POHJOIS-SAVON
AMMATTI-
KORKEAKOULU

Tekniikka, Kuopio

Janne Ahonen IA-224

OPINNÄYTETYÖ

DSP56002 JAKOSUODINKORTIN SUUNNITTELU

Työn valvoja: Arto Toppinen

22.8.2002

POHJOIS-SAVON AMMATTIKORKEAKOULU – TEKNIIKKA, KUOPIO

Koulutusohjelma

Sähkötekniikan koulutusohjelma

Tekijä

Janne Ahonen

Työn nimi

DSP56002-jakosuodinkortin suunnittelu

Työn laji	Päiväys	Sivumäärä
Päättötyö	22.8.2002	63 + 29

Työn valvoja

Yliopettaja Arto Toppinen

Tiivistelmä

Jokaisessa HiFi-laatuisessa kaiuttimessa tarvitaan jakosuodin, koska ei ole olemassa sellaista kaiutinelementtiä, joka toistaisi koko 20 Hz-20 kHz taajuusalueen. Jakosuotimen toteutus voi olla joko passiivinen tai aktiivinen. Aktiivinen toteutus, joka voi olla analoginen tai digitaalinen, tarjoaa mielenkiintoisia mahdollisuuksia. Digitaalisia signaaliprosessoreita käytetään äänentoistolaitteissa nykyään melko paljon. Huolimatta digitaalisten suotimien ilmeisistä eduista, niitä ei ole juuri käytetty kaiuttimien jakosuotimissa.

Tämän työn tarkoituksena oli suunnitella ja valmistaa sopiva signaaliprosessorikortti, jota voidaan käyttää jakosuotimena aktiivikaiuttimessa, ja vertailla joitakin suotimia analogisiin vastineisiinsa. Kortti suunniteltiin Motorolan 24-bittisen signaaliprosessorin DSP56002 pohjalle, jonka 24 bitin sananpituus mahdollistaa 144 dB:n dynaamisen alueen, joten äänenlaatu ei käsittelyssä heikkene. Digitaalinen toteutustapa sallii myös suodinkertoimien helpon muokkauksen kuuntelun aikana. Analogiaosan sydämenä on Crystalin 16-bittinen 4-kanavainen koodekki CS4225, joka sisältää kaksi A/D- ja neljä D/A-muunninta.

Monipuolisen koodekin ansiosta korttia voidaan käyttää myös muihin tarkoituksiin esim. Dolby ProLogic -dekoodaukseen ja autoäänentoistosovelluksiin. Suotimet toteutettiin FIR-tyypin digitaalisuotimella. Näin suotimesta saatiin täysin vaihevirheetön. Suotimien suunnitteluohjelmistona käytettiin MathWorksin MATLAB-ohjelmistoa.

Avainsanat

jakosuodin, DSP, FIR, aktiivikaiutin, vaihelineaarinen

Luottamuksellisuus

Julkinen

POHJOIS-SAVO POLYTECHNIC – SCHOOL OF ENGINEERING

Degree Programme

Electrical Engineering

Author

Janne Ahonen

Title of Project

Design of the DSP56002 Crossover Card

Type of Project	Date	Pages
Final Project	22.8.2002	63 + 29

Academic Supervisor

senior lecturer Arto Toppinen

Abstract

A crossover is needed in every hifi-quality speaker, because there is no transducer that could reproduce the entire 20 Hz-20 kHz frequency range satisfactorily. This crossover realization can be either passive or active. Active design approach offers interesting possibilities. Active filters can be subdivided into analog and digital realizations. The use of digital signal processors in the audio equipment of consumer electronics is quite common nowadays. Despite of the obvious advantages DSPs have not been widely used in speaker crossovers.

The purpose of this study was to design a suitable stand-alone digital signal processor board, which could be used as a crossover filter in an active loudspeaker system and to compare some filters against their analog counterparts. The board was designed around Motorola's DSP56002 24-bit digital signal processor which provides 144 dB dynamic range to prevent sound quality deterioration. The heart of the analog section is Crystal's 16-bit four-channel codec CS4225, which contains two A/D and four D/A converters. Filters were realized using FIR-type digital filter.

All phase distortion problems occuring in analog filter designs can be eliminated. Several filter configurations were created using Mathworks's MATLAB signal processing toolbox. The digital approach allows also dynamic updating of filter coefficients, so even during listening the filter can be easily modified. Because of the versatile codec, the board can also be used for other tasks like. Dolby® Surround ProLogic® decoding and car audio applications.

Keywords

crossover, DSP, FIR, active speaker, phase linear

Confidentiality

public

ALKUSANAT

Digitaalisen suodatuksen käyttö jakosuotimissa on ollut vähän tutkittu aihe. Myöskään kaupallisesti ei ole ollut juuri saatavilla kortteja, joilla pystyisi toteuttamaan reaaliaikaisen vaihevirheettömän suotimen, ei ainakaan kohtuuhintaan. Tällaisen kortin suunnittelulle tuntui olevan jonkinlainen tilaus, ainakin kiinnostuneita on riittänyt. Siispä päätin suunnitella kortin ja siihen tarvittavat ohjelmistot itse. Tämä sattui juuri sopivasti insinöörityön kannalta sopivana hetkenä ja niinpä siitä tuli insinöörityön aihe.

Tämäntyyppinen laite on työn tekohetkellä erittäin ainutlaatuinen, vastaavia järjestelmiä ei liene tietääkseni tehty. Lähes vastaava laite on Jukka Partasen insinöörityössään vuonna 1994 suunnittelema kortti, tosin se on PC:n ISA-kortiksi tehty, eikä sinällään sisällä A/D- tai D/A-muunninta. Partasen suunnittelema kortti on tosin osasyyllinen tämän kortin suunnitteluun, ideaa olen kehitellyt aina siitä asti kun ensimmäisiä kertoja kuulin digitaalisesta suodatuksesta ja sen mahdollisuuksista.

Haluan tässä yhteydessä kiittää työn valvojaa yliopettaja Arto Toppista, Aapo Olkkosta avustamisessa osien hankinnassa ja layoutin teossa sekä motivoinnissa tekemään suodinohjelmistoon tietokoneohjausmahdollisuus, Instele Oy:n Olavi Turusta hyvästä palvelusta, pohjois-savon ammattikorkeakoulun tuotekehityslaboratoriota EMC-mittauksista, Maximia hyvin toimivasta näytepiirien tilausjärjestelmästä, Motorolaa opiskelijalle sopivista ohjelmankehitystyökalujen hinnoittelusta (kääntäjän voi imuroida ilmaiseksi Internetistä) sekä niiden laadukkuudesta (käytetty asm56000-kääntäjä hakkaa mennen tullen Texasin starter kitin assemblerin 10-0) sekä Integrated Electronics OY:n Ahti Partasta koodekin pienien erien tilaamisesta, ilman tätä keskeistä komponenttia koko kortti ei olisi ikinä toteutunut. Ja tietysti Telarcia laadukkaista äänityksistä.

