PIIRILEVYPORAN OHJAUSMODUULI

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikka
Tietokone-elektroniikka
Opinnäytetyö
Syksy 2006
Jani Kalervo Kuhlman

Lahden ammattikorkeakoulu

Tietotekniikan koulutusohjelma

KUHLMAN, JANI KALERVO: Piirilevyporan ohjausmoduuli

Tietokone-elektroniikan opinnäytetyö, 35 sivua, 18 liitesivua

Syksy 2006

TIIVISTELMÄ

Tässä opinnäytetyössä kehitettiin ohjainmoduuli ajamaan askelmoottorikäytöistä piirilevyporamoduulia. Työn aihe on lähtöisin Lahden ammatti-instituutista.

Piirilevyn valmistuksessa reikien poraaminen käsin on aikaa ja tarkkuutta vaativaa. Siksi on järkevää automatisoida kyseinen tuotantovaihe, koska jo piirilevyn layoutin suunnitteluvaiheessa saadaan valmiina koordinaatit, joihin reiät porataan. Näin ollen mikrokontrolleriohjatun moduulin käyttäminen tehtävässä nopeuttaa tuotantovaihetta ja parantaa porattavien reikien kohdistustarkkuutta.

Mikrokontrolleri soveltuu tehtävään hyvin, sillä se on edullinen ja siinä on integroituna riittävä määrä I/O-portteja. Se sisältää mm. tarvittavan muistikapasiteetin, jolloin oheiskomponenttien määrä jää mahdollisimman pieneksi ja säästetään myös piirilevypinta-alaa.

Tämän opinnäytetyön tuloksena saatiin piirilevyporan ohjainmoduuli, joka soveltuu hyvin myös muihin askelmoottoriohjattuihin järjestelmiin pienen kokonsa ja helpon ohjattavuutensa vuoksi. Myös ohjaustoimintoihin on tarvittaessa helppo tehdä muutoksia, sillä järjestelmä on ohjelmoitu C-kielellä. Mikrokontrollerin ohjelmointi onnistuu myös ilman sen irrotusta ohjainmoduulista.

Asiasanat: mikrokontrolleri, moduuli, ohjattavuus

Lahti University of Applied Sciences

Faculty of Technology

KUHLMAN, JANI KALERVO: Microcontroller guided driver module for drilling system of printed circuit boards

Bachelor's thesis in computer electronics, 35 pages, 18 appendices

Autumn 2006

ABSTRACT

The objective of this thesis was to design and build a microcontroller-guided control module for driving a drilling machine in circuit board manufacture. The work was commissioned by Lahti Vocational Institute.

Drilling of holes in the manufacture of circuit board demands time and accuracy. That is why it is sensible to automate this production stage, because already in the designing of the circuit board layout we get coordinates for drilling each hole. Using a microcontroller-guided control module speeds up the production stage and increases the accuracy of drilling the holes.

A microcontroller suits the task well, because it is cheap and there are enough integrated I/O Ports. It also includes enough memory capacity, so the need of external components is as small as possible and less space is required on the circuit board.

In this thesis the microcontroller-guided control module was designed for a drilling machine but it can also be used in other stepper motor driven systems because its size is small and it is easy to drive. It is also easy to make changes to the control functions, because the whole system is programmed with C language. The programming of the microcontroller is also possible without removing it from the circuit board.

Keywords: microcontroller, module, control

SISÄLLYS

1 JOHDANTO

Työn tarkoitus on suunnitella ja toteuttaa piirilevyporan ohjausjärjestelmä, jolla on tarkoitus ohjata valmista piirilevypora-moduulia. Piirilevypora-moduulilla on tarkoitus suorittaa oppilaskäyttöön tulevien piirilevyjen poraus sekä käyttää järjestelmää mahdollisena opetusmateriaalina. Opinnäytetyön aihe on lähtöisin Lahden ammatti-instituutissa toimivalta Jarmo Salolta, ja se on jaettu kahteen erilliseen opinnäytetyöhön: käyttöliittymän suunnittelu ja toteutus sekä ohjainmoduulin suunnittelu ja toteutus. Työ tulee Lahden ammatti-instituutin käyttöön.

Käyttöliittymä on suunniteltu toimimaan Microsoft Windows-ympäristössä ja se on ohjelmoitu Microsoft Visual C++ 6 -kääntäjällä. Tämän käyttöliittymän on tarkoitus avata piirilevynsuunnitteluohjelmalla (Pads PowerPCB) luotu drl-tiedosto, joka sisältää piirilevylle porattavien reikien koordinaatit. Tämän jälkeen se muuntaa koordinaatit askelmääriksi suhteessa origoon ja järjestää siten, että siirryttävä matka on mahdollisimman pieni. Sitten ohjelma lähettää askelmäärät tietokoneen sarjaportin (RS-232) välityksellä mikrokontrollerimoduulille.

Paikkatietojen perusteella ohjainmoduuli käyttää maksimissaan kolmea askelmoottoria, jotka ovat piirilevyporan x-, y- ja z-akselit. Piirilevyporan mekaanisten ominaisuuksien vuoksi saavutetaan 0,02 mm:n toistotarkkuus ja liikeradat rajoittuvat seuraaviin ulottuvuuksiin x = 300 mm, y = 300 mm ja z = 75 mm. Porausprosessi ei ole herkkä tönäisyistä aiheutuvalle kohdistusvirheelle, sillä ohjattavien kuularuuvien tilaa voidaan seurata reaaliaikaisesti optisilla pulssiantureilla, jolloin syntyy paikkatiedon takaisinkytkentä. Näin ohjainmoduuli kykenee korjaamaan mahdolliset poikkeamat kohdistuksessa. Kunkin ulottuvuuden askelmäärä muuttujat ovat 16-bittisiä, joten ohjainmoduulissa riittäisi kyseisellä toistotarkkuudella resursseja suurempiinkin liikeratoihin.

2 PIIRILEVYPORA

Lähdin toteuttamaan ohjausta Arvo Saarisen vuonna 1994 tehdyn päättötyön pohjalta, vaikka hänen suunnittelemansa piirilevyporan ohjausjärjestelmän tekniikka on vanhentunutta ja nykyajan vaatimuksiin fyysisesti liian suuri sekä vaikeasti ohjelmoitavissa. Itse piirilevypora-osio on täysin käyttökelpoinen. Piirilevyporassa on x, y-tasossa askelmoottoreilla liikutettava pöytä, johon piirilevyaihio kiinnitetään. Lisäksi poramoduulissa on askelmoottorikäyttöinen z-akseli, johon voidaan kiinnittää pora. Piirilevyn työstöalue on 300 * 300 mm, ja poraa voidaan liikuttaa pystysuunnassa 75 mm. Toistotarkkuudet ovat askelmoottorin askelmäärästä/kierros ja kuularuuvien noususta johtuen 0,02 mm/askel.

Piirilevyporan tasoja x, y, z liikutetaan kuularuuvilla, jota pyöritetään askelmoottorilla. Kuularuuvin nousu on 4 mm/kierros ja aksiaalinen välys 0,04 - 0,2 mm, mutta välys voidaan poistaa täysin esikiristämällä kaksi kuulamutteria toisiaan vasten. Alla kuva tyypillisestä kuularuuvista. (Saarinen 1994, 15.)

KUVIO 1. Kuularuuvi

Kuularuuvia pyöritetään ISEL 110Nm askelmoottorilla, jonka tarkkuus on 1,8˚. Näin ollen yhden kierroksen askelmäärä on 200 askelta. Valmistaja lupaa askel-

moottorin askelvasteeksi 500 askelta sekunnissa, joten liikenopeudeksi saadaan 80 %:n hyötysuhteella 8 mm sekunnissa. Täyttä 500 askeleen askellustaajuutta en käytä, sillä suurempaa askellustaajuutta käytettäessä heikkenee askelmoottorin vääntömomentti.

3 PIIRILEVYPORAN OHJAINMODUULIN SUUNITTELU

Ohjainmoduulin suunnittelun lähtökohtana oli tehdä ohjainlaite, joka mahtuu yhdelle kaksipuoliselle piirilevylle ja on mahdollista sijoittaa kompaktin kokoiseen laitekoteloon. Itse kytkentäkaavion suunnittelu alkoi komponenttivalintojen tekemisellä, joista tärkein on mikrokontrolleri. Mikrokontrolleri ohjaa ja valvoo kaikkia piirilevyporan toimintoja sekä kommunikoi sarjaportin välityksellä ohjainmoduulin käyttöliittymän kanssa.

3.1 Mikrokontrollerin valinta

Mikrokontrollerin valintaa tehdessä piti huomioida seuraavia asioita:

- riittävä suoritinteho
- riittävä määrä I/O-pinnejä
- muistikapasiteetti
- ohjelmoitavuus
- ulkoisten keskeytysten mahdollisuus
- kontrollerin väylämahdollisuudet.

Mikrokontrollerivalmistajia on useita. Näistä tunnetuimpia ovat Microchip, Motorola ja Atmel. Kullakin valmistajalla on 8-, 16- ja 32-bittisiä kontrollereita, ja kellotaajuudet pääsääntöisesti 1 – 40 Mhz.

Kontrolleriksi valitsin Atmelin ATmega162-tyyppisen kontrollerin, koska se täytti kaikilta osin vaadittavat ominaisuudet ja jättää mahdollisuuden tehdä laajennuksia.

3.1.1 Mikrokontrollerin tärkeimmät ominaisuudet

ATmega162:sta on Atmelin 8-bittinen mikrokontrolleri, jolla pystytään tekemään jopa 32-bittisiä laskutoimituksia. Tärkeimpiä kontrollerin ominaisuuksia on ohjelmamuisti, joka on Flash-tyyppiä ja mahdollistaa kontrollerin uudelleen ohjelmoinnin. Lisäksi ohjelmointi tapahtuu SPI-väylän välityksellä ja voidaan suorittaa ilman piirin irrotusta piirilevystä. Kontrollerissa on kaikkiaan 5 porttia (A, B, C, D, ja E), joista 4 (A, B, C, ja D) ovat 8-bittisiä ja portti (E) on 3-bittinen. Alle on listattu tärkeimpiä mikrokontrollerin ominaisuuksia:

- 16 kt:n ohjelmamuisti (FLASH). 10 000 kirjoitus-/tyhjennyskertaa
- 1 kt:n datamuisti (RAM)
- 512 tavun EEPROM, 100 000 kirjoitus-/tyhjennyskertaa
- ohjelmointi ilman piirin irrotusta piirilevystä
- 2 kpl 8- ja 16-bittisiä laskureita
- 16 I/O-pinniä, joihin mahdollista ohjelmoida keskeytys
- 2 kpl sarjaliikenneportteja (USART)
- 0 – 16 Mhz:n kellotaajuus.

3.2 Kommunikointi isäntälaitteen kanssa

Pc:n Windows-alustalla toimivan käyttöliittymän ja mikrokontrollerimoduulin välinen liikennöinti päätettiin toteuttaa sarjaväylällä (RS232), koska tiedonsiirto on vähäistä ja lähes jokaisessa pc-tietokoneessa on sarjaportti (COM-portti). Sarjaportti on standardoitu jo vuonna 1962, ja julkiseksi se tuli vuonna 1969. väylä on suunniteltu kahden tietokonelaitteen väliseen liikennöintiin. Sarjaliitännän suurimpiin etuihin kuuluu sen yksinkertaisuus. Ilman kättelylinjoja dataa voidaan siirtää kahden johtimen avulla. Nykyään RS232-sarjaväylän suurimpana haittana on sen alhainen datansiirtonopeus. (Engdahl 1993.)

3.3 Askelmoottoreiden ohjaus

Askelmoottorin ohjauksessa päädyin käyttämään ST Microelectronics L298N -piiriä, johon on integroitu kaksi H-siltapiiriä. Kyseinen komponentti on suunniteltu ohjaamaan induktiivisia kuormia, kuten releitä, solenoideja, tasavirta, ja askelmoottoreita. Tämän piirin ohjaamiseen on suunniteltu erillinen SGS-Thomsonin L297-piiri, joka muodostaa kellopulssista ohjaussignaalit L298N-piirille siten, ettei kontrollerilla tarvitse muodostaa mitään erityistä bittikuviota askelmoottorin ohjaukseen. Näin ollen vapautuu kontrollerin I/O-nastoja toiseen käyttöön. Lisäksi L297-piirissä on mahdollinen virran rajoitustoiminto, joka on tarpeellinen matalaimpedanssisilla kuormilla, koska sillä voidaan rajoittaa kuormaan johdettavaa virtaa.

