455
#define MX_EE
#define MX_EE_TYPE3
#define MX_EE_SIZE 256
#define MX_SPI
#define MX_SPI_BCB
#define MX_SPI_SDI 0
#define MX_SPI_SDO 7
#define MX_SPI_SCK 1
#define MX_UART
#define MX_UART_C
#define MX_UART_TX 6
#define MX_UART_RX 7
#define MX_I2C
#define MX_MI2C
#define MX_I2C_B
#define MX_I2C_SDA 0
#define MX_I2C_SCL 1
#define MX_PWM
#define MX_PWM_CNT 2
#define MX_PWM_TRIS1 trisc
#define MX_PWM_1 2
#define MX_PWM_TRIS2 trisc
#define MX_PWM_2 1
#define MX_PWM_TRIS2a trisb
#define MX_PWM_2a 3

//funktiot
#include <system.h>
#pragma CLOCK_FREQ 48000000

//Konfigurointi data
#pragma DATA 0x300000, 0x20
#pragma DATA 0x300001, 0xe
#pragma DATA 0x300002, 0x3e
#pragma DATA 0x300003, 0x1e
#pragma DATA 0x300004, 0x0
#pragma DATA 0x300005, 0x81
#pragma DATA 0x300006, 0x81
#pragma DATA 0x300007, 0x0
```

```
#pragma DATA 0x300008, 0xf
#pragma DATA 0x300009, 0x80
#pragma DATA 0x30000a, 0xf
#pragma DATA 0x30000b, 0xa0
#pragma DATA 0x30000c, 0xf
#pragma DATA 0x30000d, 0x0

//Sisäiset toiminnot
#include "C:\Program Files\Matrix Multimedia\Flowcode V4\FCD\internals.h"

//Makro-funktion kuvaukset


//Muuttujakuvaukset
#define FCSZ_TEMPLCD 20
float FCV_TEMP;
short FCV_KIERROKSET;
char FCV_TEMPLCD[FCSZ_TEMPLCD];
float FCV_TEMPOUT;


//Defines:

/**** Macro Substitutions ****
portd = LED Port Register
trisd = LED Data Direction Register
8 = LED Pin Mask
1 = LED Active Polarity
******************************/



//LED0: //Makro-funktion kuvaukset

void FCD_LED0_LEDOn();
void FCD_LED0_LEDOff();
//Defines:

/**** Macro Substitutions ****
3 = Which ADC Channel
20 = Acquisition time
3 = Conversion Speed
1 = VRef+ Option
100 = VRef Voltage x 0.01V
******************************/



//ADC0: //Makro-funktion kuvaukset

void FCD_ADC0_SampleADC();
char FCD_ADC0_ReadAsByte();
short FCD_ADC0_ReadAsInt();
float FCD_ADC0_ReadAsVoltage();
void FCD_ADC0_ReadAsString(char* FCR_RETVAL, char FCR_RETVAL_SIZE, char
NumBytes);
//Defines:

/**** Macro Substitutions ****
```

```
portd = LED Port Register
trisd = LED Data Direction Register
16 = LED Pin Mask
1 = LED Active Polarity
*****************************/



//LED1: //Makro-funktion kuvaukset

void FCD_LED1_LEDOn();
void FCD_LED1_LEDOff();
//Defines:

/**** Macro Substitutions ****
portb = D1 Port
trisb = D1 Data Direction
portb = D2 Port
trisb = D2 Data Direction
portb = D3 Port
trisb = D3 Data Direction
portb = D4 Port
trisb = D4 Data Direction
portb = RS Port
trisb = RS Data Direction
portb = E Port
trisb = E Data Direction
0 = Data 1_Pin
1 = Data 2 Pin
2 = Data 3 Pin
3 = Data 4 Pin
4 = RS Pin
5 = Enable Pin
LCD_263224 = Unique Component Reference Number
1 = Row Count
16 = Column Count
*****************************/