Kuopiossa, 22.8.2002

Janne Ahonen

SISÄLLYSLUETTELO

1 JOHDANTO

Työn tarkoituksena oli toteuttaa 2-tiekaiuttimeen sopiva digitaalinen suodinyksikkö. Tätä varten suunniteltiin audiosignaalien käsittelyyn soveltuva digitaalinen signaaliprosessorikortti. Kortilla tuli olla mahdollisuus neljän kanavan lähtöön ja kahden kanavan tuloon, jotta sillä voitaisiin käsitellä stereosignaalia. Kortin tuli olla itsenäisesti toimiva, jolloin sen käynnistykseen ei tarvita mitään tietokoneohjelmia tai muuta erillisen tietokoneen apua, mutta siinä tuli olla mahdollisuus päivittää sovelluksen parametrejä RS-232C-liitännän kautta. Kortilla tuli myös olla erillinen parametrimuisti, johon voidaan tallentaa esim. suodinkertoimet. Tämän lisäksi suunniteltiin kortille sopiva suodinohjelma.

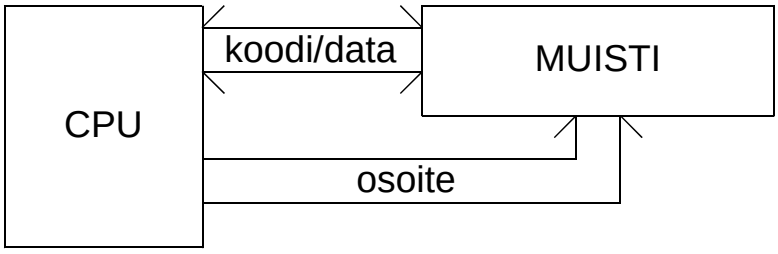
2 DIGITAALISET SIGNAALIPROSESSORIT

2.1 Mikä on DSP?

DSP eli digitaalinen signaaliprosessori on RISC-tyyppinen suppean käskykannan prosessori, joka on optimoitu suorittamaan signaalinkäsittelyssä tarvittavia operaatioita mahdollisimman nopeasti. Keskeinen operaatio signaalinkäsittelyssä on tulojen summan laskeminen (mac-operaatio). Koska pyritään mahdollisimman nopeaan kerto-yhteenlaskuoperaatioiden suorittamiseen, tarvitaan kovolla toteutettu kertoja. Tyypillisiä ominaisuuksia ovat myös tuki FFT:n vaatimalle bittikäännetylle osoitukselle sekä nopeat keskeytysten vasteajat. Signaaliprosessorit sisältävät myös yleensä jonkin verran sisäistä data- ja ohjelmamuistia.

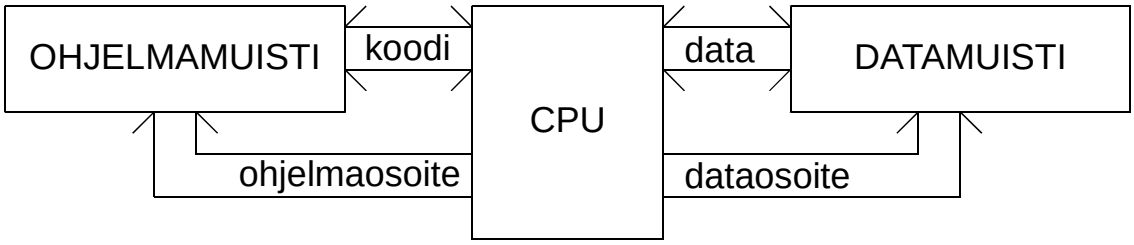
2.2 Arkkitehtuurit

Prosessori voidaan toteuttaa kahta eri arkkitehtuuria käyttäen. Yleiskäyttöiset mikroprosessorit perustuvat yleensä von Neumann-arkkitehtuuriin, jossa koodi ja data ovat samassa muistissa. Tästä aiheutuu se, että prosessoriydin ei periaatteessa ole sataprosenttisessa käytössä, koska data ja koodi täytyy hakea muistista peräkkäin. Tosin esim. Motorolan MC68060 ja Intelin Pentium-prosessorit soveltavat sisäisesti Harvard-arkkitehtuuria, vaikka ulkoisesti ovatkin von Neumann -tyyppisiä prosessoreita.



Kuva 1. Von Neumann -arkkitehtuuri

Vaihtoehtoinen arkkitehtuuri on Harvard -arkkitehtuuri, jossa ohjelmakoodille ja datalle on omat muistinsa. Näin voidaan hakea ohjelmakoodi ja käsiteltävä data yhtä aikaa muistista ilman peräkkäisten muistihakujen aiheuttamaa hidastumista. Kuitenkin ohjelmoijan pitää pystyä hallitsemaan monta rinnakkaista muistiavaruutta, mikä monimutkaistaa ohjelmointia. Datamuisteja on yleensä kaksi, jolloin molemmat operandit voidaan hakea muistista samanaikaisesti.



Kuva 2. Harvard-arkkitehtuuri

Texas Instruments käyttää omissa signaaliprosessoreissaan modifioitua Harvard-arkkitehtuuria, jossa on dual-access- (DARAM) ja single-access (SARAM) -tyyppistä RAM muistia. Prosessori pystyy tekemään yhden muistijakson aikana kirjoitus- ja lukuoperaation samanaikaisesti DARAM-tyyppiseen muistiin ja luku- tai kirjoitusoperaation SARAM-tyyppiseen muistiin.

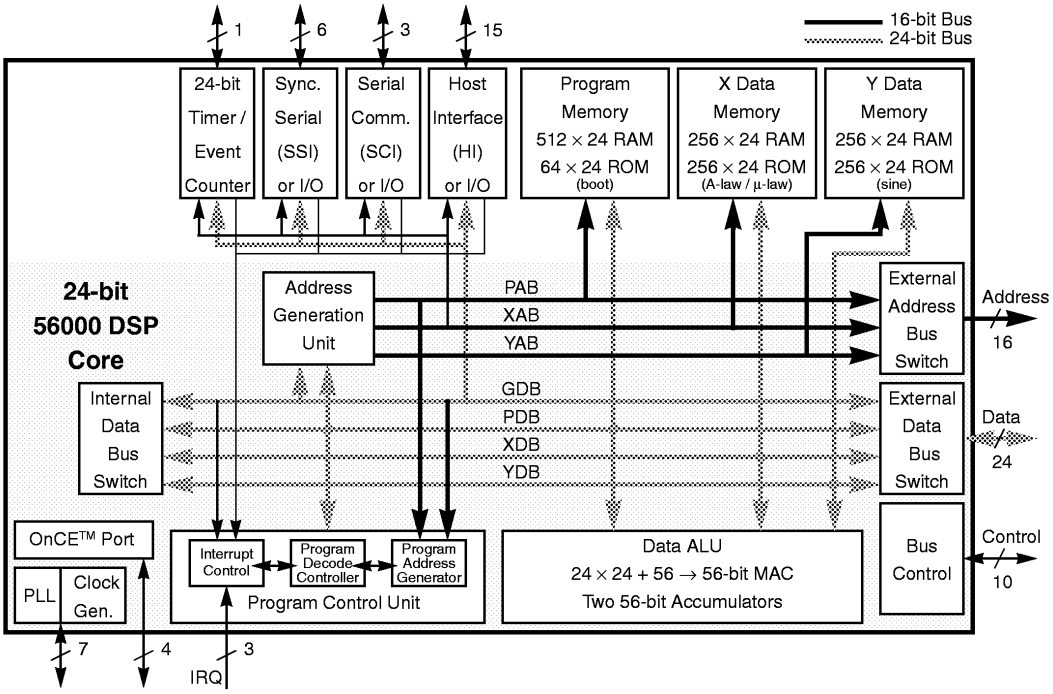
2.3 Motorola 56002

2.3.1 Yleistä

Työssä käytetty 40 MHz Motorolan 56002 kuuluu 56000-prosessoriperheeseen, jossa on useita eri käyttöön tarkoitettuja versioita. Motorola lienee ainut valmistaja jolla on 24-bittinen kokonaisluku-DSP. Täten sen 56000-sarja on saanut suuren suosion audiolaitteita valmistavan teollisuuden keskuudessa. Tämän piirin muunnosta 56004 käytetään yleisesti Dolby Surround Pro Logic -dekooderina ja 56007 ja 56009 -versioita DTS- ja Dolby Digital -dekoodereina.