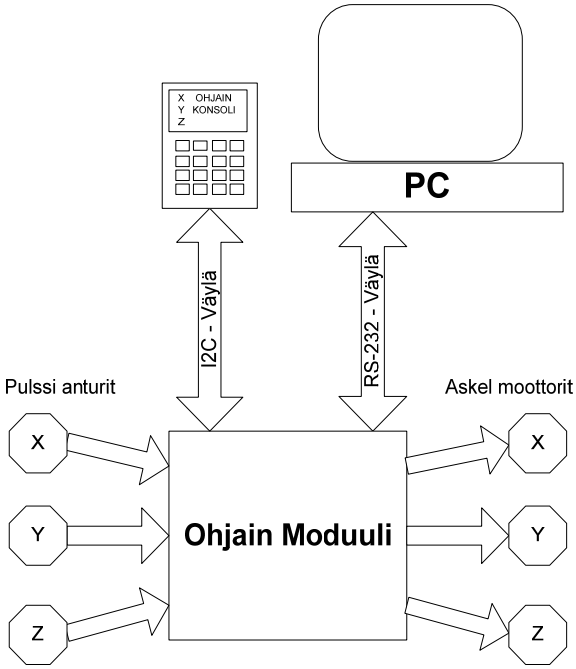
3.4 Kuularuuvien tilan seuranta

Jotta mahdolliset iskut ja tönäisyt eivät saattaisi piirilevyporan X, Y-tasoa epäkohdistukseen, on kuularuuvin pyörittävä täsmälleen oikea määrä kierroksia/askeleita. Täten tarvitaan kuularuuvin liikkeen seuranta. Tehtävään on poramoduulin suunnittelija Arvo Saarinen valinnut optisen pulssianturin, jonka erottelukyky on tarvittavat 200 pulssia/kierros. Näin ollen voidaan kuularuuvin asentoa seurata reaaliajassa ja korjata mahdolliset kohdistukseen syntyvät poikkeamat.

4 PIIRILEVYPORA JÄRJESTELMÄN RAKENNE

Järjestelmää ohjaa Pc-tietokone, joka syöttää sarjaportin välityksellä mikrokontrollerimoduulille askelmääriä. Saamansa datan perusteella mikrokontrollerimoduuli ohjaa kuularuuveja pyörittäviä askelmoottoreita ja kerää pulssiantureilta takaisinkytkentätietoa kuularuuvien liikkeistä. Lisäksi tulevaisuuden sovelluksia varten on mikrokontrollerimoduuliin suunniteltu liitäntä erilliselle ohjainkonsolille. Konsolin näytölle voidaan tulostaa mm. x-, y- ja z-akseleiden tila, ja konsolin näppäimistö on mikrokontrollerimoduulin ohjausta varten. Kommunikointi oh-

jainkonsolin ja mikrokontrollerimoduulin välillä tapahtuu I2C-väylällä. Lisäksi näppäimistön painallus ohjainkonsolissa aiheuttaa tilamuutoksen yhdessä ohjainkonsolin ja mikrokontrollerimoduulin välisessä liitäntäjohtimessa, joka välitetään mikrokontrollerin ulkoisen keskeytyksen I/O-nastaan.



KUVIO 2. Järjestelmän lohkokaavio

4.1 Ohjainkonsolin rakenne

Ohjainkonsoli sisältää taustavalaistun 4 x 20 merkkiä sisältävän näytön, jolla voidaan näyttää erilaisia prosessitietoja mikrokontrollerimoduulista. Lisäksi ohjainkonsolissa on 4 x 4 näppäimistö, jolla voidaan ohjata mikrokontrollerimoduulia. Nämä laitteet on yhdistetty mikrokontrolleriin kahdella I/O-laajennuspiirillä (PCF8574). Piirit ovat I2C-piirejä, ja niillä voidaan siirtää 8-bittistä rinnakkaismuotoista dataa kahteen suuntaan. Syötettäessä dataa ohjainkonsolista eli näppäimistöä painettaessa välittyy mikrokontrollerille erillistä johdinta pitkin keskeytystieto. Täten mikrokontrollerilla osataan lukea ohjauskonsolista syötettyä dataa vain silloin, kun sitä syötetään eikä suoritinaikaa kulu näppäimistön pollaamiseen.

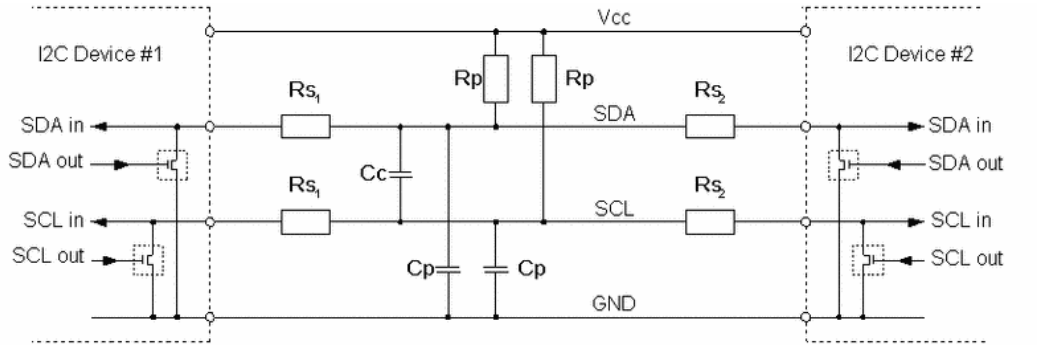
4.1.1 I2C-Liityntä

I2C-väylä on Philipsin kehittämä kaksisuuntainen, synkroninen väylä, jolla haluttiin tehdä helpoksi kommunikointi samalla piirilevyllä sijaitsevien komponenttien kesken. Alun perin siirtonopeudeksi määriteltiin enimmillään 100 Kbit/sec, ja monet sovellukset eivät vaadikaan nopeampaa tiedonsiirtoa. Niille sovelluksille, jotka tarvitsevat nopeampaa siirtokaistaa, on mahdollista käyttää 400 Kbit/sec siirtonopeutta, ja vuonna 1998 julkaistiin 3,4 Mbit:n siirtonopeudella toimiva I2C-väylä. Väylä toimii avoin kollektori -periaatteella, jolloin linjalla on oltava ylösvetovastukset. Poikkeuksena on kuitenkin nopea I2C-väylä, jossa laitteet ajavat myös ykkösbitin johtimeen. (Telos EDV Systementwicklung GmbH.)

Vaikka I2C.-väylä on suunniteltu yhdistämään samalla piirikortilla sijaitsevia komponentteja, ei sitä kuitenkaan käytetä yhdistämään ainoastaan samalla piirikortilla sijaitsevia komponentteja, vaan myös kaapelin välityksellä yhdistettyjä komponentteja. Väyläprotokollan yksinkertaisuus ja soveltuvuus moniin käyttökohteisiin, kuten kulutuselektroniikkaan ja ajoneuvoihin, tekee siitä erittäin houkuttelevan erilaisiin väyläsovelluksiin.

Lista I2C-Väylän parhaista ominaisuuksista.

- Tarvitaan vain kaksi johdinta
- Ei tarkkaan määriteltyä tiedonsiirron kelloa, vaan Master-laite muodostaa kellosignaalin
- Kaikkien väylään liitettyjen komponenttien välillä on yksinkertainen isäntä/orja suhde
- I2C-väylä on oikea moni-isäntä väylä, joka tarjoaa datan törmäyssuojan
- Samaan väylään on mahdollista liittää kahdeksan laitetta.



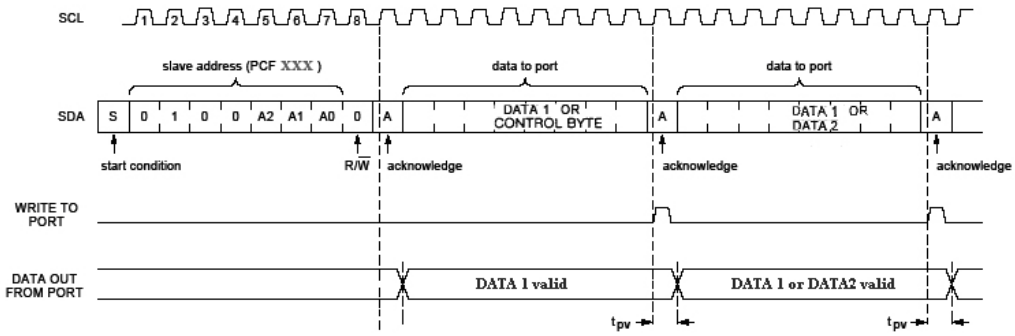
KUVIO 3. I2C-väylän rakenne

TAULUKKO 1. I2C-Väylän rakennekuvan selitteet

Vcc	I2C-väylän käyttöjännite. Tyypillisesti 1,2-5,5V.
GND	Maataso.
SDA	Sarjadata (I2C datalinja).
SCL	Sarja kello (I2C datan kellotus linja).
Rp	Ylösvetovastus.
Rs	Sarjaresistanssi.
Cp	Kapasitanssi maata vastaan.
Cc	Linjojen välinen kapasitanssi.

4.1.2 I2C-väylän sarjaliikenneprotokolla

Kuvio 3 esittää kahden väylään kytketyn laitteen välistä linjaa ja taulukko1:ssä on selitetty kuvio 3:ssa kuvatut suureet. Väylän ollessa vapaa, kummatkin johtimet ovat tilassa 1. Liikenne väylällä alkaa aloitus bitillä, jonka master-laite muodostaa vetämällä SDA-johtimen tilaan 0, SCL-johtimen ollessa tilassa 1. Seuraavaksi master-laite kellottaa 8 databittiä, joista osa on laitekohtaisia. Pääsääntöisesti ensimmäinen tavu sisältää kuitenkin kohdelaitteen osoitteen ja tiedon siitä onko kyseessä kirjoitus vai lukutoiminto. Data kellotetaan SDA-johtimesta laitteelle SCL:n nousevalla reunalla. Tämän jälkeen Master-laite lähettää yhden ylimääräisen (yhdeksännen) kellopulssin, jonka orjalaite kuittaa vetämällä SDA-johtimen tilaan 0, SCL-johtimen ollessa tilassa 1. Tätä kutsutaan acknowledge-bitiksi eli kuittausbitiksi. Kuittausbitin muodostaa kulloinkin dataa vastaanottava laite, jokaisen vastaanotetun tavun jälkeen. Kuviossa 4 on esitetty I2C-väylän toiminta.

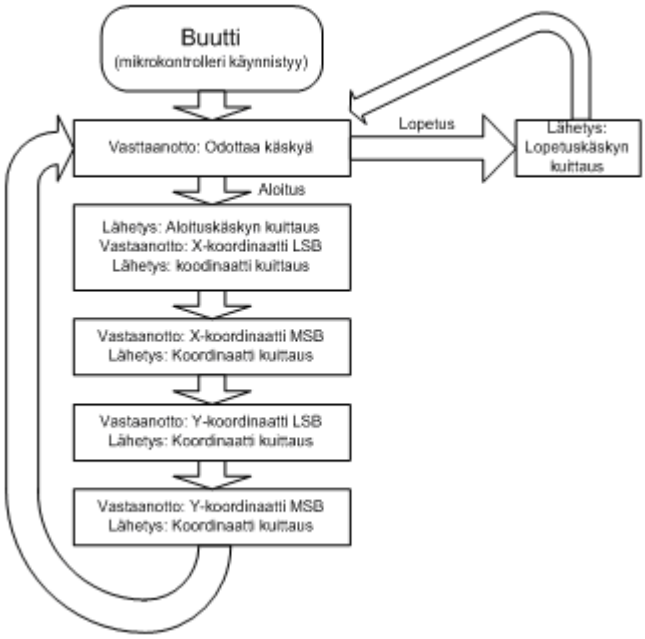


KUVIO 4. Datan siirto I2C-väylällä.

Alussa I2C-väylän standardi toimintajännite oli 5 volttia. Toisen sukupolven I2C-väylä julkistettiin vuonna 1998, jolloin I2C:n vertailujännite laskettiin 2 volttiin. (Telos EDV Systementwicklung GmbH.)

4.2 Kommunikointi pc:n ja mikrokontrollerin välillä

Sarjaliikenne tietokoneen ja kontrollerikortin välillä tapahtuu RS232-väylällä, jossa TXD-pinni on lähetyslinja ja RXD-pinni on datan vastaanottoon. Näiden lisäksi ei tässä väylä-sovelluksessa tarvitse kytkeä muuta, kuin maataso. Datan siirto sarjaväylällä on asynkronista, eli dataa ei kelloteta kohdelaitteeseen. RS232-väylällä käsitellään tietynmittaisia bittilohkoja (datapaketteja). Koska paketin kooksi on määritelty 8 bittiä ja porauskoordinaatti on 16-bittinen, pitää se lähettää kahdessa osassa. Kuviossa 5 on selvennetty sarjaliikenteen vuokaavio.