                //component connections
                #define LCD_263224_PORT0    portb
                #define LCD_263224_TRIS0    trisb
                #define LCD_263224_PORT1    portb
                #define LCD_263224_TRIS1    trisb
                #define LCD_263224_PORT2    portb
                #define LCD_263224_TRIS2    trisb
                #define LCD_263224_PORT3    portb
                #define LCD_263224_TRIS3    trisb
                #define LCD_263224_PORT4    portb
                #define LCD_263224_TRIS4    trisb
                #define LCD_263224_PORT5    portb
                #define LCD_263224_TRIS5    trisb
                #define LCD_263224_BIT0                 0
                #define LCD_263224_BIT1                 1
                #define LCD_263224_BIT2                 2
                #define LCD_263224_BIT3                 3
                #define LCD_263224_RS      4
                #define LCD_263224_E       5
                #define LCD_263224_ROWCNT               1
                #define LCD_263224_COLCNT               16
```

```
#ifdef _BOOSTC
  #define LCD_263224_DELAY   delay_10us(10)
#endif
#ifdef _C2C_
  #define LCD_263224_DELAY   delay_us(100)
#endif
#ifndef LCD_263224_DELAY
  #define LCD_263224_DELAY   delay_us(100)
#endif



//LCDDisplay0: //Makro-funktion kuvaukset

void FCD_LCDDisplay0_RawSend(char in, char mask);
void FCD_LCDDisplay0_Start();
void FCD_LCDDisplay0_Clear();
void FCD_LCDDisplay0_PrintASCII(char Character);
void FCD_LCDDisplay0_Command(char in);
void FCD_LCDDisplay0_Cursor(char x, char y);
void FCD_LCDDisplay0_PrintNumber(short Number);
void FCD_LCDDisplay0_PrintString(char* String, char MSZ_String);
void FCD_LCDDisplay0_ScrollDisplay(char Direction, char Num_Positions);
void FCD_LCDDisplay0_ClearLine(char Line);
void FCD_LCDDisplay0_RAM_Write(char nIdx, char d0, char d1, char d2, char d3, char
d4,
 char d5, char d6, char d7);



//LED0: //Makro-toteutukset


void FCD_LED0_LEDOn()
{

                        trisd = trisd & ~8;
        //Convert pin to output

                        if( 1 )
        //Active high polarity
                                portd = portd | 8;
                        else
        //Active low polarity
                                portd = portd & ~8;

}

void FCD_LED0_LEDOff()
{

                        trisd = trisd & ~8;
        //Convert pin to output

                        if( 1 )
        //Active high polarity
                                portd = portd & ~8;
                        else
        //Active low polarity
                                portd = portd | 8;
```

```
}


//ADC0: //Makro-toteutukset


void FCD_ADC0_SampleADC()
{

                /*******Supported De-
vices***************************************************************
                // 18F2220, 18F2221, 18F2320, 18F2321, 18F24J10, 18F2410,
                // 18F2420, 18F2423, 18F2450,
                // 18F2455, 18F2480, 18F25J10, 18F2510, 18F2515, 18F2520,
                // 18F2523, 18F2525, 18F2550,
                // 18F2580, 18F2585, 18F2610, 18F2620, 18F2680, 18F4220,
                //18F4221, 18F4320, 18F4321,
                // 18F44J10, 18F4410, 18F4420, 18F4423, 18F4450, 18F4455,
                //18F4480, 18F45J10, 18F4510,
                // 18F4515, 18F4520, 18F4523, 18F4525, 18F4550, 18F4580,
                //18F4585, 18F4610, 18F4620,
                // 18F4680, 18F4682, 18F4685
                ************************************************************************************
****/

                                #define MX_ADC_CHANNEL          3
                                #define MX_ADC_SAMP_TIME        20
                                #define MX_ADC_CONV_SP          3
                                #define MX_ADC_VREF_OPT                 1

                                //set up ADC conversion
                                char old_tris, cnt;
                                adcon2 = MX_ADC_CONV_SP & 0x07;