Tässä sarjassa 56002 on lähinnä yleiskäyttöistä prosessoria, koska siinä on mm. asynkronisen sarjaliitännän RS-232C mahdollistava SCI-yksikkö ja muistiavaruuden laajennusväylän käyttömahdollisuus. 56002 korvaakin jo ikääntyneen 56001-prosessorin.

Prosessorin muistiavaruus on jaettu kolmeen osaan: Ohjelma (P), X-data (X) ja Y-data (Y) -muisti. Monien sisäisten väylien (Harvard-arkkitehtuuri) ansiosta prosessori pystyy osoittamaan kaikkia kolmea muistia samanaikaisesti. Prosessorin lohkokaavio on esitetty kuvassa 3. Piirissä on sisäistä X- ja Y-datamuistia 256 sanaa, sekä ohjelmamuistia 512 sanaa. Käskyjakson pituus on 50 ns 40 MHz:n kellotaajuudella.



Kuva 3. 56002-prosessorin lohkokaavio.

Tiedonsiirto eri väylien välillä tapahtuu käyttäen väyläkytkintä, joka pystyy kytkemään yhteen mitkä tahansa kaksi eri väylää, ilman liukuhihnaviivettä.

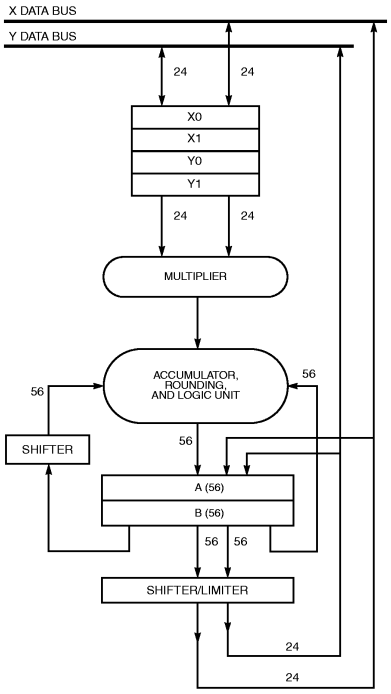
2.3.2 Data-ALU ja MAC-yksikkö

24 bitin sananpituus mahdollistaa 144 dB:n dynaamisen alueen. Tämä on täysin riittävä useimpiin käytännön sovelluksiin, koska useimmat muuntimet ovat 16 bittisiä tai epätarkempia, eivät ainakaan yli 24 bittisiä. Sisäinen 56 bitin laskentatarkkuus ALUn sisällä mahdollistaa 336 dB:n dynamiikan, joten tarkkuuden menetystä välitulosten takia ei tapahdu.

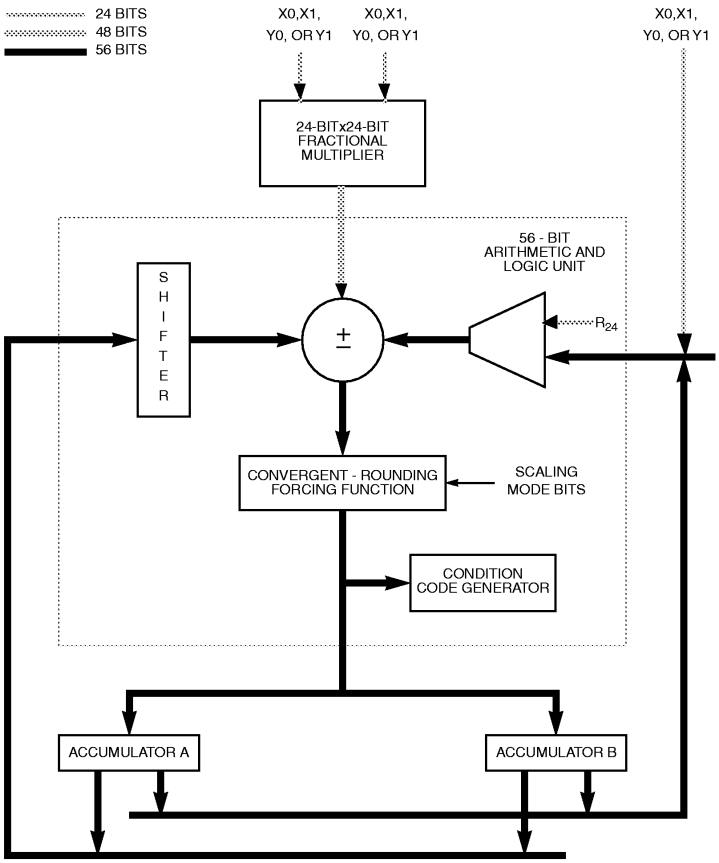
Data-ALU koostuu seuraavista osista:

- neljä 24-bittistä tulorekisteriä (X0,X1,Y0,Y1)
- rinnakkainen yhden jakson liukuhihnaton kerto-yhteenlasku (MAC) yksikkö
- kaksi 48-bittistä akkua (A (A1,A0) ja B (B1,B0))
- kaksi 8-bittistä laajennusrekisteriä (A2,B2)
- akun siirtäjä
- kaksi väyläsiirrintä / rajoitinta

Kerto-yhteenlasku (MAC) -yksikkö on esitetty kuvassa 4. Tämä yksikkö suorittaa kaiken tietojenkäsittelyn prosessorissa.



Kuva 4. Data-ALUn lohkokaavio



Kuva 5. MAC-yksikkö

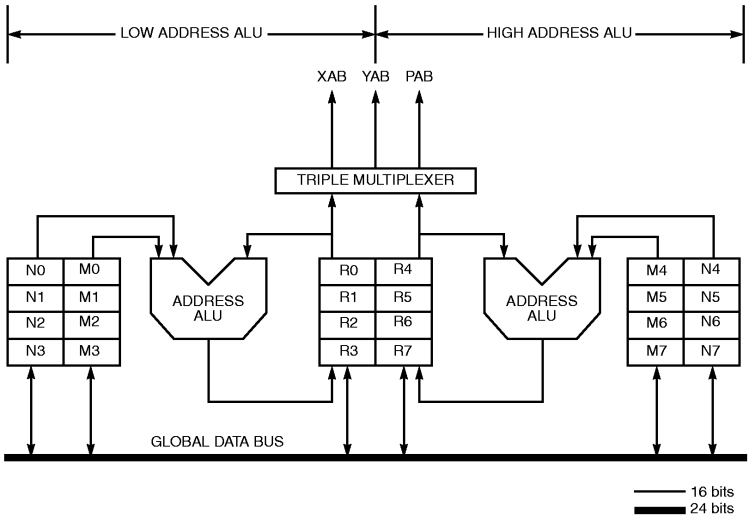
Aritmeettinen yksikkö sisältää kertojapiirin ja kaksi akkua. Kertojan tulo voi tulla ainoastaan X- tai Y-rekistereistä (X0,X1,Y0,Y1). Kertoja kertoo kaksi 24-bittistä lukua keskenään joista saatu 48-bittinen tulos tasataan oikealle ja lisätään 56-bittisen akun entiseen sisältöön. Jos suoritetaan pelkkä kertolaskukäsky (MPY), niin kohdeakun sisältö nollataan ennen kertolaskua.