KUVIO 5. Sarjaliikenteen vuokaavio

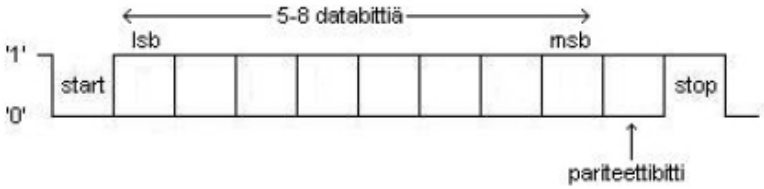
Sarjaväylän kättelyihin käytettävät merkit ovat seuraavat:

- aloituskäskyn kuittaus = 0 x 61 = ’a’
- koordinaatti kuittaus = 0 x 6f = ’o’
- lopetuskäskyn kuittaus = 0 x 6c = ’l’.

4.2.1 RS232datasiirto

Signaalitaso väylän johtimessa on koko ajan tilassa 1, jos dataa ei lähetetä. Datasiirto aloitetaan synkronisoimalla eli lähettämällä aloitus- eli start-bitti, jolla signaalitaso vedetään alas (KUVIO 6). Sitä seuraa 5 - 8 bitin datapaketti (koko määriteltävissä), LSB-bitti ensimmäisenä ja MSB-bitti viimeisenä. MSB-bitin jälkeen voidaan lähettää pariteettibitti, jos halutaan käyttää virheen havaitsemista. Varsinaista virheen korjausta sillä ei kyetä tekemään, joten lisäluotettavuutta tarvittaessa joudutaan käyttämään ohjelmallista virheentarkistusta. Pariteettibitti ei ole pakollinen, mutta sitä on suositeltavaa käyttää suuremmilla siirtonopeuksilla. Parillinen pariteetti asetetaan sen mukaan, onko datapaketissa olevien ykkösbittien määrä parillinen. Datapaketti lopetetaan lopetus- eli stop-bittiin. Lopetusbitti on pakollinen, ja sen pituus on määriteltävissä 1, 1,5 tai 2 bitin mittaiseksi. Lopetusbitin pituudella määrätään kahden datapaketin välinen etäisyys eli se aika, mikä

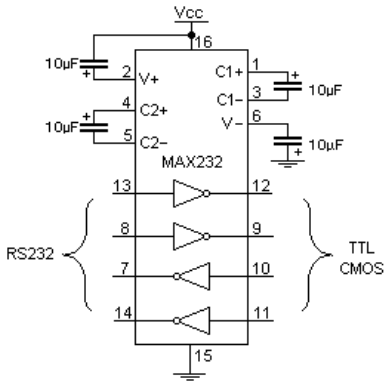
vastaanottajalla on datan prosessointiin ennen uuden datapaketin lähetyksen alkua. Datan siirtonopeudeksi valittiin riittävä 9600 baudin nopeus. (Engdahl 1993.)



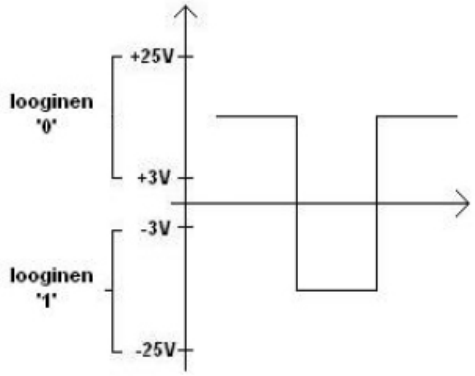
KUVIO 6. RS232datapaketti

4.2.2 RS232liitynnän sovitus TTL-tasoon

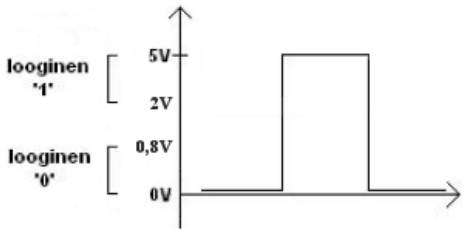
Koska RS232sarjaliikenteen loogiset jännitetasot poikkeavat TTL-logiikan loogisista jännitetasoista, on välillä tehtävä jännitesovitus. Tähän käytin MAX232 piiriä, joka muodostaa -10 V:n, ja +10 V:n jännitteet yhdestä +5 V:n käyttöjännitteestä. Piiri sisältää kaksi lähetin- ja 2 vastaanotinpiiriä samassa piirissä.



KUVIO 7. Tyypillinen MAX232-piiri



KUVIO 8. RS232loogiset jännitetasot

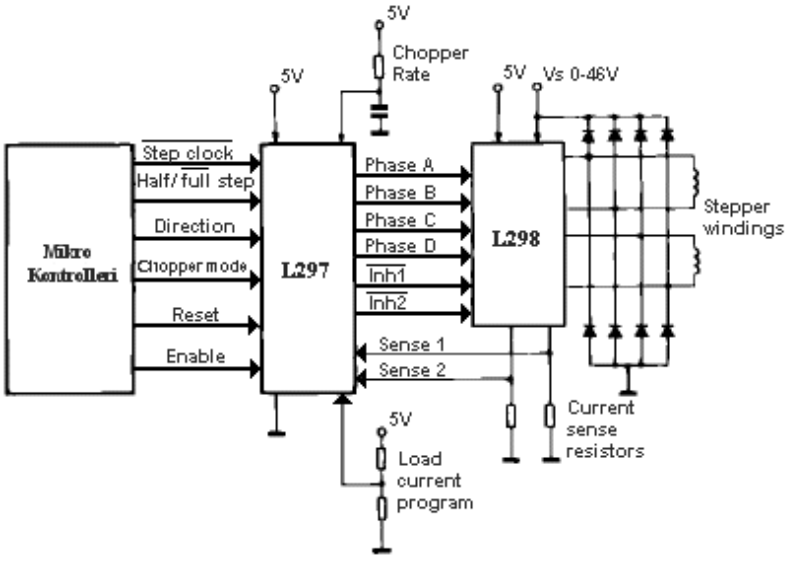


KUVIO 9. TTLloogiset jännitetasot

Kuviot 8 ja 9 osoittavat, millaisen jännitesovituksen MAX232-piiri tekee.

4.3 Askelmoottoriohjaus

Mikrokontrollerilla ohjataan epäsuorasti kolmea askelmoottoria, sillä itse mikrokontrolleri on kytketty kolmeen L297-piiriin. L297-piiri on askelmoottorikontrolleri, joka muodostaa tarvittavat bittikuviot L298N-piirille. L298N-piiri nostaa ohjaus-signaalitason ja vahvistaa ohjausvirran askelmoottoreille vietävälle tasolle. Järjestelmän mahdolliset käyttöjännitteet on ilmoitettu kuviossa 10. Kyseisessä sovelluksessa käytämme askelmoottorin ohjainpäätteen (L298N) askelmoottori käyttöjännitteenä 12 V:n jännitettä.



KUVIO 10. Askelmoottorin ohjauskytkentä

Kuviossa 10 on mikrokontrollerin kytkentä askelmoottorikontrolleriin sekä askelmoottorikontrollerin ja askelmoottoripäätteen välinen kytkentä. Mikrokontrolleri on kytketty kaikkiaan kolmeen askelmoottorikontrolleriin, joista ohjaussignaalit step clock, direction ja enable viedään suoraan kullekin askelmoottorikontrollerille. Reset-, Chopper mode ja half/full step -ohjaukset ovat kaikille askelmoottorikontrolleri-piireille yhteiset. Alla on esitetty askelmoottorikontrollerin I/O-nastojen toiminnat.

- Reset: Ajamalla tähän pinniin looginen tila 0. Palautuvat askelmoottorikontrollerin lähdöt ”koti positioon” eli (ABCD = 0101).

- Half / full step: Asettamalla looginen 0 tähän pinniin askelmoottori toimii normaalisti. Asettamalla looginen 1 saadaan askelmoottori ottamaan puoliaskelia. Tätä ominaisuutta emme kuitenkaan käytä, sillä ollessaan puoliaskeltilassa moottoria jouduttaisiin pitämään koko ajan ”jännityksessä” ja ohjauksen katketessa ei voida olla varmoja kohdistuksesta. Lisäksi pulssianturien tarkkuus ei riitä kyseisen ominaisuuden käyttämiseen.

- Direction: Tähän pinniin vietävä looginen tila määrittää askelmoottorin pyörintäsuunnan.

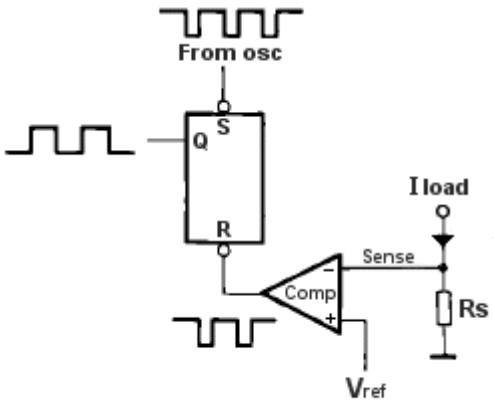
- Chopper mode: Tähän pinniin vietävä looginen tila määrää ”chopper” moodin. Asettamalla tilaksi 0 chopper kytkeytyy L298N-piirin INH1- ja INH2-nastoihin, ja asettamalla tilaksi 1 kytkeytyy se L298N-piirin vaihelinjoihin eli A-, B-, C- ja D-sisääntulopinneihin.

- Enable: Tällä pinnillä aktivoidaan askelmoottorikontrollerin ulostulot eli ohjaukset INH1, INH2, A, B, C, ja D. Aktivointi tapahtuu asettamalla tilaksi looginen 1.

- Step clock: Viemällä kellopussia tähän pinniin askelmoottorikontrolleri generoi pulssikuviota askelmoottoripäätteille siten, että jokaisella kellopulssin nousevalla reunalla askelmoottori ottaa askeleen.

4.4 Askelmoottorikontrollerin toiminta

L297 on askelmoottorikontrolleri, joka on suunniteltu toimimaan yhdessä L293E:n ja L298N:n (H-siltapiirien) kanssa. L297-piiri vastaanottaa ohjaussignaalit mikrokontrollerilta ja tarjoaa kaikki tarvittavat ohjaussignaalit L298N-piirille. L297 askelmoottorikontrolleri sisältää myös ”PWM-chopper” ominaisuuden, eli askelmoottorille vietävä askelluspulssi jaetaan useampaan osaan. Näin estetään virtaa nousemasta yli sallitun raja-arvon askelmoottorin käämissä. Virran suuruutta pitää rajoittaa myös päätteenä toimivan L298N-piirin suurimman sallitun nimellisvirran takia, joka on 2 A.



KUVIO 11. Chopper-kytkentä

Kuvion 11 RS-kiikku virittyy jokaisella oskillaattorin pulssilla aktivoiden ulostulon, jolloin virta alkaa kasvaa kuorman yli ja jännite Rs-vastuksen yli kas-

vaa.Virran lopulta saavuttaessa Vref-jännitteen, jolloin rs-kiikku nollautuu deaktivoiden ulostulon seuraavaan oskillaattoripulssiin saakka.