                                //find appropriate bit
                                #if (MX_ADC_CHANNEL == 0)
                                        #define MX_ADC_TRIS_REG  trisa
                                        #define MX_ADC_TRIS_MSK  0x01
                                        adcon1 = 0x0E;
                                #endif
                                #if (MX_ADC_CHANNEL == 1)
                                        #define MX_ADC_TRIS_REG  trisa
                                        #define MX_ADC_TRIS_MSK  0x02
                                        adcon1 = 0x0D;
                                #endif
                                #if (MX_ADC_CHANNEL == 2)
                                        #define MX_ADC_TRIS_REG  trisa
                                        #define MX_ADC_TRIS_MSK  0x04
                                        adcon1 = 0x0C;
                                #endif
                                #if (MX_ADC_CHANNEL == 3)
                                        #define MX_ADC_TRIS_REG  trisa
                                        #define MX_ADC_TRIS_MSK  0x08
                                        adcon1 = 0x0B;
                                        #if (MX_ADC_VREF_OPT != 0)
                                        #pragma error "Target device is currently
using AN3 for VREF+"
                                        #endif
                                #endif
```

```
#if (MX_ADC_CHANNEL == 4)
                #define MX_ADC_TRIS_REG  trisa
                #define MX_ADC_TRIS_MSK  0x20
                adcon1 = 0x0A;
#endif
#if (MX_ADC_CHANNEL == 5)
                #define MX_ADC_TRIS_REG  trise
                #define MX_ADC_TRIS_MSK  0x01
                adcon1 = 0x09;
#endif
#if (MX_ADC_CHANNEL == 6)
                #define MX_ADC_TRIS_REG  trise
                #define MX_ADC_TRIS_MSK  0x02
                adcon1 = 0x08;
#endif
#if (MX_ADC_CHANNEL == 7)
                #define MX_ADC_TRIS_REG  trise
                #define MX_ADC_TRIS_MSK  0x04
                adcon1 = 0x07;
#endif
#if (MX_ADC_CHANNEL == 8)
                #define MX_ADC_TRIS_REG  trisb
                #define MX_ADC_TRIS_MSK  0x02
                adcon1 = 0x06;
#endif
#if (MX_ADC_CHANNEL == 9)
                #define MX_ADC_TRIS_REG  trisb
                #define MX_ADC_TRIS_MSK  0x10
                adcon1 = 0x05;
#endif
#if (MX_ADC_CHANNEL == 10)
                #define MX_ADC_TRIS_REG  trisb
                #define MX_ADC_TRIS_MSK  0x02
                adcon1 = 0x04;
#endif
#if (MX_ADC_CHANNEL == 11)
                #define MX_ADC_TRIS_REG  trisb
                #define MX_ADC_TRIS_MSK  0x10
                adcon1 = 0x03;
#endif
#if (MX_ADC_CHANNEL == 12)
                #define MX_ADC_TRIS_REG  trisb
                #define MX_ADC_TRIS_MSK  0x01
                adcon1 = 0x02;
#endif

//sanity check
#ifndef MX_ADC_TRIS_REG
                #pragma error
        "ADC Type 13 conversion code error - please contact
technical support"
#endif

//assign VREF functionality
#if (MX_ADC_VREF_OPT != 0)
                set_bit(adcon1, VCFG0);
#endif

//store old tris value, and set the i/o pin as an input
old_tris = MX_ADC_TRIS_REG;
```

```
                                MX_ADC_TRIS_REG = MX_ADC_TRIS_REG |
MX_ADC_TRIS_MSK;

                                //turn ADC on
                                adcon0 = 0x01 | (MX_ADC_CHANNEL << 2);

                                //wait the acquisition time
                                cnt = 0;
                                while (cnt < MX_ADC_SAMP_TIME) cnt++;

                                //begin conversion and wait until it has finished
                                adcon0 = adcon0 | 0x02;
                                while (adcon0 & 0x02);

                                //restore old tris value, and reset adc registers
                                MX_ADC_TRIS_REG = old_tris;
                                adcon1 = 0x0f;
                                adcon0 = 0x00;