Loogisessa yksikössä suoritetaan kaikki tavalliset (AND, OR, XOR, NOT) loogiset operaatiot. Nämä operaatiot käyttävät akkujen keskimmäistä rekisteriä (ylempää 24 bitin puoliskoa 48 bitin akusta, rekisteriä A1).

Data-ALU voi tarvittaessa käyttää saturointiaritmetiikkaa, jolloin mahdollisesti ylivuotava arvo rajoitetaan maksimiarvoonsa. Esimerkiksi jos lähdeoperandin arvo olisi 01,100 (+1,5 desimaalisena) ja kohderekisteri olisi 4-bittinen, se saisi arvokseen 1,100 (-1,5 desimaalisena). Tämä on selvästikin virhetilanne, koska ylivuoto on tapahtunut. Onkin toivottavaa kirjoittaa kohteeseen sen maksimiarvo, jolloin esimerkissä kohderekisteri saisi arvon 0,111 (+0,875 desimaalisena) joka on lähempänä arvoa +1,5 kuin –1,5 ja niinpä aiheutunut virhe jää myös pienemmäksi.

2.3.3 Osoitteenmuodostusyksikkö (AGU)

Osoitteenmuodostusyksikön (AGU = Address Generation Unit) rakenne käy ilmi seuraavasta lohkokaaviosta:



Kuva 6. Datamuistin osoitteenmuodostusyksikkö.

Datan osoiterekisterit (R0-R7) on jaettu kahteen ryhmään: Alapuoliseen ryhmään kuuluvat osoiterekisterit R0-R3 ja yläpuoliseen R4-R7. Osoitteenmuodostusyksikkö pystyy kahden osoite-ALUn avulla päivittämään ala- ja yläryhmän osoiterekistereitä samanaikaisesti. Osoiterekistereihin liittyvät läheisesti offset-rekisterit N0-N7 sekä modulorekisterit M0-M7. Offset-rekistereillä voidaan toteuttaa indeksointia ja osoiterekisterin kasvatusta halutulla offsetilla muistiviittausten yhteydessä.

Modulo-rekistereiden M0-M7 avulla voidaan toteuttaa rengaspuskureita, jolloin modulo-rekisterin arvoksi asetetaan puskurin koko-1. Myös FFT-algoritmeissä käytettävä käänteinen bittiosoitus saadaan asettamalla Mn:n arvoksi 0, jolloin carry-bitti etenee käänteisesti. Normaali muistin osoitus saadaan asettamalla Mn:n arvoksi $FFFF.

Osoite-ALU tukee seuraavia osoitusmuotoja:

Epäsuora osoitus	Käyttää Mn re-kisteriä	Sallitut operandit									Assem-blersyn-taksi
		S	C	D	A	P	X	Y	L	XY	
Ei muutosta	E					X	X	X	X	X	(Rn)
Jälkikasvatus 1:llä	K					X	X	X	X	X	(Rn)+
Jälkivähennys 1:llä	K					X	X	X	X	X	(Rn)-
Jälkikasvatus offsetilla Nn	K					X	X	X	X	X	(Rn)+Nn
Jälkivähennys offsetilla Nn	K					X	X	X	X		(Rn)-Nn
Indeksoitu offsetilla Nn	K					X	X	X	X		(Rn+Nn)
Esivähennys 1:llä	K					X	X	X	X		-(Rn)

Taulukko 1. Osoite-ALUn tukemat osoitusmuodot

Huom!
S = prosessorin HW-pino
C = ohjelmakontrollerin rekisteriviittaus
D = data-ALUn rekisteriviittaus
A = osoite-ALUn rekisteriviittaus
P = ohjelmamuistiviittaus
X=X-datamuistin viittaus
Y=Y-datamuistin viittaus
L=L-muistiviittaus
XY=XY-muistiviittaus

2.4 Ohjelmakontrolleri (PCU)

Ohjelmakontrolleri (PCU, Program Control Unit) sisältää kolme päälohkoa: käskydekooderin (PDC, Program Decode Controller), ohjelmaosoitegeneraattorin (PAG, Program Address Generator) ja keskeytysohjaimen (PIC, Program Interrupt Controller).

Ohjelmakontrolleri sisältää kolmitasoisen liukuhihnan käskyn suoritukselle. Käskyn haku, dekoodaus ja suoritus toimivat siten rinnakkain.

Tavallisen DSP-prosessoreista löytyvän käskyn toistokäskyn REP lisäksi ohjelmakontrolleri tukee lohkontoistokäskyä DO, jolla saadaan halutun mittainen lohkosilmukka pyörimään viiveettä, kolmen käskyjakson mittaisen alustuksen jälkeen.

Jotta pinon käsittely olisi tehokasta on prosessorissa kovolla toteutettu 15-tasoinen 32-bittinen pinomuisti paluuosoitteen ja tilarekisterin tallennusta varten.

2.4.1 Piirin sisäinen emulointi (OnCE)

Tämän prosessorin ainutlaatuinen ominaisuus on sen sisäänrakennettu emulointiportti. Portin käyttö ei mitenkään häiritse prosessorin muuta toimintaa, eikä varaa muita resursseja, kuten esim. sarjaliikennepiirin avulla toteutettu debuggeri väistämättä tekisi. OnCE-liitännän kautta päästään tutkimaan prosessorin tilaa millä tahansa hetkellä, ajamaan käskyjä, lukemaan tietoja muistista ja tutkimaan oheislaitteiden tiloja. Näin prosessori voi olla kohdelaitteessa kiinni, eikä tarvita kallista emulaattoria/kaapelointia.

2.4.2 Keskeytykset

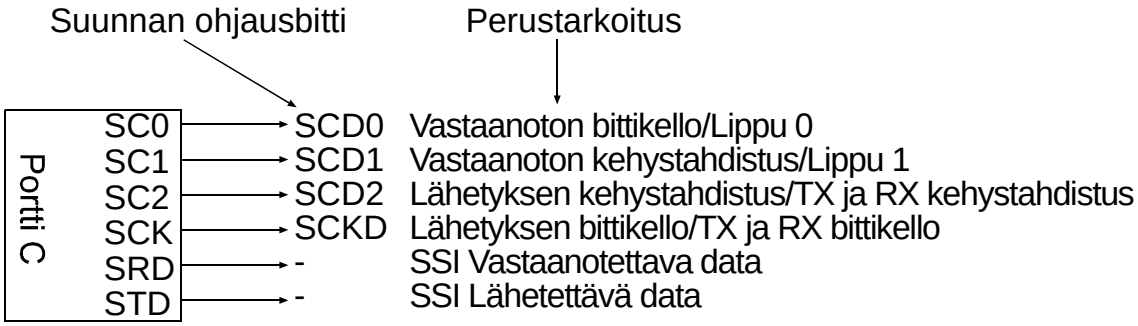
56002:ssa on kahdenlaisia keskeytyksiä: Tavallisia ja nopeita (fast interrupt). Tavallisen keskeytyksen latenssi on 7 konejaksoa ja nopean keskeytyksen 5 konejaksoa. Ulkoisen keskeytyksen aiheuttajia on kolme ulkoista keskeytystä, IRQA, IRQB ja NMI. Näistä IRQA ja IRQB ovat maskattavia keskeytyksiä. Lisäksi prosessorin sisäiset oheislaitteet ja virhetilanteet voivat aiheuttaa keskeytyksen kuten esim. kovopinon ylivuoto tai datan vastaanotto SSI-liitännästä.