Oskillaattoripulssi muodostetaan yhdellä L297-piirillä. Loput L297-piirit synkronoidaan samaan kellopulssiin yhdistämällä Sync-nastat ja maadoittamalla muiden askelmoottorikontrolleripiirien Osc-nastat. Oskillaattoripulssi muodostetaan L297-piiriin kytkettävällä RC-parilla, jonka värähtelytaajuus on karkeasti (1/0,7) * RC-vakio (vastusarvon pitää olla vähintään 10 kΩ). Askelmoottorin käämissä kulkeva virta on laskettavissa seuraavilla laskukaavoilla:

$I = E / R * (1 - e^{-t/\tau})$

$C = 1 / 0,7 * (1/ (R * f)$

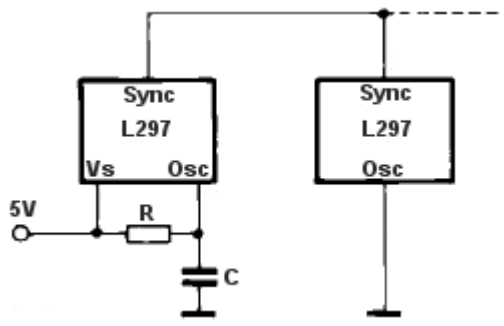
$t_p = (1 / f_{(E)}) * 100\ t_p$

$t_p = 50\ \mu s$

$C = 1 / 0,7 * (1/ (22\ k * 20000) = 3,3\ nF$

$I = E / R * (1 - 2,72^{-(50\mu s/\tau)})$

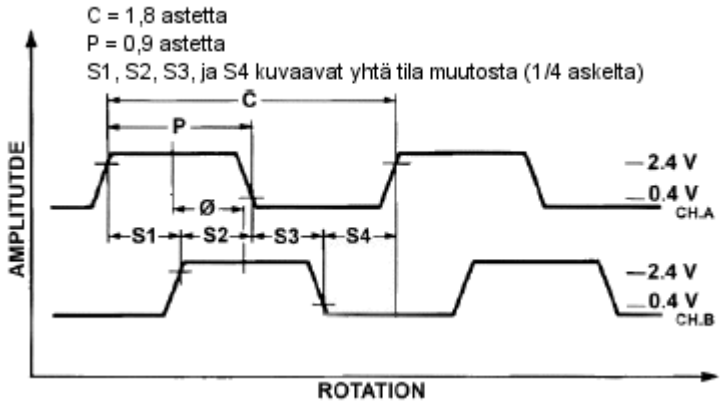
jossa E = käämin jännite (V)
R = käämin resistanssi (Ω)
C = kapasitanssi
e = 2,72
t = jännitteen vaikutusaika (s)
τ = käämin aikavakio (L/T) kuparilla 5 * 10^-14



KUVIO 12. Oskillaattorikytkentä

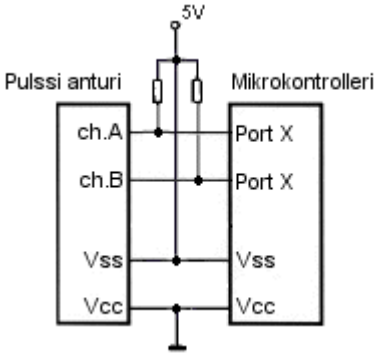
4.5 Kuularuuvin takaisinkytkentätieto

Kuularuuvien liikkeitä seurataan optisilla pulssiantureilla, joiden tarkkuus on sama kuin askelmoottorien askelmäärä/kierros eli 200/kierros. Pulssianturit ovat Fauhaberin valmistamat ja tyypiltään HEDS-5500. Pulssianturissa on kaksi kanavaa, jotka on suoraan kytketty mikrokontrollerin I/O-nastoihin. Kuviossa 13 on kuvattu pulssianturin antamaa kanavien A- ja B-pulssikuviota.



KUVIO 13. Heds-5500 pulssikuvio

Tästä Optisen anturin pulssikuviosta pystytään mikrokontrollerilla ohjelmallisesti päättelemään, kumpaan suuntaan pulssianturiin kytketty kuularuuvi pyörii.



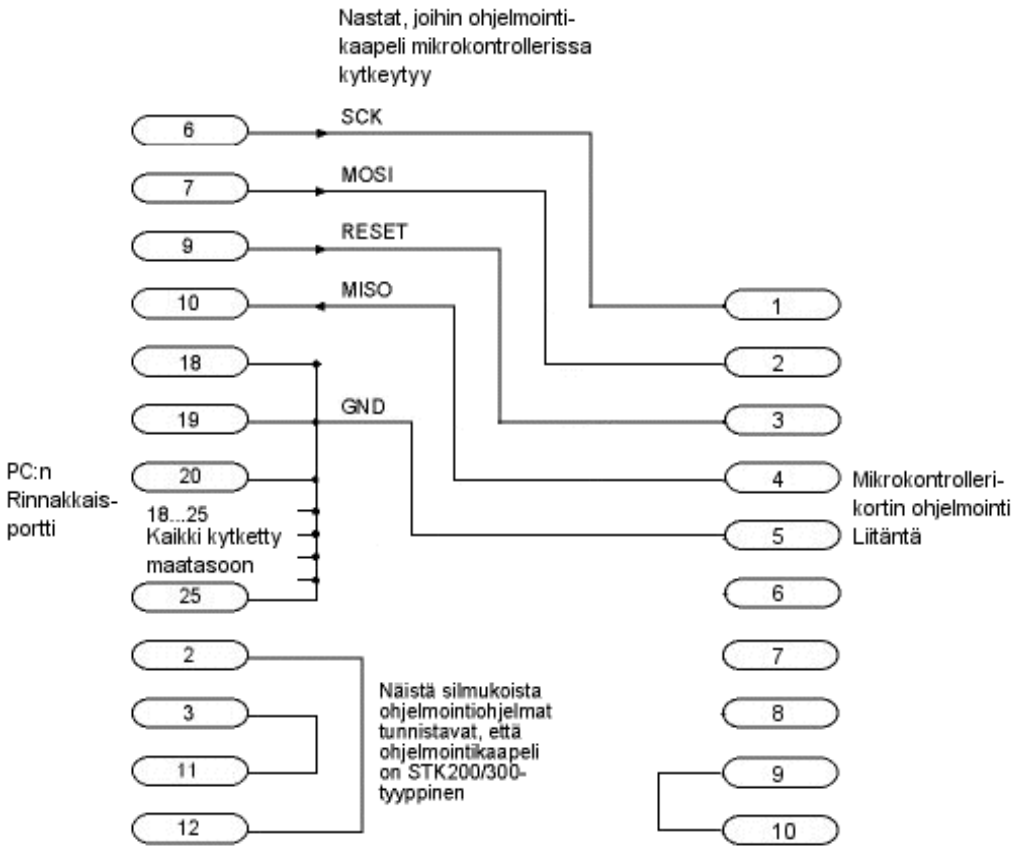
KUVIO 14. Pulssianturi kytkentä

Kuviossa 14. on pulssianturikytkentä. Pulssianturi tarvitsee käyttöjännitteen ja maatason, jotta optinen veräjä voisi toimia. Lisäksi pulssianturin loogisen tilan 1. lähtöjännite on varsin lähellä TTL-tason loogisen tilan 1. alarajaa, joten signaalia

vahvistetaan johtimissa 1 KΩ:n ylösvetovastuksin. Jokainen tilanmuutos pulssianturin johtimissa aiheuttaa mikrokontrollerissa keskeytystoiminnon, jolloin luetaan jokaisen pulssianturin lähdöt.

5 MIKROKONTROLLERIN OHJELMOINTI

Mikrokontrollerin ohjelmointi suoritetaan PC-tietokoneen rinnakkaisporttiin kytkettävällä kaapelilla, jonka toinen pää kytketään mikrokontrollerikortilla sijaitsevaan ohjelmointiliittimeen Pc:n rinnakkaisporttiin. Mikrokontrollerikortin ja rinnakkaisportin logiikan jännitetasot ovat yhtenevät, joten jännitesovitusta ei tarvita. Mikrokontrolleri voidaan ohjelmoida Kanda Systemsin suunnitteleman STK200/300harjoittelukortin ISP-ohjelmointia tukevilla sovelluksilla. (Tietomyrsky Oy.)



KUVIO 15. Mikrokontrollerikortin ohjelmointikaapeli

Mikrokontrolleri on ohjelmoitu pelkästään C-kielellä, lukuun ottamatta valmiita kirjastoja, joista muutamia käytettiin ohjelmassa. Kääntäjä tekee konekielisen käännöksen C-ohjelmakoodista, joka siirretään mikrokontrolleriin. Ohjelmoinnissa käytin CodeVisionAVR-ohjelmaa, jossa on integroitu laitteistoon kytketyn AVR kontrollerin ohjelmointi (In-System AVR Chip Programmer), jolla voidaan siirtää konekielelle käännetty ohjelma suoraan mikrokontrollerille irrottamatta sitä piirilevystä. CodeVisionAVR:ään integroitua ohjelmointitoimintoa käyttäen voidaan mikrokontrolleri ohjelmoida myös jollakin muulla sovelluksella luodulla, valmiiksi konekielelle käännetyllä ohjelmalla.

5.1 Laitteistojen rajapinnat

Sovitettaessa yhteen eri suunnittelijoiden kehittämiä laitteita tai ohjelmia tarvitaan laitteistojen, kuten tässäkin tapauksessa käyttöliittymän ja ohjattavan moduulin välisten kättelyiden toteuttamiseen tarkat määritykset. Tässä tapauksessa piti sopia mm. sarjaliikenneprotokollan asetukset ja laitteistokättelyt siten, ettei ole mahdollista tulkita dataliikenteeksi tarkoitettua datapakettia väärin eli laitteistojen väliseksi kättelykomennoksi.

Lisäksi sovittiin, että käyttöliittymä prosessoi datan siten että ohjainmoduulin tarvitsee tehdä siitä mahdollisimman pieni osuus. Muutoin ohjainmoduuli mahdollisesti joutuisi sovittamaan PowerPCB-ohjelmasta saatavia mittayksikköjä askelmääriksi ja järjestämään koordinaatteja siten, että reiältä toiselle kuljettu matka olisi mahdollisimman pieni. Tämä vaatisi, että koordinaatteja ladattaisiin mikrokontrollerin muistiin, minkä jälkeen ne organisoitaisiin porausjärjestykseen. Tämä on kuitenkin järkevintä jättää käyttöliittymän suoritettavaksi, sillä mikrokontrollerin muistikapasiteetti on varsin rajallinen. Lisäksi ohjainmoduulia on helpompi soveltaa muihin käyttökohteisiin, jos se ei prosessoi dataa, vaan ottaa suoraan vastaa komentoja ja askelmääriä.

5.2 Mikrokontrolleriohjelman alustukset

Mikrokontrollerikoodin kirjoittaminen aloitetaan kirjastomäärityksillä

```
#include <mega162.h>
#include <i2c.h>
#include <delay.h>
#include <stdio.h>
#include <usart.h>
#include <stdlib.h>
```

KUVIO 16. Ohjelmaan sisällytetyt kirjastot

Ohjelmoinnissa käytetyistä kirjastoista suurin osa oli valmiina. Kirjastot ovat kontrolleriin liitettävien laitteiden ja mikrokontrollerin sisäisten laitteiden ohjaamiseen suunniteltuja. Kirjastot ovat laite-/toimintokohtaisia tietueita, jotka sisältävät funktiot mm. sarjaliikenteelle, I2C-väylälle, viiveen määrittelyyn, standardien I/O-toimintojen suorittamiseen ja I/O-rekisterien määrittämiseen. Myös valmiiden kirjastojen muokkaaminen on mahdollista, sillä tein itse lisäyksen sarjaliikennekirjastoon, jotta pystyisin vastaanottamaan 16-bittisen muuttujan suoraan pääohjelmasta.

Jotta mikrokontrolleri toimii halutulla tavalla, pitää sen sisäisten toimintojen, kuten sarjaliikennepuskuri, laskurit/ajastimet, keskeytyspalvelut, I/O-pinnien data suunta, jne. alustaa toimimaan halutulla tavalla. Se tehdään kirjoittamalla haluttu alustusarvo kyseisten laitteiden ohjain-rekistereihin. Ohjainrekisterit ovat osa kontrolleripiirin muistiavaruutta, ja ne on määritetty kontrollerikohtaisessa kirjastossa, jonka nimi viittaa itse kontrolleripiiriin. (LIITE 6.)

Laitekohtaisille ohjainrekistereille on määritelty tietyt osoitteet, ja ne voidaan helposti alustaa haluttuihin arvoihin ohjelmasta käsin. Tehdasasetuksina ne on asetettu siten, ettei haittaa, vaikkei kaikkia ohjainrekistereiden arvoja määritetä. Ohjelmakoodi mikrokontrollerin sisäisten laiteiden alustuksista löytyy liitteestä 7.

5.3 Pääohjelmaluuppi

Pääohjelmakoodi muodostuu silmukasta, josta ei poistuta ”koskaan”, lukuun ottamatta keskeytyspalvelua. Keskeytyksen tullessa suoritetaan sillä hetkellä suorituksessa oleva operandi loppuun, minkä jälkeen tallennetaan rekisterien tila, ja siirrytään suorittamaan keskeytyspalvelufunktiota. Keskeytyspalvelufunktion suorituksen päätyttyä palautetaan rekisterien arvot ja palataan suorittamaan ohjelmakoodia siitä kohdasta eteenpäin, jossa se oli sillä hetkellä, kun siirryttiin itse keskeytyspalveluun.