                                #undef MX_ADC_TRIS_REG
                                #undef MX_ADC_TRIS_MSK
                                #undef MX_ADC_SAMP_TIME
                                #undef MX_ADC_CHANNEL
                                #undef MX_ADC_CONV_SP
                                #undef MX_ADC_VREF_OPT

}

char FCD_ADC0_ReadAsByte()
{

                                FCD_ADC0_SampleADC();

                                return adresh;

}

short FCD_ADC0_ReadAsInt()
{

                                short iRetVal;

                                FCD_ADC0_SampleADC();

                                iRetVal = (adresh << 2);
                                iRetVal = iRetVal | (adresl >> 6);

                                return (iRetVal);

}

float FCD_ADC0_ReadAsVoltage()
{


                                int iSample;
                                float fSample, fVoltage, fVperDiv;

                                #define MX_ADC_VREF_V                   100
```

```
                                iSample = FCD_ADC0_ReadAsInt();
//Read as 10-bit Integer

                                fVoltage = float32_from_int32(MX_ADC_VREF_V);
//Convert reference voltage count to floating point (0 - 500 x 10mV)
                                fVoltage = float32_mul(fVoltage, 0.01);
//Convert reference voltage count to actual voltage (0 - 5)
                                fVperDiv = float32_mul(fVoltage, 0.000976);  //Convert ac-
tual voltage to voltage per division (VRef / 1024)
                                fSample = float32_from_int32(iSample);
//Convert to floating point variable
                                fVoltage = float32_mul(fSample, fVperDiv);    //Calculate
floating point voltage

                                #undef MX_ADC_VREF_V
                                return (fVoltage);

}

void FCD_ADC0_ReadAsString(char* FCR_RETVAL, char FCR_RETVAL_SIZE,
 char NumBytes)
{

                                float fVoltage;

                                if (NumBytes > FCR_RETVAL_SIZE)
                                                NumBytes = FCR_RETVAL_SIZE;

                                fVoltage = FCD_ADC0_ReadAsVoltage();
                FCI_FLOAT_TO_STRING(fVoltage, 2, FCR_RETVAL, NumBytes);
                //Convert to String

}



//LED1: //Makro-toteutukset


void FCD_LED1_LEDOn()
{

                                trisd = trisd & ~16;
                //Convert pin to output

                                if( 1 )
                //Active high polarity
                                                portd = portd | 16;
                                else
                //Active low polarity
                                                portd = portd & ~16;

}

void FCD_LED1_LEDOff()
{

                                trisd = trisd & ~16;
                //Convert pin to output
```

```
                        if( 1 )
                //Active high polarity
                                        portd = portd & ~16;
                        else
                //Active low polarity
                                        portd = portd | 16;

}


//LCDDisplay0: //Makro-toteutukset


void FCD_LCDDisplay0_RawSend(char in, char mask)
{
                        unsigned char pt;