2.4.3 SSI-liitäntä

SSI-liitäntää käytetään oheislaitteiden liittämiseen prosessoriin. Sopivia ovat monet A/D- ja D/A muuntimet, koodekit, yms. SSI-yksikkö on hyvin monipuolinen ja siksi osittain vaikeasti ohjelmoitava ainakin aluksi. Liitännällä voidaan lisäksi toteuttaa 32:n prosessorin verkko, jossa jokainen prosessori kuuntelee omaa aikaväliään. Liitäntä on synkroninen sarjamuotoinen liitäntä, jossa on kuusi signaalia:

- STD SSI Transmit Data (Lähetettävä data)
- SRD SSI Receive Data (Vastaanotettava data)
- SCK SSI Serial Clock (Kello)
- SC0 Serial Control 0 (toiminta riippuu SSI:n moodista)

- SC1 Serial Control 1 (toiminta riippuu SSI:n moodista)
- SC2 Serial Control 2 (toiminta riippuu SSI:n moodista)



Kuva 7. SSI-liitännän pinnit ja niiden pääasiallinen käyttötarkoitus.

Seuraavat toiminnot ovat ohjelmoitavissa:

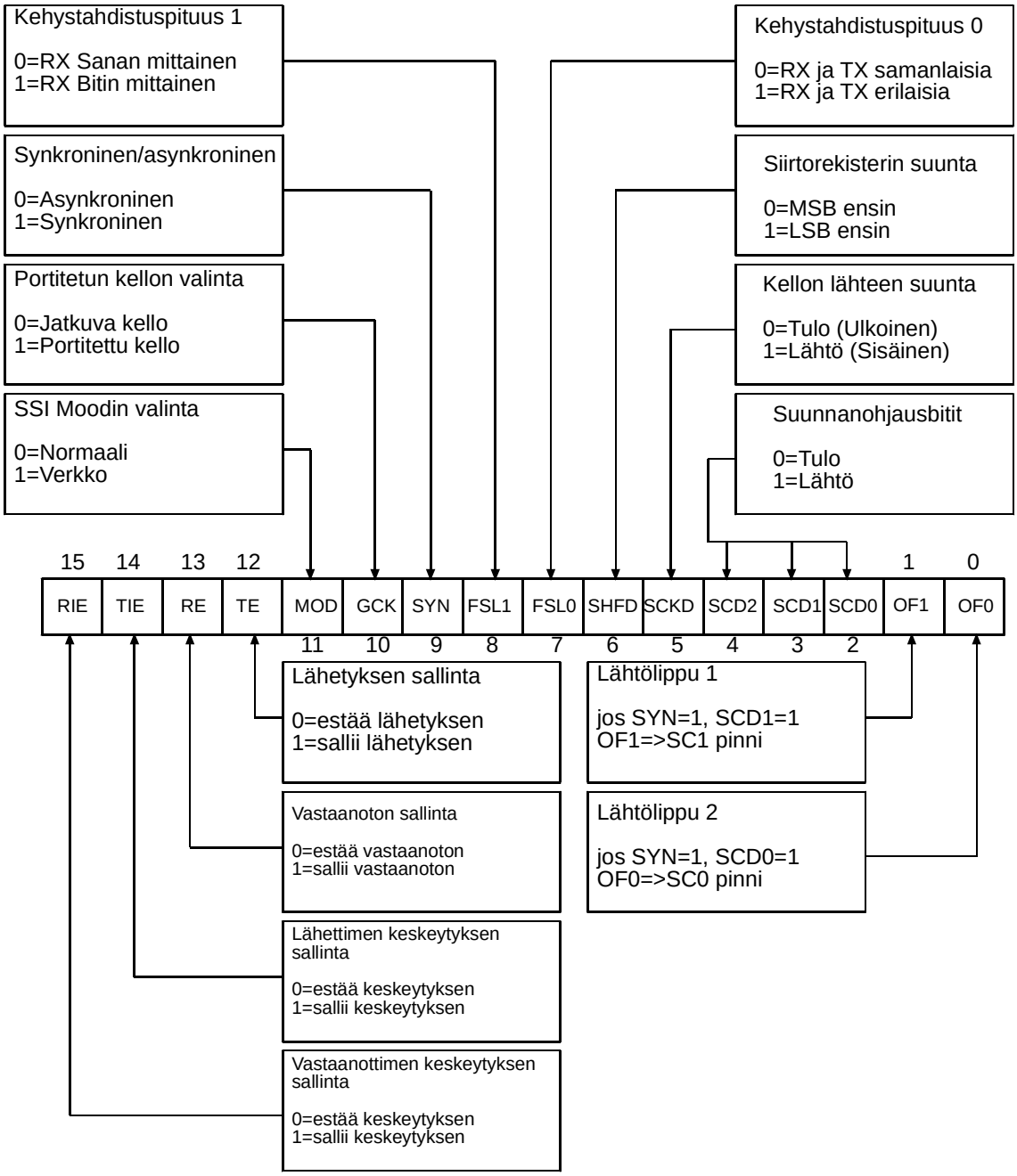
- Kello – Jatkuva, Portitettu, Sisäinen, Ulkoinen
- Tahdistussignaalit – Bitin pituinen tai sanan pituinen
- Lähettimen/vastaanottimen ajoitus – Synkroninen tai asynkroninen
- Toimintatilat – Normaali, Verkko, Tarvittaessa
- Sanan pituudet – 8, 12, 16 tai 24 bittiä
- Bittikello ja sanantahdistussignaalien generointi

SSI-liitäntää ohjataan kahden rekisterin, SSI Control Register A (CRA) ja SSI Control Register B:n (CRB) avulla.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

Ohjausrekisteri A:n biteillä PM7-PM0 (Prescaler Modulus) ja PSR asetetaan kellosignaalin taajuus (jos DSP master). DC4-DC0 biteillä valitaan, montako sanaa/kehys siirretään. Se määrää normaalissa tilassa siis sanatahdistussignaalin taajuuden. WL1-WL0 (Word Length) biteillä valitaan siirrettävien sanojen pituus.

Ohjausrekisteri B ohjaa keskeytyksiä, lähettimen/vastaanottimen sallintaa, kellon lähteen valintaa, siirtorekisterin suuntaa (LSB vai MSB ensin).