Koska osa funktioista on sijoitettu laitekohtaisiin kirjastoihin ja neljään erilliseen laitekohtaiseen ohjelmakoodiosioon (päättis.c, stepper_drive.c, interrupt_handle.c, ja console.c), jää varsinainen pääohjelmaluupin osuus pienemmäksi, ja sisältää pääasiassa pelkkiä funktiokutsuja.

```
while (1)
      {


      apu = getchar();       // vastanottaa tavun

switch (apu) {

      case 0x61: {           // hex 0x61 on kirjain a (aloitus käsky)
     delay_ms(10);
      putchar(0x61);         // Kuittaa aloituksen
     delay_ms(10);
      getint();             // c on 16-bittinene apu muuttuja, joka viedään
     delay_us(500);
      z_tx_input = c;        // z-akselin ajo koordinaatiksi. lopullisessa versiossa tämä koordinaatti viedään x-
akselin ajo koordinaatiksi.
     c = 0;
      stepper_drive_z();     // ajetaan moottori kyseiseen koordinaattiin
      getint();             // vastaan otetaan toinen tavupari protokortissa vain kättelybn takia.
     delay_us(500);
      y_tx_input = c;        // lopullisessa versiossa tämä koordinaatti viedään y-akselin ajo koordinaatiksi.
     c = 0;
      stepper_drive_y();
      apu = 0;
}
break;

      case 0x6c: {           // Mikäli Tietokone antaa tämän tavun. tietää kontrolleri lopettaa ajon.

      putchar(0x6c);         // Kuitataan vastaanotettu käsky.
      z_tx_input = 0;        // asetetaan nolla pisteen koordinaatti z-akselille.
```

```
      stepper_drive_z();      // ja ajetaan siihen
      y_tx_input = 0;         // lopullisessa versiossa tämä koordinaatti viedään y-akselin ajo koordinaatiksi.
      stepper_drive_y();
      apu = 0;
}
break;
      }
```

KUVIO 17. Pääohjelmaluuppi

5.4 Laitekohtaiset lähdekoodit

Pääohjelmakoodista tehdään funktiokutsuja laitekohtaisiin ohjelmakoodiosioihin, joita ovat kirjastojen lisäksi erilliset ohjelmoijan määrittämät lähdekoodit. Niihin on kyseisen sovelluksen ohjelmakoodissa määritelty mm. askelmoottoreiden ajo-funktiot, ja tulevaisuudessa tullaan määrittämään myös ohjainkonsolin toiminnot. Kuviossa 18 on esimerkkinä yhden askelmoottorin ajofunktio.

```
 void stepper_drive_x (void)

 {
    x_tx_input = (x_tx_input + 32768);
       while (x != x_tx_input)
         {
     if (x > x_tx_input) {                    // Määrätään x-poran suunta
           suunta = 0;
     }
     else {
           suunta = 1;
     }
     if (suunta == 0) {
           PORTB = 0x83;                      // 03=täysi askel 83=puoliaskel, poista reset, Suunta taakse, aseta L1
     }
     else {
           PORTB = 0xf3;                      // 73=täysi askel f3=puoliaskel, poista reset, Suunta eteen, aseta L1
     }
           PORTC = 0x02;                      // Pulssi ylös
           delay_ms(stepper_delay);
           PORTC = 0x40;                      // Pulssi alas
           delay_ms(stepper_delay);

      //      if (x_tx_input > x)
       //       {  x++; }                     // vähentää x:n arvoa. korvataan optojen antamalla tiedolla
        //                                    // kunhan saadaan ne kytkettyä.
         //    else { x--; }

           if (x == x_tx_input) {
           PORTB = PORTB &= 0x01;  // poista L1
            }
         }
         x_tx_input = (x_tx_input -32768);
```

```
}
```

KUVIO 18. X-askelmoottorin ajo-funktio

Optisten antureiden lukufunktiot ovat keskeytysvektorifunktioita. Keskeytysvektorifunktioihin ei käyttäjä tee kutsuja, vaan mikrokontrolleri siirtyy automaattisesti suorittamaan ohjelmakoodia keskeytysvektorifunktion osoitteesta, tilamuutoksen ilmettyä pinnissä, joka on määritetty keskeytyslähteeksi. Keskeytysvektorin osoitteessa on hyppykäsky varsinaisen keskeytysfunktion alkuun, jonka sisällön voi käyttäjä itse määrittää. (LIITE 10.)

5.5 Virheiden etsintä lähdekoodista

Ohjelmakoodisegmentin valmistuttua on seuraava vaihe testata sen toimintaa. Mikäli ohjelmakoodi ei virheiden etsimisestä huolimatta toimi halutulla tavalla, on ohjelmaa hyvä pystyä simuloimaan virtuaalisessa ympäristössä. Tähän käytin Atmelin AVR Studio 4-ohjelmaa, jossa on mahdollista käyttää virtuaalista mikrokontrolleria ja ajaa siinä valmiiksi käännettyä ohjelmakoodia virtuaalisesti. Ohjelmakoodin suoritusta voi seurata rivi kerrallaan ja syöttää dataa virtuaalisen kontrollerin I/O-pinneihin, jolloin tilanne vastaa lähes realistista tilannetta. Tätä toimintoa varten CodeVision-kääntäjä luo erityisen COF-tiedoston, jonka AVR Studio 4 osaa tulkita.

6 LAITTEISTON TESTAUS

Laitteiston testaus on oleellinen osa sen kehitystä, ja yleensä eri osakokonaisuudet testataan jo suunnitteluvaiheessa. Sarjaliikenne, pulssianturien luku, askelmoottori ohjaus ja ohjainkonsoliin kirjoittaminen testattiin tässäkin projektissa yksittäin, mutta tuolloin ei vielä ollut käytettävissä kaikilta osin samoja komponentteja, kuin lopullisessa laitteistokokonaisuudessa. Lisäksi mm. moottorien kuormitus lopulliseen käyttökohteeseen verrattuna on vaikeasti simuloitavissa. Lopullisessa käyt-

tökohteessa kaikkien toimintojen yhtaikaisen toiminnan toteuttaminen saattaa myös vaatia muutoksia ohjelmakoodiin.

Laitteiston testaus tehtiin minimiperiaatteella, jolloin siihen käytettiin mahdollisimman vähän resursseja. Näin tehtiin, koska käytettävissä ollut aika oli vähäinen ja kaikki laitteistoon tarvittavia komponentteja ei olut käytettävissä. Testauksen määrä ja laatu todettiin kuitenkin riittäviksi, sillä testaukset tehtiin ensimmäisellä proto-versiolla, ja laitteistoon on mahdollisesti tulossa vielä muutoksia.

6.1 Sarjaliikenteen toiminnan testaus

Sarjaliikenteen testaukseen mikrokontrollerin ja pc-tietokoneen välillä tehtiin erityinen mikrokontrollerisovellus. Sovelluksen tehtävänä oli ottaa yksi tavu vastaan sarjaportin välityksellä ja lähettää data muuttumattomana takaisin tietokoneelle. Näin testattiin yhtaikaisesti sarjaliikenteen toimivuus kumpaankin suuntaan. Sarjaliikenteen lähettämiseen tietokoneelta käytettiin konsoli ohjelmaa (RealTerm Ver:1.99.0.34), jolla voidaan samanaikaisesti lähettää ja vastaanottaa sarjadataa tietokoneen sarjaportista.

KUVIO 16. Sarjaliikenteen testikytkentä

6.2 Askelmoottoriohjauksen testaus

Askelmoottoriohjausta testattaessa syötettiin mikrokontrollerille koordinaatteja (askelmääriä), jolloin mikrokontrolleri generoi ohjauskellosignaalit ja asetti muut Askelmoottorikontrollerin signaali tulot määriteltyihin tiloihin (KUVIO 10). Jär-

kevä tapa syöttää dataa mikrokontrollerille oli sarjaliikenneportti, koska se oli jo aiemmin todettu toimivaksi. Näin ollen annettiin mikrokontrollerille määrätty syöte, jonka perusteella mikrokontrolleri ohjasi askelmoottorikontrolleria. Askelmoottorikontrollerille tehdyn syötteen mukaan askelmoottori askelsi tietyn määrän askelia ennalta määrättyyn suuntaan. Mikrokontrollerille tehtyjen syötteiden, ja askelmoottorin ottamien askelmäärien perusteella voitiin olettaa askelmoottori ohjauksen toimivan moitteetta.

6.3 Optisen pulssianturin toiminnan testaus

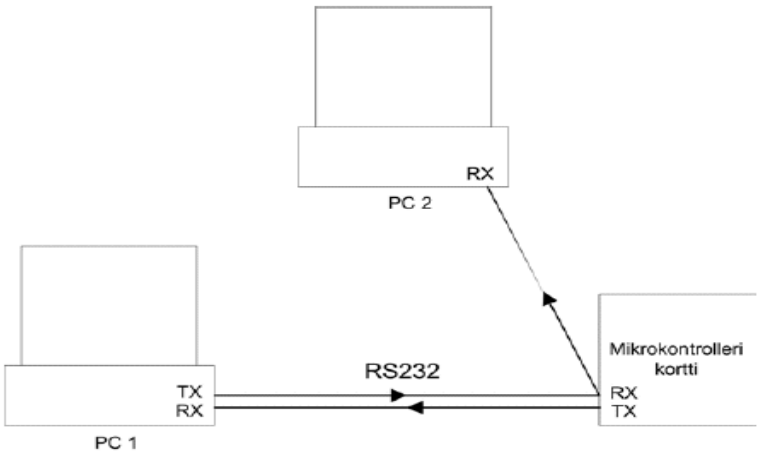
Pulssianturin toiminnan testaamiseksi tehtiin mikrokontrolleriin erillinen sovellus, joka lähetti pulssianturin tilatietoa sarjaportin välityksellä ohjainkonsoli ohjelmaan. Tilatietona toimi 16-bittinen muuttuja, jonka arvo kasvoi, kun anturia pyöritettiin, ja väheni, kun anturia pyöritettiin vastakkaiseen suuntaan. Pulssianturin lukutoiminta näytti saadun datan perusteella toimivan, mutta hieman virheellisesti. Sillä saadussa tilatiedossa oli pieniä poikkeumia. Tämä saattaa tosin johtua käytettävissä olleen pulssianturin optisista poikkeavuuksista, sillä anturin lukukiekko oli kärsinyt mekaanisista rasituksista, eikä täten ollut suora. Koska ohjelmakoodi oli tarkastettu jo useaan kertaan ja pulssianturi viallinen, päätin, että tarkempia testauksia optisille antureille tehdään, kunhan saadaan uusi komponentti.

6.4 Laitekokonaisuuden testaus

Laitekokonaisuuden testaus jouduttiin suorittamaan ilman optisen pulssianturin takaisinkytkentätietoa, sillä uutta komponenttia ei opinnäytetyön valmistumisajankohtaan mennessä hankittu. Tätä varten ohjelmakoodiin piti tehdä pieni väliaikainen muutos, jonka avulla testauksen pystyi suorittamaan ilman pulssiantureita. Kyseinen koodin kohta on kommentoitu askelmoottorien ajofunktioissa.

Laitteistokokonaisuuden testaus tapahtui siten, että RealTerm-ohjainkonsoli-ohjelman sijasta käytettiin kyseisen laitteiston käyttöön suunniteltua käyttöliittymä-ohjelmaa, jonka Mikko Ranua on suunnitellut opinnäytetyönään. Lisäksi ohjainmoduuli oli kytketty ohjaamaan lopullista käyttökohdettaan piirilevyporaa, ja testiohjelmien sijaan käytettiin mikrokontrollerissa lopullista piirilevypora sovellusta.

Sarjaliikenne mikrokontrollerin ja käyttöliittymän välillä toimi moitteetta, joten voitiin todeta käyttöliittymän toimivan ennalta sovittujen määritysten mukaisesti. Käyttöliittymän lähettämän sarjaliikenteen testaus tapahtui kuvanmukaisella kytkennällä.



KUVIO 17. Käyttöliittymän sarjaliikenteen testauskytkentä

Laitteisto (PC1, ja Mikrokontrollerikortti) on kytkettynä yhteen nollamodeemi kaapelilla. PC2-tietokoneeseen asennetulla RealTerm-terminaali ohjelmalla seurattiin käyttöliittymän ja mikrokontrollerin lähettämä dataa.

6.5 Tulokset ja johtopäätökset

Kuvion 17. mukaisella testikytkennällä tehtyjen testien (datan kaappauksen, ja suoritettujen testiajojen) perusteella käyttöliittymä ohjelman, ja mikrokontrollerimoduulin välinen sarjaliikenne toimi virheettä. Käyttöliittymä ohjelmaan on kui-

tenkin jäänyt jostakin syystä väärä askelsuhde mittayksikköä (millimetri) kohden. Tämän takia on piirilevyn ohjainmoduuliin lisätty ohjelmallisesti toteutettu askelsuhteen korjaus. Kyseinen korjaus on kommentoitu liitteen 9 ohjelmakoodissa.

Mikrokontrollerimoduulin askelmoottorien ohjaus toimii piirilevyporalla suoritettujen testiajojen perusteella virheettä. Optisten pulssianturien testaus jäi tässä tapauksessa vajaaksi, koska uusia komponentteja ei opinnäytetyön valmistumisajankohtaan mennessä hankittu. Tästä huolimatta voitiin kuitenkin todeta, kuten luvussa 6.3 Optisen pulssianturin toiminnan testaus todettiin, että pulssianturien luku toimii, mutta luvussa tapahtuu virheitä. Tämän todettiin saattavan johtua itse pulssianturista.