                        clear_bit(LCD_263224_PORT0, LCD_263224_BIT0);
                        clear_bit(LCD_263224_PORT1, LCD_263224_BIT1);
                        clear_bit(LCD_263224_PORT2, LCD_263224_BIT2);
                        clear_bit(LCD_263224_PORT3, LCD_263224_BIT3);
                        clear_bit(LCD_263224_PORT4, LCD_263224_RS);
                        clear_bit(LCD_263224_PORT5, LCD_263224_E);
                        pt = ((in >> 4) & 0x0f);
                        if (pt & 0x01)
                           set_bit(LCD_263224_PORT0, LCD_263224_BIT0);
                        if (pt & 0x02)
                           set_bit(LCD_263224_PORT1, LCD_263224_BIT1);
                        if (pt & 0x04)
                           set_bit(LCD_263224_PORT2, LCD_263224_BIT2);
                        if (pt & 0x08)
                           set_bit(LCD_263224_PORT3, LCD_263224_BIT3);
                        if (mask)
                           set_bit(LCD_263224_PORT4, LCD_263224_RS);
                        LCD_263224_DELAY;
                        set_bit (LCD_263224_PORT5, LCD_263224_E);
                        LCD_263224_DELAY;
                        clear_bit (LCD_263224_PORT5, LCD_263224_E);
                        pt = (in & 0x0f);
                        LCD_263224_DELAY;
                        clear_bit(LCD_263224_PORT0, LCD_263224_BIT0);
                        clear_bit(LCD_263224_PORT1, LCD_263224_BIT1);
                        clear_bit(LCD_263224_PORT2, LCD_263224_BIT2);
                        clear_bit(LCD_263224_PORT3, LCD_263224_BIT3);
                        clear_bit(LCD_263224_PORT4, LCD_263224_RS);
                        clear_bit(LCD_263224_PORT5, LCD_263224_E);
                        if (pt & 0x01)
                           set_bit(LCD_263224_PORT0, LCD_263224_BIT0);
                        if (pt & 0x02)
                           set_bit(LCD_263224_PORT1, LCD_263224_BIT1);
                        if (pt & 0x04)
                           set_bit(LCD_263224_PORT2, LCD_263224_BIT2);
                        if (pt & 0x08)
                           set_bit(LCD_263224_PORT3, LCD_263224_BIT3);
                        if (mask)
                           set_bit(LCD_263224_PORT4, LCD_263224_RS);
                        LCD_263224_DELAY;
                        set_bit (LCD_263224_PORT5, LCD_263224_E);
                        LCD_263224_DELAY;
                        clear_bit (LCD_263224_PORT5, LCD_263224_E);
```

```
                LCD_263224_DELAY;
}

void FCD_LCDDisplay0_Start()
{

                clear_bit(LCD_263224_TRIS0, LCD_263224_BIT0);
                clear_bit(LCD_263224_TRIS1, LCD_263224_BIT1);
                clear_bit(LCD_263224_TRIS2, LCD_263224_BIT2);
                clear_bit(LCD_263224_TRIS3, LCD_263224_BIT3);
                clear_bit(LCD_263224_TRIS4, LCD_263224_RS);
                clear_bit(LCD_263224_TRIS5, LCD_263224_E);

                Wdt_Delay_Ms(12);

                FCD_LCDDisplay0_RawSend(0x33, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x33, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x32, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x2c, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x06, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x0c, 0);
                Wdt_Delay_Ms(2);

                //clear the display
                FCD_LCDDisplay0_RawSend(0x01, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x02, 0);
                Wdt_Delay_Ms(2);

}

void FCD_LCDDisplay0_Clear()
{

                FCD_LCDDisplay0_RawSend(0x01, 0);
                Wdt_Delay_Ms(2);
                FCD_LCDDisplay0_RawSend(0x02, 0);
                Wdt_Delay_Ms(2);

}

void FCD_LCDDisplay0_PrintASCII(char Character)
{

                FCD_LCDDisplay0_RawSend(Character, 0x10);

}

void FCD_LCDDisplay0_Command(char in)
{

                FCD_LCDDisplay0_RawSend(in, 0);
                Wdt_Delay_Ms(2);

}
```

```
void FCD_LCDDisplay0_Cursor(char x, char y)
{
		#if (LCD_263224_ROWCNT == 1)
		  y=0x80;
		#endif

		#if (LCD_263224_ROWCNT == 2)
			if (y==0)
				y=0x80;
			else
				y=0xc0;
		#endif

		#if (LCD_263224_ROWCNT == 4)
			if (y==0)
				y=0x80;
			else if (y==1)
				y=0xc0;

			#if (LCD_263224_COLCNT == 16)
				else if (y==2)
					y=0x90;
				else
					y=0xd0;
			#endif

			#if (LCD_263224_COLCNT == 20)
				else if (y==2)
					y=0x94;
				else
					y=0xd4;
			#endif
		#endif

			FCD_LCDDisplay0_RawSend(y+x, 0);
			Wdt_Delay_Ms(2);