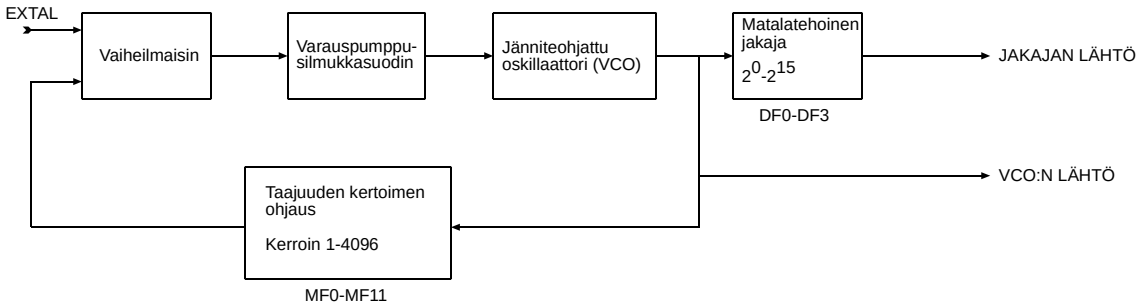


Kuva 8. SSI:n ohjausrekisteri B:n (CRB) kuvaus. Tämä rekisteri ohjaa mm. keskeytyksiä ja siirtorekisterin suuntaa.

2.4.4 PLL-kellogeneraattori

Prosessorin tarvitsema kellotaajuus generoidaan ulkoisesta kiteestä, joka voidaan PLL:n avulla kertoa halutulla kokonaisluvulla väliltä 1-4096. Prototyypissä käytetty kide on taajuudeltaan 6 MHz, josta saatu taajuus kerrotaan PLL:n avulla kahdeksalla jolloin saadaan 48 MHz sisäinen kellotaajuus. PLL mahdollistaa joustavan kellotaajuuden

valinnan, tulotaajuutena voi olla lähes mikä tahansa saatavilla oleva ulkoinen kellotaajuus, lopullinen kellotaajuus voidaan asettaa PLL:n ohjausrekisterin avulla.



Kuva 9. PLL-kellogeneraattorin rakenne.

Lopullinen prosessorin kellotaajuus määräytyy yhtälöstä

$$f_{kello} = \frac{f_{EXTAL} \cdot MF}{DF} = \frac{f_{VCO}}{DF}. \qquad (1)$$

Missä:
DF = Jakajan arvo, määritellään PCTL -rekisterin DF0-DF3 biteillä
f_{EXTAL} = ulkoisen kellon tai oskillaattorin taajuus
f_{VCO} = VCO:n lähtötaajuus
MF = PLL:n kerroin, määritellään PCTL -rekisterin MF0-MF11 biteillä

PLL -yksikön jakajalla voidaan siis pienentää kellotaajuutta hetkessä menettämättä PLL:n vaihelukittua tilaa. Jos DF -kerrointa muutetaan, häviää PLL:n vaihelukitus ja ohjelman suoritus pysähtyy siksi aikaa, kunnes vaihelukittu tila on jälleen saavutettu.

2.4.5 Prosessorin käynnistäminen (Bootstrap-ROM ohjelma)

Kun prosessorin reset-nasta nousee ’1’-tilaan, tutkii prosessori pinnien MODA,MODB ja MODC tilat. Näiden mukaan suorittaa bootstrap-ROM käynnistyksen seuraavasti:

Taulukko 2. Boot-moodit

MODA:MODB:MODC	Selitys
0:0:1	Ohjelman lataus ROM-muistista
1:0:x	Ohjelman lataus Host-liitännästä
1:1:x	Ohjelman lataus SCI-liitännästä

ROM-muistin tapauksessa ladataan 512 ohjelmasanaa sisäiseen ohjelmamuistiin alkaen osoitteesta $C000. Tähän osoitteeseen sijoitetaan 8-bittinen ulkoinen ROM-muisti, johon tallennetaan ohjelma siten, että osoitteessa $C000 on vähiten merkitsevä tavu ensimmäisestä ohjelmamuistipaikasta, $C001 seuraavaksi merkitsevä ja osoitteessa $C002 eniten merkitsevä tavu. Tämän jälkeen tulee seuraava ohjelmasana samalla tavalla jne.

Host-liitännän tapauksessa ladataan sama määrä ohjelmasanoja ohjelmamuistiin osoitteeseen P:$0. Lisäksi lataus voidaan lopettaa asettamalla host-lippu 0 (HF0). Tämä lopettaa latauksen ja käynnistää ohjelman suorituksen osoitteesta P:$0.

SCI-liitännästä bootattaessa annetaan ensin ladattavien ohjelmasanojen määrä ja alkuosoite ohjelmamuistiin. Sen jälkeen syötetään itse ohjelmakoodi. Latauksen jälkeen ohjelmakoodin suoritus käynnistyy siitä osoitteesta, mistä koodin lataus aloitettiin. SCI-liitäntä toimii asynkronisessa tilassa, parametreinä 8 databittiä, 1 stopbitti, ei pariteettia. Kellolähde on ulkoinen ja sen täytyy olla taajuudeltaan 16-kertainen siirtonopeuteen nähden. Jokaisen tavun vastaanottamisen jälkeen, tavu kaiutetaan takaisin SCI:n lähettimellä /2/.

3 MUUNTIMET

3.1 Yleistä

A/D-muuntimia tarvitaan muuntamaan analogiasignaali digitaalimuotoon. Tämä tehtävä on vaativa, sillä muunnosnopeudet (usein luokkaa kymmeniä kilohertsejä) tarkkuuden (16-24 bittiä) huomioon ottaen audiotekniikassa ovat melko suuria. Lisäksi muuntimen täytyy suodattaa ei-toivotut taajuudet mahdollisimman tarkasti pois, jotta laskostumiselta vältyttäisiin. Lisäksi muunninta edeltäviin analogiapiireihin kohdistuu suuria vaatimuksia kohinan suhteen, koska signaalikohinasuhde pitäisi saada arvoon >100dB, mikäli mahdollista.

3.2 A/D-muuntimet

3.2.1 Flash

Flash-tyyppinen muunnin on muunnin jossa on jokaista digitaalista lähtöarvoa kohti oma komparaattori. Näin ollen saadaan muunnosnopeus, joka on ainoastaan riippuvainen komparaattorien ja koodauslogiikan viiveestä.

Flash-muuntimien ongelma on se, että komparaattoreita tarvitaan käytännön muuntimissa hyvin paljon. Esimerkiksi 8-bittisessä muuntimessa tarvitaan 256 kpl komparaattoreita. Tämä rajoittaa tämän tyyppisten muuntimien käytön suhteellisen pieniin sananleveyksiin. Tyypillinen käyttösovellus on videosignaalin A/D-muunnos, jossa tarvitaan jopa usean kymmenen megahertsin muunnosnopeuksia.

3.2.2 Sigma-delta

Sigma-delta-tyyppisessä A/D-muuntimessa muunnos suoritetaan komparaattorin ja integraattorin avulla. Signaaliin lisätään sopiva ns. dither-kohina, jolloin digitaalisesti–suodattamalla komparaattorilta saatavaa 1-bittistä bittivirtaa saadaan haluttu monibittinen lähtö.