Ohjainkonsolin toiminta todennettiin kirjoittamalla sillä sijaitsevalle näytölle tekstiä, mutta rajallisen aikataulun puitteissa ei ehditty tekemään mitään järkevää sovellusta ohjainkonsolin hyödyntämiseksi laitteistossa. Tulevaisuudessa ohjainkonsolin soveltaminen järjestelmään ei ole välttämätöntä, sillä käyttöliittymäohjelmaan on mahdollista sijoittaa kaikki samat toiminnot kuin erilliseen ohjainkonsoliinkin.

Kaiken kaikkiaan laitteisto toimii suunnitellun mukaisesti. Kaikkia laitteistoon liitettäviä ominaisuuksia ei vielä ollut mahdollista hyödyntää, mutta testausten perusteella laitteisto on todettu toimivaksi.

7 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa piirilevyporakoneen ohjainmoduuli. Tärkeimpinä vaatimuksina laitteelle oli pystyä ohjaamaan askelmoottoreita Windows-ympäristössä toimivan käyttöliittymän kautta.”sarjaliikenneportin välityksellä lähettämän ohjausdatan perusteella”. Käyttöliittymä muokkaa PowerPCB-ohjelmalla luodun poraustiedoston mikrokontrollerille sopivaksi ja lähettää sen sarjaväylää käyttäen mikrokontrollerille.

Opinnäytetyön tuloksena saatiin ohjainmoduuli piirilevynporausjärjestelmään. Ohjausmoduuli toimii itsenäisesti sarjaliikenneportilta saamansa datan perusteella ja täyttää annetut vaatimukset. Ohjainmoduulin resurssit mahdollistavat myös laajennukset tulevaisuudessa. Ohjainmoduulilla voidaan ohjata kolmea askelmoottoria, jotka liikuttavat eri tasoja (x, y, ja z) piirilevyporassa. Ohjainmoduuli on varustettu takaisinkytkennällä, joka mahdollistaa mm. tasoihin kohdistuneiden töytäisyjen havainnoinnin, jolloin kohdistusta voidaan korjata. Vaatimus, että piirilevyporakoneella tulee pystyä poraamaan 300 mm * 300 mm kokoinen piirilevy, täyttyy, koska ohjainmoduulilla on teoriassa mahdollista suorittaa kyseisellä tarkkuudella jopa 1310,72 mm * 1310,72 mm kokoisen piirilevyn poraaminen. Käyttöliittymä muuntaa koordinaatit yhden askeleen tarkkuuteen ja lähettää ne mikrokontrollerille askelina. Piirilevyporan tarkkuudeksi saadaan yksi askel, joka on poramoduulin mekaniikasta johtuen 0,02 mm. Ohjainmoduuli ei ole riippuvainen mekaanisista ratkaisuista, joiden mukaan laitteen toistotarkkuus määräytyy. Pienin ohjainmoduulilla tehtävä tilan muutos on puoli askelmoottorin askelta, jolla on mahdollista nostaa toistotarkkuus 0,01 mm:iin. Tätä ominaisuutta ei kuitenkaan käytetä kyseisessä projektissa.

Käyttöliittymälle suoritetut testit osoittivat, että sarjaväyläliityntä käyttöliittymän ja mikrokontrollerin välillä toimii moitteettomasti. Tästä voidaan olettaa, ettei tule suuria ongelmia yhdistää käyttöliittymää, ohjauselektroniikkaa sekä poramoduulia toimivaksi piirilevyporakoneeksi.

Opinnäytetyön tekemisen aloittamista miettiessäni minulle tarjottiin mahdollisuutta tähän opinnäytetyöhön. Vaikka minulla olikin valmiina jo toinen aihe, oli tämä työ laajuutensa kannalta järkevämpi alkaa toteuttamaan. Työn aihe sopii hyvin Tietokone-elektroniikan opinnäytetyöksi, jota olen pääaineena opiskellut. Aloitin työn tekemisen kesällä 2004. Tutustuttuani piirilevyporamoduuliin ja sen teknisiin yksityiskohtiin aloin työn suunnittelemalla piirilevyporan ohjainmoduulin kytkentää. Tein kytkentäkaaviosuunnittelun ohessa piirilevyn layout-suunnittelua, sillä on helpompaa alkaa ”latomaan” komponentteja ja vetämään johdinvetoja sitä mukaa kun kytkennän suunnittelu etenee. Kytkentäkaavion tekemiseen käytin Pads PowerLogic -ohjelmaa ja piirilevysuunnittelussa Pads PowerPCB -ohjelmaa.

Ohjelmat voi synkronoida, jolloin kytkentä siirtyy suoraan ohjelmasta toiseen. Suunniteltuani piirilevyn valmistin sen kotikonstein ja hankin tarvittavat komponentit sekä ladoin ne paikoilleen. Ohjelmoinnin aloitin jo ennestään tutulla CodeVision, C-kääntäjällä, jota olin käyttänyt aiemmin muissa projekteissa. Työtä tehdessäni opin hyödyllistä tietoa C-kääntäjästä ja laiteläheisestä C-ohjelmointikielestä. Kaiken kaikkiaan tehtävä oli haastava ja mielenkiintoinen.

Jatkokehitysmahdollisuuksia ohjainmoduulin kehittämiseksi on olemassa, joskin kovin montaa I/O-nastaa ei jäänyt vapaaksi muihin tarkoituksiin. Sovelluskohteita sen sijaan on varmasti useita. Pidemmälle menevää jatkokehitystä koko piirilevyporakoneelle voisi olla piirilevyjyrsimen kehittäminen.

LÄHTEET

Engdahl, T. 1993.RS-232C [verkkojulkaisu]. [viitattu 5.9.2006]. Saatavissa: http://users.tkk.fi/~then/mytexts/rs-232c.html

Saarinen, A. 1994.Piirilevyporan ohjausjärjestelmä. Lahden Teknillinen Oppilaitos. Opinnäytetyö.

telos EDV Systementwicklung GmbH. I2C-BUS [verkkojulkaisu]. [viitattu 13.10.2006]. Saatavissa: http://www.i2c-bus.org

Tietomyrsky Oy. Passiivisen johdon kytkentä [verkkojulkaisu]. [viitattu 16.10.2006]. Saatavissa: http://www.tietomyrsky.fi/kuvat/avrispjohdot.gif

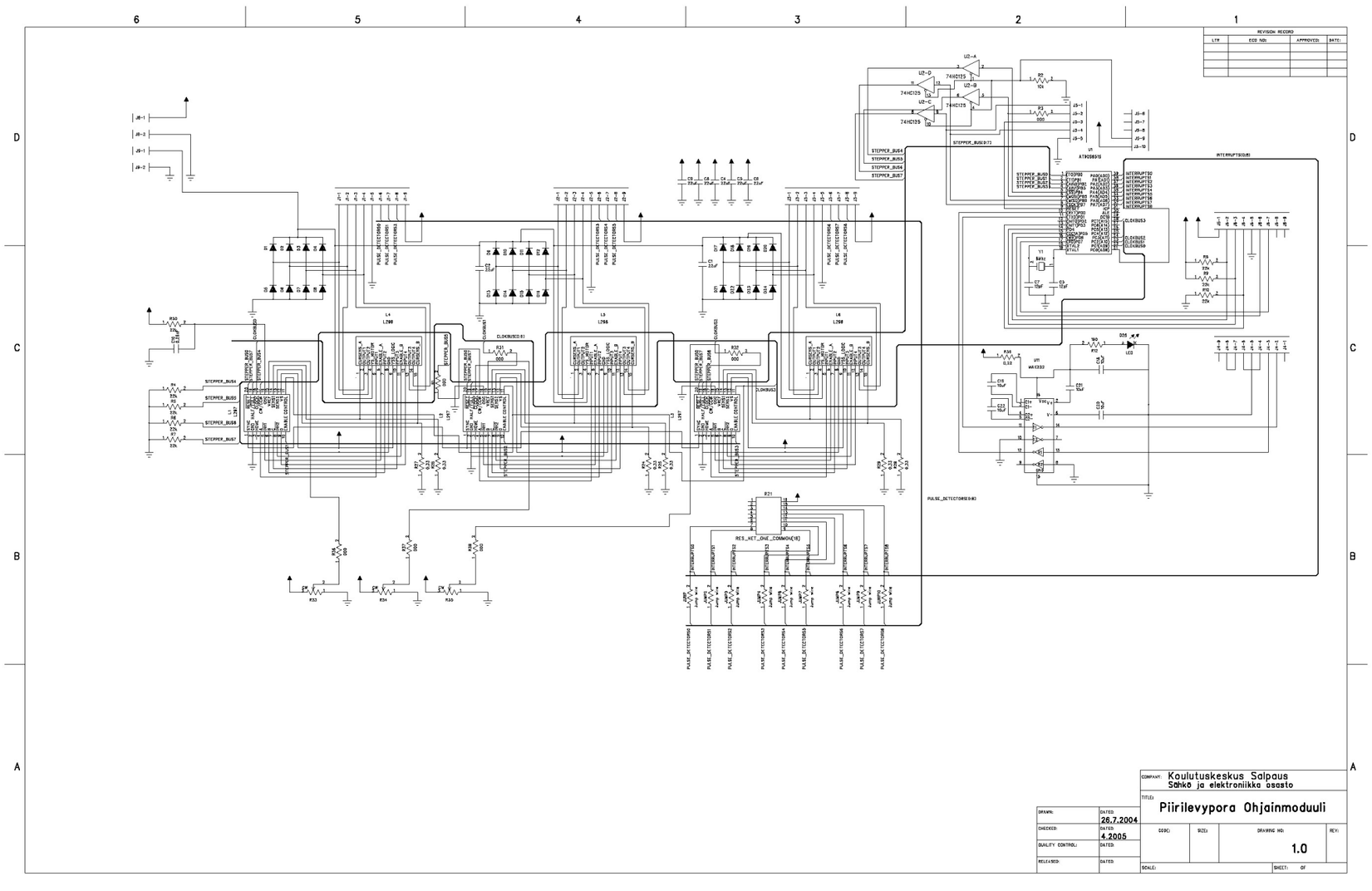
LIITTEET

LIITE 1. Komponenttiluettelo

Tunnus	Sijoitus tunnus		Arvo	Nimitys
C1	1,2,4,5,6,8,9		22uF,	
C2	18,19,20,21,22		10uF,	
C3	3,7		12pF,	
C4	10		0,26nF,	
D1	1,2,3,4,5,6,7,8,9,10,11,12,13,14		BAW62	
D1	15,16,17,18,19,20,21,22,23,24		BAW62	
D2				LED
J1	DCON9F			D- liitin
J2	DCON9F			D- liitin
J3	DCON9F			D- liitin
J4	DCON9F			D- liitin
J5	HEADER10			Piikki rima
J6	DCON9F			D- liitin
J7	DCON9F			D- liitin
J8	CON-MATENLOK-2P			Virta liitin
J9	CON-MATENLOK-2P			Virta liitin
K1	AK-1604			4x4- näppäimistö
L1	1,2,3		L297	Askelmoottorikontrolleri
L2	4,5,6		L298	H- siltapiiri
Q1	BD139			Transistori
R1	2		1/8W,10k	
R2	4,5,6,7,8,9,10,23,30		1/8W,22k	
R3	12		1/4W,190	
R4	RES_NET_ONE_COMMON(16)		10k	Vastussilta
R5	22,33,34,35	VRES-TOP-ADJ		Potentiometri
R6	24,25,26,27,28,29		3W 0,33	Tehovastus
R7	39		1/4W 0,22	
U1	AT90S8515			Atmel mikrokontrolleri
U2	74HC125			TTL- logiikka
U3	PCF8574			I2C- I/O laajennin
U4	PCF8574			I2C- I/O laajennin
U5	L1634			Lcd näyttö
U11	MAX232			Usart- linjamuunnin
Y1	8Mhz			Kide