}

void FCD_LCDDisplay0_PrintNumber(short Number)
{

			short tmp_int;
			char tmp_byte;
			if (Number < 0)
			{
				FCD_LCDDisplay0_RawSend('-', 0x10);
				Number = 0 - Number;
			}

			tmp_int = Number;
			if (Number >= 10000)
			{
				tmp_byte = tmp_int / 10000;
				FCD_LCDDisplay0_RawSend('0' + tmp_byte,
0x10);

				while (tmp_byte > 0)
				{
					tmp_int = tmp_int - 10000;
```

```
                                                tmp_byte--;
                                        }
                                }
                                if (Number >= 1000)
                                {
                                        tmp_byte = tmp_int / 1000;
                                        FCD_LCDDisplay0_RawSend('0' + tmp_byte,
0x10);

                                        while (tmp_byte > 0)
                                        {
                                                tmp_int = tmp_int - 1000;
                                                tmp_byte--;
                                        }
                                }
                                if (Number >= 100)
                                {
                                        tmp_byte = tmp_int / 100;
                                        FCD_LCDDisplay0_RawSend('0' + tmp_byte,
0x10);

                                        while (tmp_byte > 0)
                                        {
                                                tmp_int = tmp_int - 100;
                                                tmp_byte--;
                                        }
                                }
                                if (Number >= 10)
                                {
                                        tmp_byte = tmp_int / 10;
                                        FCD_LCDDisplay0_RawSend('0' + tmp_byte,
0x10);

                                        while (tmp_byte > 0)
                                        {
                                                tmp_int = tmp_int - 10;
                                                tmp_byte--;
                                        }
                                }
                                FCD_LCDDisplay0_RawSend('0' + tmp_int, 0x10);

}

void FCD_LCDDisplay0_PrintString(char* String, char MSZ_String)
{

                                char idx;
                                for (idx=0; idx<MSZ_String; idx++)
                                {
                                        if (String[idx]==0)
                                        {
                                                break;
                                        }
                                        FCD_LCDDisplay0_RawSend(String[idx],
0x10);
                                }

}

void FCD_LCDDisplay0_ScrollDisplay(char Direction, char Num_Positions)
{
```

```
                        char cmd = 0;
                        char count;

                        //Choose the direction
                        switch (Direction)
                        {
                                case 0:
                                case 'l':
                                case 'L':

                                        cmd = 0x18;
                                        break;

                                case 1:
                                case 'r':
                                case 'R':

                                        cmd = 0x1C;
                                        break;

                                default:
                                        break;
                        }

                        //If direction accepted then scroll the specified amount
                        if (cmd)
                        {
                                for (count = 0; count < Num_Positions;
count++)

                FCD_LCDDisplay0_Command(cmd);
                        }

}

void FCD_LCDDisplay0_ClearLine(char Line)
{

                        char count;
                        char rowcount;

                        //Define number of columns per line
                        #if (LCD_263224_ROWCNT == 1)
                                rowcount=80;
                        #endif

                        #if (LCD_263224_ROWCNT == 2)
                                rowcount=40;
                        #endif

                        #if (LCD_263224_ROWCNT == 4)
                                #if (LCD_263224_COLCNT == 16)
                                        rowcount=16;
                                #endif
                                #if (LCD_263224_COLCNT == 20)
                                        rowcount=20;
                                #endif
                        #endif

                        //Start at beginning of the line
```

```
                                FCD_LCDDisplay0_Cursor (0, Line);

                                //Send out spaces to clear line
                                for (count = 0; count < rowcount; count++)
                                        FCD_LCDDisplay0_RawSend(' ', 0x10);

                                //Move back to the beginning of the line.
                                FCD_LCDDisplay0_Cursor (0, Line);

}

void FCD_LCDDisplay0_RAM_Write(char nIdx, char d0, char d1, char d2, char d3, char
d4,
 char d5, char d6, char d7)
{
                //set CGRAM address
                FCD_LCDDisplay0_RawSend(64 + (nIdx << 3), 0);
                delay_ms(2);