3.3 D/A-muuntimet

3.3.1 R-2R

R-2R-tyyppinen D/A-muunnin on rakenteeltaan vastusketju, joita kytketään lähtöön sopiva kombinaatio, ja näin saadaan haluttu lähtövirta. Vastusketju muodostaa kahden potenssien suuruisia virtoja lähtöön, jolloin mikä tahansa virta voidaan esittää käyttämällä sopivaa bittikombinaatiota. Lähtövirrat summataan, ja syötetään virta-jännitemuuntimeen, josta saadaan lähtöjännite.

Tällaisen rakenteen hyvä puoli on pieni kohina sekä hyvä tarkkuus ja pieni särö suurilla antotasoilla. Rakenteen heikkoutena taas on suuret tarkkuusvaatimukset vastusketjussa. Pienikin toleranssi vastuksissa aiheuttaa pienillä lähtöarvoilla epälineaarisuutta, josta aiheutuu säröä lähtösignaaliin.

Käytännössä pienimmät tasot eivät ole lineaarisia, joten säröä aiheutuu väistämättä pienille tasoille.

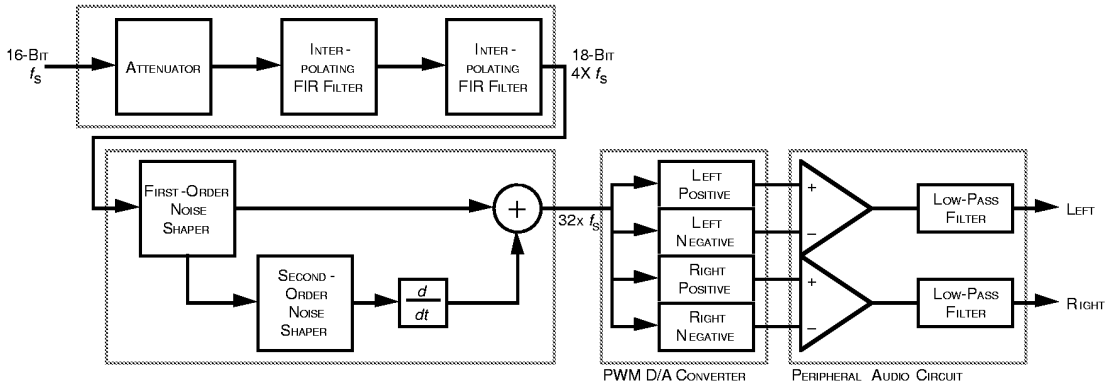
3.3.2 SIGMA-DELTA

Sigma-delta muunnin on komparaattorin ja integraattorin avulla toimiva. Tämä muunnintyyppi on yleistynyt nykyään voimakkaimmin, koska se on valmistusteknisesti helpoin tehdä. Siinä ei ole juuri kriittisiä osia, kuten vastuksiin perustuvissa muuntimissa. Ainut seikka jolle tämä muunnintyyppi on melko herkkä, on tulevan masterkellon jitteri, joka aiheuttaa lähtösignaaliin säröä, tosin tämä alkaa audiokäytössä olla merkittävää vasta hyvin suurilla arvoilla, joten nykyisillä kunnollisten oskillaattoreiden jittereillä sillä ei ole juuri merkitystä. Lisäksi tämän tyyppisellä muuntimella on huonona puolena melko suuri kohina nollasignaalilla. Tämä pyritään minimoimaan mykistämällä muunnin aina, kun siihen on tuotu peräkkäin tietty määrä nollia.

Tämäntyyppisiä muuntimia on monella eri valmistajalla, eri kauppanimillä. Esim. Philips myy omaa piiriään Bitstream- nimellä ja Japanin teollisuusministeriön kehittämä MASH (**M**ulti-st**A**ge noise **SH**aping). MASH-tyyppinen muunnin toimii pulssinleveysmodulaatioperiaatteella (PWM), kun taas Philipsin Bitstream toimii pulssintiheysmodulaatioperiaatteella (PDM).

MASH-tyyppinen muunnin sisältää 4-kertaisella näytteenottotaajuudella toimivan digitaalisuotimen, jota seuraavat ensimmäisen ja toisen kertaluvun kohinanmuokkaimet rinnakkain. Kohinanmuokkainten lähtö syötetään pulssinleveysmodulaattoriin, jonka lähtö alipäästösuodatetaan /6/. MASH-muuntimen lohkokaavio on seuraavassa kuvassa.

Tätä ”yhden bitin” periaatetta sovelletaan myös äskettäin julkistetussa super-audio-CD (SACD)-järjestelmässä. Siinä pulssitaajuus on 2,8224 MHz ja toistokaista jatkuu aina 100 kHz:n asti.



Kuva 10. MASH-muuntimen lohkokaavio

3.4 YLINÄYTTEISTYS (OVERSAMPLING)

Signaalista tehtävä näytteenotto täytyy tehdä Nyquistin näytteenottoteoreeman mukaisesti vähintään kaksinkertaisella taajuudella verrattuna muunnettavan signaalin suurimpaan taajuuteen. Mikäli näytteenottotaajuus on pienempi, tapahtuu ns. laskostuminen, jolloin näytteenottotaajuuden puolikkaan ylittävät signaalit ”heijastuvat” väärälle puolelle spektriä /6/.

Näin ollen ei-toivotut tulosignaalin taajuudet täytyy suodattaa pois ennen A/D-muunnosta. Teoriassa tämä voidaan tehdä äärettömän jyrkällä alipäästösuotimella ennen A/D-muunnosta. Tällainen suodin on vaikea toteuttaa analogiatekniikalla. Lisäksi jyrkillä analogisilla suotimilla on epäedullinen vaihekäyttäytyminen, joka ei ainakaan paranna audiosignaalin laatua.

Nämä ongelmat voidaan ratkaista seuraavasti: signaalista otetaan näytteitä esim. 4-kertaisella näytteenottotaajuudella tarvittavaan lopulliseen signaaliin nähden. Tämä näytejono suodatetaan sitten digitaalisella suotimella haluttuun rajataajuuteen (tässä

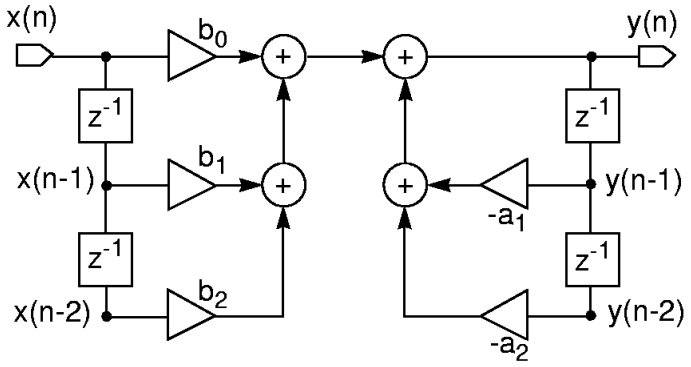
tapauksessa siis yhteen kahdeksasosaan näytteenottotaajuudesta). Tästä suodatetusta signaalista otetaan sitten tässä tapauksessa joka neljäs näyte, jolloin on saatu haluttu lopputulos. Näin tarvittavan analogisen suotimen tarvittavat jyrkkyysvaatimukset jäävät huomattavasti lievemmiksi, jolloin myös vaihekäyttäytyminen on hyvin maltillinen. Tämä ylinäytteistysprosessi sisältyy nykyään moniin audiokäyttöön tehtyihin A/D-muuntimiin.