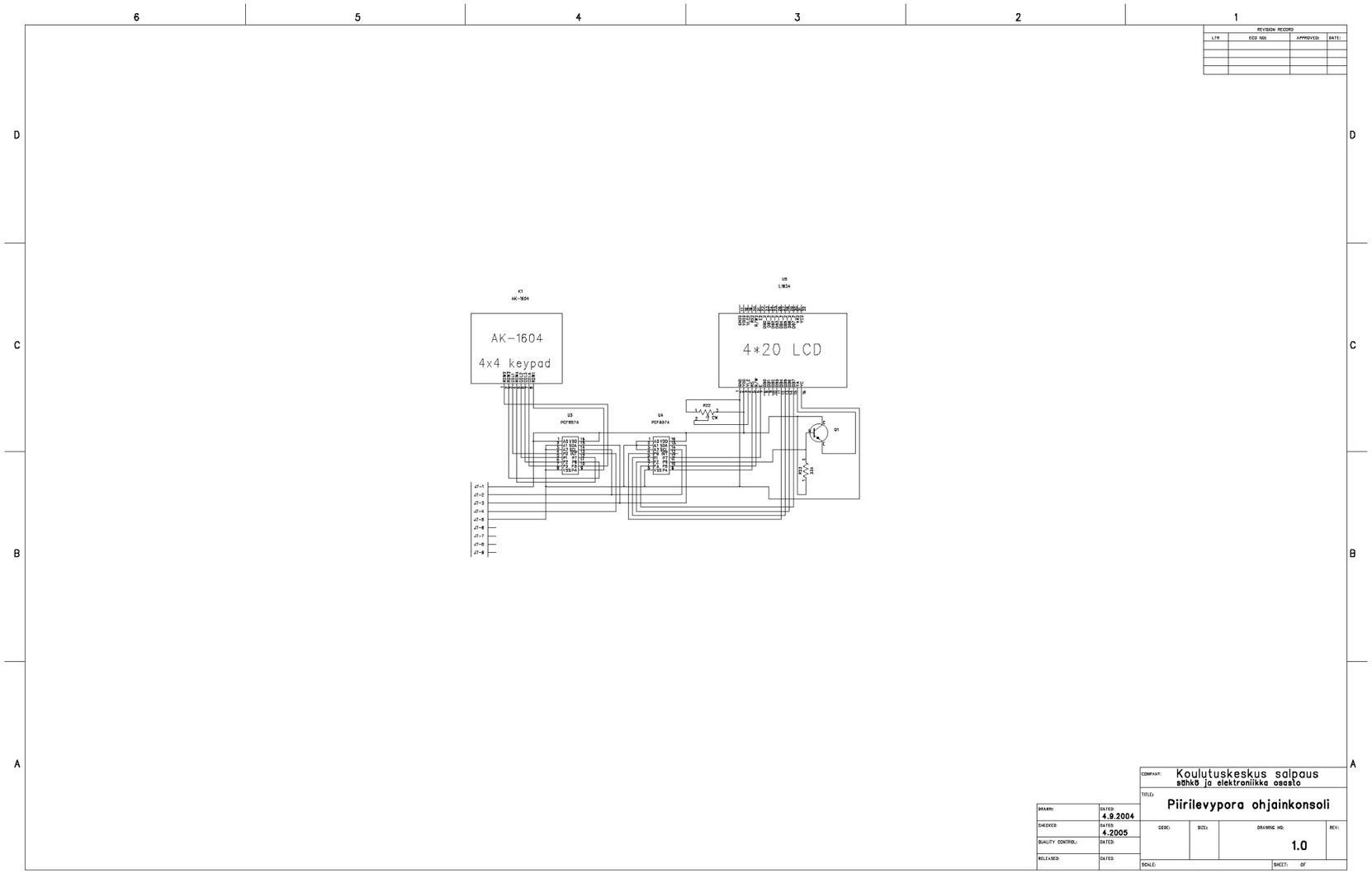
LIITE 2. Mikrokontrollerimoduulin piirikaavio

stepper drivers2.sch-1 - Mon Nov 20 19:46:44 2006

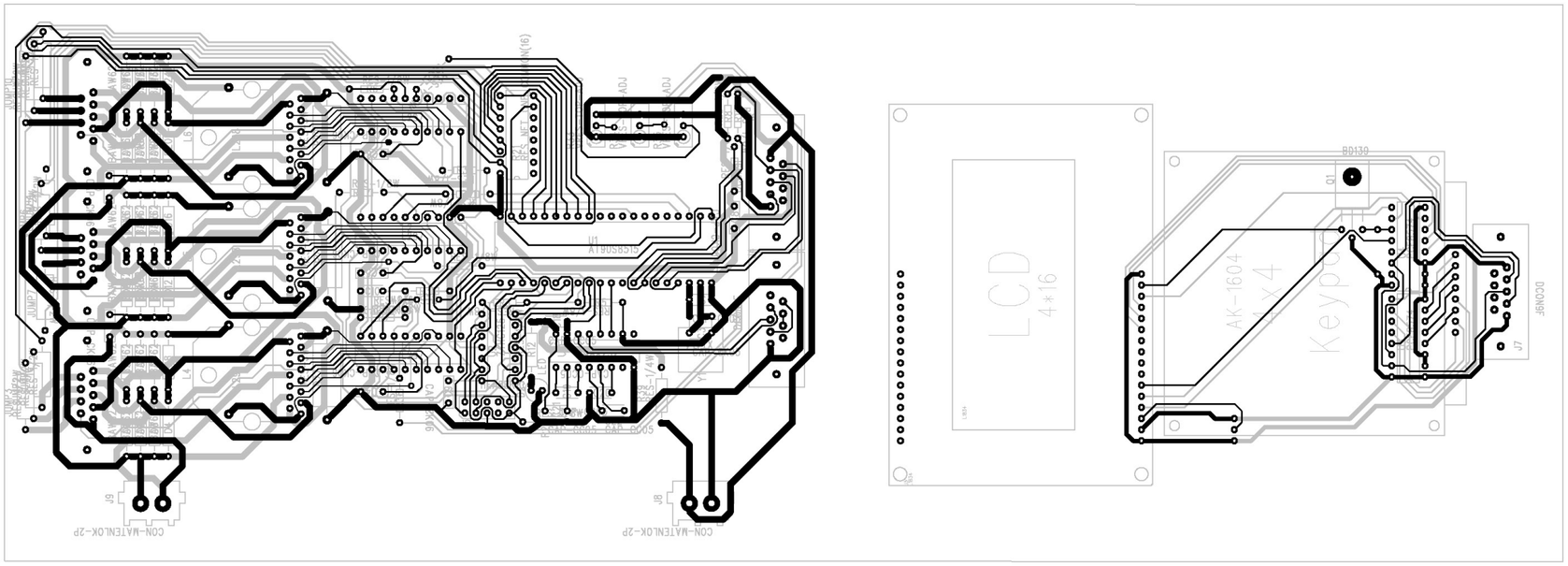


LIITE 3. Ohjainkonsolin piirikaavio

stepper drivers2.sch-2 - Mon Nov 20 21:43:02 2006



LIITE 4. Mikrokontrollerimoduulin ja ohjainkonsolin LAYOUT-piirustukset



Stepper drivers3.pcb – Mon Nov 20 21:55:32 2006

LIITE 5. ATmega 162-piirin sisäisten laitteiden I/O rekisterimääritykset

```
// CodeVisionAVR C Compiler
// (C) 1998-2004 Pavel Haiduc, HP InfoTech S.R.L.

// I/O rekisterien määritykset ATmega162:een

#ifndef _MEGA162_INCLUDED_
#define _MEGA162_INCLUDED_

#pragma used+
sfrb UBRR1L=0;
sfrb UCSR1B=1;
sfrb UCSR1A=2;
sfrb UDR1=3;
sfrb OSCCAL=4;
sfrb OCDR=4;
sfrb PINE=5;
sfrb DDRE=6;
sfrb PORTE=7;
sfrb ACSR=8;
sfrb UBRR0L=9;
sfrb UCSR0B=0xa;
sfrb UCSR0A=0xb;
sfrb UDR0=0xc;
sfrb SPCR=0xd;
sfrb SPSR=0xe;
sfrb SPDR=0xf;
sfrb PIND=0x10;
sfrb DDRD=0x11;
sfrb PORTD=0x12;
sfrb PINC=0x13;
sfrb DDRC=0x14;
sfrb PORTC=0x15;
sfrb PINB=0x16;
sfrb DDRB=0x17;
sfrb PORTB=0x18;
sfrb PINA=0x19;
sfrb DDRA=0x1a;
sfrb PORTA=0x1b;
sfrb EECR=0x1c;
sfrb EEDR=0x1d;
sfrb EEARL=0x1e;
sfrb EEARH=0x1f;
sfrw EEAR=0x1e;   // 16 bittinen käsittely
sfrb UBRR0H=0x20;
sfrb UCSR0C=0x20;
sfrb WDTCR=0x21;
sfrb OCR2=0x22;
sfrb TCNT2=0x23;
sfrb ICR1L=0x24;
sfrb ICR1H=0x25;
sfrb ASSR=0x26;
sfrb TCCR2=0x27;
sfrb OCR1BL=0x28;
sfrb OCR1BH=0x29;
sfrw OCR1B=0x28;  // 16 bittinen käsittely
```

LIITE 5 jatkuu

```
sfrb OCR1AL=0x2a;
sfrb OCR1AH=0x2b;
sfrw OCR1A=0x2a;  // 16 bittinen käsittely
sfrb TCNT1L=0x2c;
sfrb TCNT1H=0x2d;
sfrw TCNT1=0x2c;  // 16 bittinen käsittely
sfrb TCCR1B=0x2e;
sfrb TCCR1A=0x2f;
sfrb SFIOR=0x30;
sfrb OCR0=0x31;
sfrb TCNT0=0x32;
sfrb TCCR0=0x33;
sfrb MCUCSR=0x34;
sfrb MCUCR=0x35;
sfrb EMCUCR=0x36;
sfrb SPMCR=0x37;
sfrb TIFR=0x38;
sfrb TIMSK=0x39;
sfrb GIFR=0x3a;
sfrb GICR=0x3b;
sfrb UBRR1H=0x3c;
sfrb UCSR1C=0x3c;
sfrb SPL=0x3d;
sfrb SPH=0x3e;
sfrb SREG=0x3f;
#pragma used-

#define CLKPR  (*(unsigned char *) 0x61)
#define PCMSK0 (*(unsigned char *) 0x6b)
#define PCMSK1 (*(unsigned char *) 0x6c)
#define ETIFR  (*(unsigned char *) 0x7c)
#define ETIMSK (*(unsigned char *) 0x7d)
#define ICR3L  (*(unsigned char *) 0x80)
#define ICR3H  (*(unsigned char *) 0x81)
#define OCR3BL (*(unsigned char *) 0x84)
#define OCR3BH (*(unsigned char *) 0x85)
#define OCR3AL (*(unsigned char *) 0x86)
#define OCR3AH (*(unsigned char *) 0x87)
#define TCNT3L (*(unsigned char *) 0x88)
#define TCNT3H (*(unsigned char *) 0x89)
#define TCCR3B (*(unsigned char *) 0x8a)
#define TCCR3A (*(unsigned char *) 0x8b)

// Keskeytys vektori määritykset

#define EXT_INT0 2
#define EXT_INT1 3
#define EXT_INT2 4
#define PCINT0 5
#define PCINT1 6
#define TIM3_CAPT 7
#define TIM3_COMPA 8
#define TIM3_COMPB 9
#define TIM3_OVF 10

#define TIM2_COMP 11
```

LIITE 5 jatkuu

```
#define TIM2_OVF 12
#define TIM1_CAPT 13
#define TIM1_COMPA 14
#define TIM1_COMPB 15
#define TIM1_OVF 16
#define TIM0_COMP 17
#define TIM0_OVF 18
#define SPI_STC 19
#define USART0_RXC 20
#define USART1_RXC 21
#define USART0_DRE 22
#define USART1_DRE 23
#define USART0_TXC 24
#define USART1_TXC 25
#define EE_RDY 26
#define ANA_COMP 27
#define SPM_RDY 28

#endif
```