                //write CGRAM data
                FCD_LCDDisplay0_RawSend(d0, 0x10);
                FCD_LCDDisplay0_RawSend(d1, 0x10);
                FCD_LCDDisplay0_RawSend(d2, 0x10);
                FCD_LCDDisplay0_RawSend(d3, 0x10);
                FCD_LCDDisplay0_RawSend(d4, 0x10);
                FCD_LCDDisplay0_RawSend(d5, 0x10);
                FCD_LCDDisplay0_RawSend(d6, 0x10);
                FCD_LCDDisplay0_RawSend(d7, 0x10);

                //Clear the display
                FCD_LCDDisplay0_RawSend(0x01, 0);
                delay_ms(2);
                FCD_LCDDisplay0_RawSend(0x02, 0);
                delay_ms(2);
}

//Makro-toteutukset

void main()
{

        //alustus
        adcon1 = 0x0F;
        ucfg = 0x08;


        //Keskeytyksen alustuskoodi



        //Kommentti:
        //KOHTA 1.


        //Call Component Macro
        //Kutsu komponenttimakro: LED(0)::LEDOff
        FCD_LED0_LEDOff();


        //Call Component Macro
        //Kutsu komponenttimakro: LED(1)::LEDOff
```

```
FCD_LED1_LEDOff();


//Kommentti:
//KOHTA 2.


//silmukka
//Silmukka: While 1
while (1)
{
//Kommentti:
//KOHTA 3.


//Call Component Macro
//Kutsu komponenttimakro: LCDDisplay(0)::Start
FCD_LCDDisplay0_Start();


//Kommentti:
//KOHTA 4.


//Call Component Macro
//Kutsu komponenttimakro: temp=ADC(0)::ReadAsVoltage
FCV_TEMP = FCD_ADC0_ReadAsVoltage();


//Kommentti:
//KOHTA 5.


//Decision
//Päätös: temp<=0.25?
if (FCV_TEMP<=0.25)
{
                //Kommentti:
                //KOHTA 8.


                //Call Component Macro
                //Kutsu komponenttimakro: LED(0)::LEDOff
                FCD_LED0_LEDOff();


                //Kutsu komponenttimakro
                //Kutsu komponenttimakro: LED(1)::LEDOn
                FCD_LED1_LEDOn();


                //Kommentti:
                //KOHTA 9.


                //Output
                //Lähtö: 0 -> B7
                trisb = trisb & 0x7f;
                if (0)
                                portb = (portb & 0x7f) | 0x80;
                else
```

```
                                        portb = portb & 0x7f;


                                } else {
                                //Kommentti:
                                //KOHTA 6.


                                //Kutsu komponenttimakro
                                //Kutsu komponenttimakro: LED(1)::LEDOff
                                FCD_LED1_LEDOff();


                                //Call Component Macro
                                //Kutsu komponenttimakro: LED(0)::LEDOn
                                FCD_LED0_LEDOn();


                                //Kommentti:
                                //KOHTA 7


                                //Output
                                //Lähtö: 1 -> B7
                                trisb = trisb & 0x7f;
                                if (1)
                                        portb = (portb & 0x7f) | 0x80;
                                else
                                        portb = portb & 0x7f;


                                }

                        //Kommentti:
                        //KOHTA 10.


                        //Calculation
                        //Laskutoimitus:
                        //  tempout = fmul(temp,100.00)
                        FCV_TEMPOUT = float32_mul(FCV_TEMP,100.00);


                        //Kommentti:
                        //KOHTA 11.


                        //String Manipulation
                        //Merkkijonon käsittely:
                        //  tempLCD = FloatToString$(tempout)
                        FCI_FLOAT_TO_STRING(FCV_TEMPOUT,6,FCV_TEMPLCD,FCSZ_TE
MPLCD);


                        //Kommentti:
                        //KOHTA 12.


                        //Call Component Macro
                        //Kutsu komponenttimakro: LCDDisplay(0)::PrintString(tempLCD)
```

```
                FCD_LCDDisplay0_PrintString(FCV_TEMPLCD,FCSZ_TEMPLCD);


                //Kommentti:
                //KOHTA 13.


                //Delay
                //Viive: 60 s
                delay_s(60);


                }


                mainendloop: goto mainendloop;
}

void interrupt(void)
{
}
```