LIITE 6. Pääohjelmassa suoritetavat mikrokontrollerikohtaiset laitealustukset

```
void main(void)
{
// tänne määritetään paikalliset muuttujat
// unsigned char data;

// Kide oskillaattorin jako kerroin: 1
CLKPR=0x80;
CLKPR=0x00;

// Sisään/ulos portien määrittely
// Port A Alustus
// Func7=Sisään Func6=Sisään Func5=Sisään Func4=Sisään Func3=Sisään Func2=Sisään Func1=Sisään
Func0=Sisään
// Tila7=T Tila6=T Tila5=T Tila4=T Tila3=T Tila2=T Tila1=T Tila0=T
PORTA=0x00;
DDRA=0x00;

// Port B Alustus
// Func7=Ulos Func6=Ulos Func5=Ulos Func4=Ulos Func3=Ulos Func2=Ulos Func1=Ulos Func0=Ulos
// Tila7=0 Tila6=0 Tila5=0 Tila4=0 Tila3=0 Tila2=0 Tila1=0 Tila0=0
PORTB=0x00;
DDRB=0xFF;

// Port C Alustus
// Func7=Ulos Func6=Ulos Func5=Sisään Func4=Sisään Func3=Ulos Func2=Ulos Func1=Ulos
Func0=Sisään
// Tila7=0 Tila6=0 Tila5=T Tila4=T Tila3=0 Tila2=0 Tila1=0 Tila0=T
PORTC=0x00;
DDRC=0xFE;

// Port D Alustus
// Func7=Sisään Func6=Sisään Func5=Sisään Func4=Sisään Func3=Sisään Func2=Sisään Func1=Sisään
Func0=Sisään
// Tila7=T Tila6=T Tila5=T Tila4=T Tila3=T Tila2=T Tila1=T Tila0=T
PORTD=0x00;
DDRD=0x00;

// Port E Alustus
// Func2=Sisään Func1=Sisään Func0=Sisään
// Tila2=T Tila1=T Tila0=T
PORTE=0x00;
DDRE=0x00;

// Ajastin/laskuri 0 Alustus
// Kello lähde: Järjestelmä kello
// KEllo value: Timer 0 Pysäytetty
// Moodi: Normaal1 huippu = 0xFFh
// OC0 Ulos: Irti kytketty
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Ajastin/laskuri 1 Alustus
// Kello lähde: Järjestelmä kello
// Kello arvo: 7,813 kHz
```

```
// Moodi: Normaali huippu = 0xFFFFh
// OC1A Ulostulo: Irti kytketty.
// OC1B Ulostulo: Irti kytketty.
// Kohinan poisto: Pois kytketty
// Sisääntulon kello laskevalla reunalla
TCCR1A=0x00;
TCCR1B=0x05;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Ajastimen/Ajastimien & Laskurin/Laskurien & Keskeytyksen/Keskeytyksien Alustus
TIMSK=0x00;
ETIMSK=0x00;

// Ajastin/laskuri 2 Alustus
// Kello lähde: Järjestelmä kello
// Kello arvo: 62,500 kHz
// Moodi: Normaali huippu = 0xFFh
// OC2 Ulosput: Irti kytketty.
ASSR=0x00;
TCCR2=0x05;
TCNT2=0x00;
OCR2=0x00;

// Ulkoisten keskeytysten alustus
// INT0: Pois
// INT1: Päällä
// INT1: Moodi: Laskevalla reunalla
// INT2: Pois
// Keskeytys jokaisen pinnin muutoksesta PCINT0-7: Päällä
// Keskeytys jokaisen pinnin muutoksesta PCINT8-15: Päällä
GICR|=0x98;
PCMSK0=0xFF;
PCMSK1=0x01;
MCUCR=0x08;
EMCUCR=0x00;
GIFR=0x98;

// USART0 Alustus
// Kommunikointi parametrit: 8 Data, 1 Stop bitti, Ei pariteettia
// USART0 Receiver: Päällä
// USART0 Transmitter: Päällä
// USART0 Moodi: Asynkroninen
// USART0 Baud määrä: 9600
UCSR0A=0x00;
UCSR0B=0xD8;
UCSR0C=0x86;
UBRR0H=0x00;
UBRR0L=0x33;
```

```
// Analog Comparator Alustus
// Analog Comparator: Off
// Analog Comparator Sisäänput Capture by Timer/Counter 1: Off
// Analog Comparator Ulosput: Off
ACSR=0x80;

// I2C Väylän Alustus
i2c_init();


// Globaali keskeytysten sallinta
#asm("sei")

stepper_setup();
console_setup();
```

Liite 7. Pääohjelmaluuppi

```
while (1)
      {

      apu = getchar();          // vastanottaa tavun

switch (apu) {

      case 0x61: {              // hex 0x61 on kirjain a (aloitus käsky)

      putchar(0x61);            // Kuittaa aloituksen (rs-232)
      getint();                 // c on 16-bittinene apu muuttuja, joka viedään
      z_tx_input = usart_int; // z-akselin ajo koordinaatiksi. lopullisessa versiossa tämä koordinaatti
                                // viedään x-akselin ajo koordinaatiksi.
      stepper_drive_z();        // ajetaan moottori kyseiseen koordinaattiin
      getint();                 // vastaan otetaan toinen tavupari protokortissa vain kättelyn takia.
                                // lopullisessa versiossa tämä koordinaatti viedään y-akselin ajo koordinaatiksi.
}
break;

      case 0x6c: {          // hex 0x6c on kirjain l (lopetuskäsky).
                            // Mikäli Tietokone antaa tämän tavun. tietää kontrolleri lopettaa ajon.

      putchar(0x6c);        // Kuittaa lopetuksen (rs-232)
      z_tx_input = 0;       // asetetaan nolla pisteen koordinaatti z-akselille.
      stepper_drive_z();  // ja ajetaan siihen
}
break;
      }
```

Liite 8. Askelmoottorien ajoalustukset ja ajofunktiot

```
/*
  Jani Kuhlman
  jkuhlman@lpt.fi   8.2.2006
  Prototypes for stepper management functions
*/

// unsigned int x_tx_input, y_tx_input;    // Luetaan koordinaatti x, y, ja z
// volatile unsigned int z_tx_input;       // näihin muuttujiin. rs-232 portista. Volatile siksi, että
register unsigned int x;
register unsigned int y;                // muuttujia x,y, ja z käytetään keskeytyksissä, jolloin ne on parempi sijoittaa
rekisteriin
register unsigned int z;
unsigned char stepper_delay = 1;        // Askeleen väli millisekunteina 0-510ms. sijoitettava arvo 1-255 Oletus
500ms/askel
bit suunta;                             // 1= Eteenpäin 0= Taaksepäin

      void stepper_setup(void)
      {

      x=32768;
      y=32768;                  // vastaa nolla kohtaa
      z=32768;
      x_tx_input = 0;
      y_tx_input = 0;            // x/y/z koordinaatistosta
      z_tx_input = 0;

      PORTB = 0x00;                 // Enable reset, Suunta taakse
      DDRB = 0xff;

      DDRC = 0xfe;
      PORTC = 0x40;                 // Enable led
      delay_ms(20);

      }

 void stepper_drive_x (void)

  {

      x_tx_input = (x_tx_input + 32768);



        while (x != x_tx_input)
         {

      if (x > x_tx_input) {   // Määrätään x-poran suunta
            suunta = 0;
       }
      else {
            suunta = 1;
       }

      if (suunta == 0) {
            PORTB = 0x83;  // 03=full 83=half, Disable reset, Suunta taakse, Enable L1
```

```
    }
    else {
          PORTB = 0xf3;  // 73=full f3=half, Disable reset, Suunta eteen, Enable L1
    }

          PORTC = 0x02;                 // Pulssi ylös
          delay_ms(stepper_delay);
          PORTC = 0x40;                 // Pulssi alas
          delay_ms(stepper_delay);

    //      if (x_tx_input > x)
     //       {  x++; }                   // vähentää x:n arvoa. korvataan optojen antamalla tiedolla
      //                                  // kunhan saadaan ne kytkettyä.
       //    else { x--; }


         if (x == x_tx_input) {
         PORTB = PORTB &= 0x01;          // Disable L1
          }
       }

        x_tx_input = (x_tx_input -32768);

}


void stepper_drive_y (void)
{
    y_tx_input = (y_tx_input / 0.6);      // kerroin, jota käytetään, koska käyttöliittymän
                                          // askelmäärä on määritelty väärin (600askelta/ 1cm)
    y_tx_input = (y_tx_input + 32768);

      while (y != y_tx_input)
       {

    if (y > y_tx_input) {   // Määrätään y-poran suunta
          suunta = 0;
    }
    else {
          suunta = 1;
    }

    if (suunta == 0) {
          PORTB = 0x85;  // 05=full 85=half, Disable reset, suunta taakse, Enable L2
    }
    else {
          PORTB = 0xf5;  // 75=full f5=half, Disable reset, Suunta eteen, Enable L2
    }

          PORTC = 0x04;                 // Pulssi ylös
          delay_ms(stepper_delay);
          PORTC = 0x40;                 // Pulssi alas
          delay_ms(stepper_delay);

         if (y_tx_input > y)
         {  y++; }            // vähentää y:n arvoa. korvataan optojen antamalla tiedolla
```

```
                        // kunhan saadaan ne kytkettyä.
          else { y--; }

          if (y == y_tx_input) {
          PORTB = PORTB &= 0x01;          // Disable L2
            }
         }

         y_tx_input = (y_tx_input -32768);
         y_tx_input = (y_tx_input * 0.6);   // kertoimen purku
}


void stepper_drive_z (void)
{

    z_tx_input = ( z_tx_input / 0.6);   // kerroin, jota käytetään, koska käyttöliittymän
                                // askelmäärä on määritelty väärin (600askelta/ 1cm)
    z_tx_input = (z_tx_input + 32768);

      while (z != z_tx_input)
       {

    if (z > z_tx_input) {   // Määrätään z-poran suunta
          suunta = 0;
    }
    else {
          suunta = 1;
    }

    if (suunta == 0) {
          PORTB = 0x89;  // 09=full 89=half, Disable reset, suunta taakse, Enable L3
    }
    else {
          PORTB = 0xf9;  // 79=full f9=half, Disable reset, Suunta eteen, Enable L3
    }

          PORTC = 0x08;                 // Pulssi ylös
          delay_ms(stepper_delay);
          PORTC = 0x40;                 // Pulssi alas
          delay_ms(stepper_delay);

          if (z_tx_input > z)
           {  z++; }            // vähentää z:n arvoa. korvataan optojen antamalla tiedolla
                               // kunhan saadaan ne kytkettyä.
          else { z--; }

         if (z == z_tx_input) {
         PORTB = PORTB &= 0x01;          // Disable L3
           }
        }

        z_tx_input = (z_tx_input -32768);
        z_tx_input = ( z_tx_input * 0.6);  // kertoimen purku
}
```

Liite 9 Pulssianturien lukufunktiot

```
// CodeVisionAVR C Compiler
// Interrupt functions

#include <mega162.h>

bit op1_set = 0;
bit op2_set = 0;    // Ehdot, joiden pitää täyttyä, jotta muuttujaa voidaan
bit op3_set = 0;    // kasvattaa, tai pienentää.
bit op1_v = 0;
bit op2_v = 0;
bit op3_v = 0;
bit testi = 0;

// Ulkoisen keskeytyksen 1 palvelu rutiini
// Tätä keskeytystä käytetään näppäimistön lukuun
interrupt [EXT_INT1] void ext_int1_isr(void)
{

if (testi == 0)
      {
PORTC = 0x40;
testi = 1;
      }
else
      {
PORTC = 0x00;
testi = 0;
      }
}

// Pinnien 0-7 keskeytyspalvelu rutiini

interrupt [PCINT0] void pin_change_isr0(void)
{

char op1,op2,op3;      // optojen vertailutiedon väliaikainen varasto
op3 = (0xc0 & PINA);
op2 = (0x18 & PINA);
op1 = (0x03 & PINA);


switch (op3){

case 0x40:            // Koodi, joka suoritetaan, jos op3 == tämä arvo
 if (op3_set == 1){
 z++;
 op3_set = 0;
 }
  break;
                  // z-akselin opton käsittely
case 0xc0:
 if (op3_v == 1){
 op3_set = 1;
 op3_v = 0;
 }
```

```
  break;

case 0x80:
 if (op3_set == 1){
 z--;
 op3_set = 0;
 }
  break;

case 0x00:
 op3_v = 1;
  break;
 }                    // end of z

switch (op2){

case 0x08:
 if (op2_set == 1){
 y++;
 op2_set = 0;
 }
  break;
                     // y-akselin optojen käsittely
case 0x18:
 if (op2_v == 1){
 op2_set = 1;
 op2_v = 0;
 }
  break;

case 0x10:
 if (op2_set == 1){
 y--;
 op2_set = 0;
 }
  break;

case 0x00:
 op2_v = 1;
  break;
 }                    // end of y

switch (op1){

case 0x01:
 if (op1_set == 1){
 y++;
 op1_set = 0;
 }
  break;
                     // x-akselin optojen käsittely
case 0x03:
 if (op1_v == 1){
 op1_set = 1;
 op1_v = 0;
 }
```

```
  break;

case 0x02:
 if (op1_set == 1){
 y--;
 op1_set = 0;
 }
  break;

case 0x00:
 op1_v = 1;
 break;
 }
                    // end of x
}


// Pinnien 8-15 keskeytyspalvelu rutiini
interrupt [PCINT1] void pin_change_isr1(void)
{


}
```

Liite 10. Ohjainkonsolin luku- ja kirjoitusfunktiot

```
/*
  Jani Kuhlman
  jkuhlman@lpt.fi   8.2.2006
  Prototypes for console management functions
*/

#define KEYPAD_READ_ADDRESS 0x43        // Ohjainkonsolin luku,
#define LCD_WRITE_ADDRESS 0x40          // ja kirjoitusosoitteet

void console_setup() {
unsigned char data;
//PORTD = ( 0x08 | PORTD);
PORTD = (~0x08 & PORTD);
DDRD = (~0x08 & DDRD);
i2c_start();
i2c_write(0x43);
}


// kirjoittaa tavun ohjainkonsolin nestekidenäytön mikrokontrollerille.

void i2cwrite(unsigned char address, unsigned char data) {

i2c_start();
i2c_write(LCD_WRITE_ADDRESS);
i2c_write(address);
i2c_write(data);
i2c_stop();
}

void i2cread(unsigned char address, unsigned char data) {

i2c_start();
i2c_write(KEYPAD_READ_ADDRESS);
i2c_read(data);
i2c_stop();
}
```