4 DIGITAALISUOTIMET

4.1 IIR

IIR (Infinite Impulse Response)-tyyppinen suodin on digitaalisten suotimien perustapaus, koska siinä lähtöarvoon vaikuttavat tuloarvojen lisäksi myös lähtöarvot. Kuten nimi kertoo on suotimella äärettömän pitkä impulssivaste. Tämä suodintyyppi on oikeastaan analogisten suotimien digitaalinen vaste, ja IIR-suotimet suunnitellaankin usein analogisten vastaavien s-tason siirtofunktioiden avulla muuntamalla s-siirtofunktiot z-tasoon ns. bilineaarimuunnoksen avulla.

Realisoitaessa IIR-tyyppisiä suotimia, on otettava huomioon mahdollisten pyöristysvirheiden aiheuttamat napojen siirtymät, jotka voivat aiheuttaa ns. limit cycle värähtelyä ja suotimen muuttumista epästabiiliksi. Niinpä usein korkean kertaluvun suotimet pilkotaan useammaksi toisen kertaluvun suotimiksi, jolla tarvittava lukualueen suuruus on pienempi.

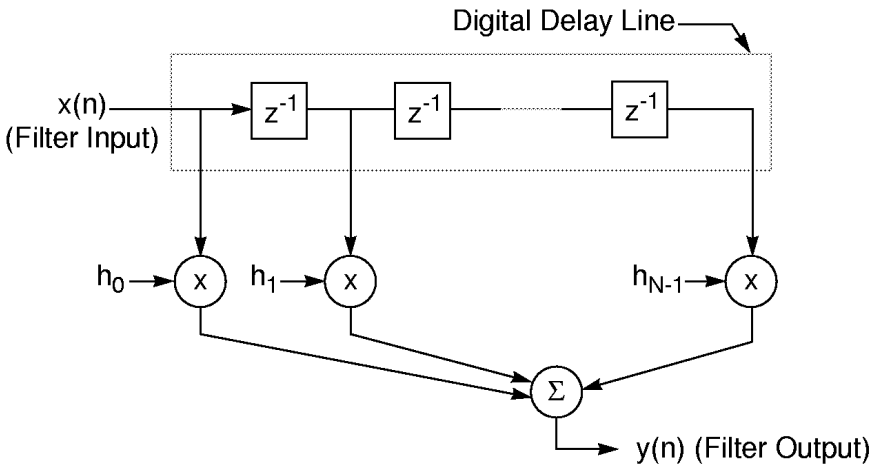


Kuva 11. IIR-Suotimen rakenne. (Suora muoto I)

4.2 FIR

FIR (Finite Impulse Response)-tyyppinen suodin on rakenteeltaan puolikas IIR-tyyppisestä suotimesta. Ainoastaan tulon vanhat arvot vaikuttavat lähtöarvoihin. Suodin on siitä erikoinen, että sillä on ainoastaan nollakohtia, ei yhtään napoja. Tästä seuraa, että suodin on aina taatusti stabiili. Huono puoli on, että saman amplitudivasteen aikaansaamiseksi tarvitaan enemmän laskutoimituksia kuin IIR-tyyppisellä suotimella. Toisaalta FIR-suotimella voidaan toteuttaa vaihelineaarinen suodin, mihin ei päästä

tavallisella IIR-rakenteella. Tämä on tärkeää sovelluksissa jossa signaalin muoto ei saa vääristyä.



Kuva 12. FIR-suotimen rakenne.

FIR-suotimen laskenta tapahtuu siis konvoluutiosumman laskennan avulla kaavalla

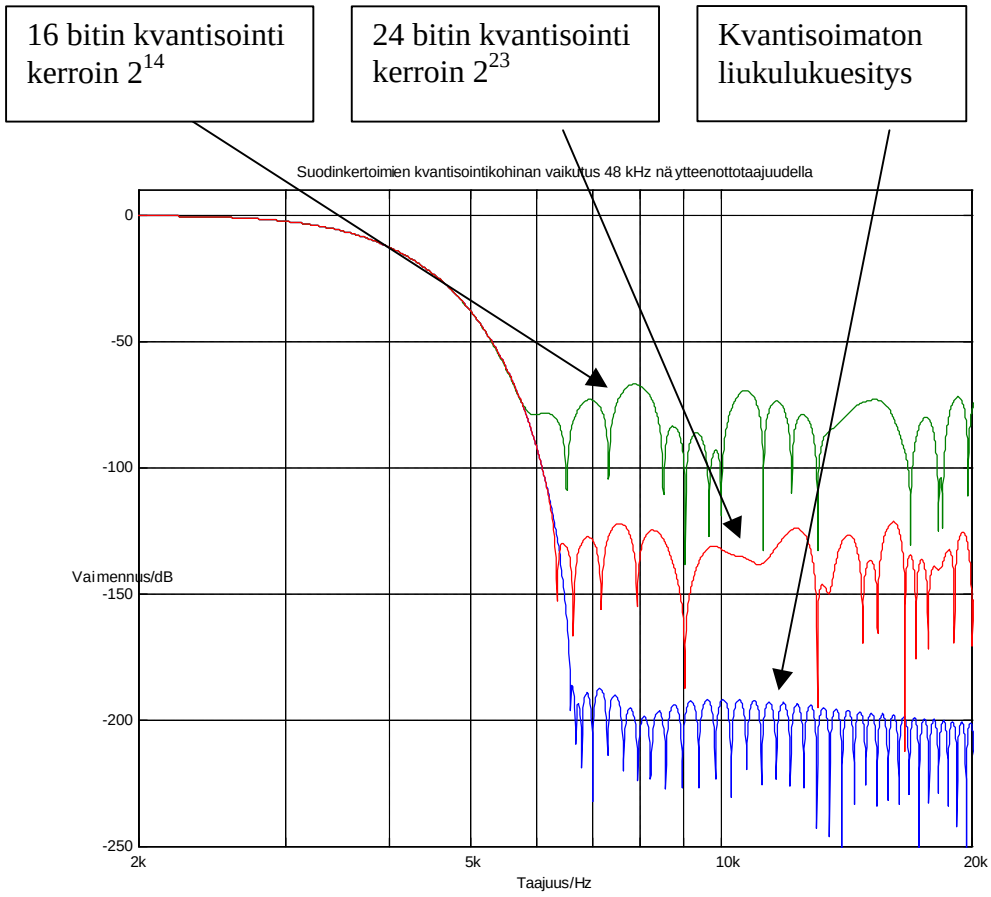
$$y[n] = \sum_{k=0}^{NTAPS-1} x[n-k]h[k]. \qquad (2)$$

Kaavassa (2) NTAPS tarkoittaa suodinkertoimien lukumäärää, eli siis suotimen kertalukua.

4.3 Suodinkertoimien kvantisointivirheen vaikutus FIR-suotimen kohinatasoon

Käsiteltäessä signaalia kokonaislukutyyppisellä signaaliprosessorilla, joudutaan suodinkertoimet pyöristämään lähimpään kokonaislukuun. Tästä aiheutuu väistämättä kohinatason nousua, vaikka alun perin häiriöt olisivatkin olleet pienemmät. Tämä tapahtuu suunnittelemalla suodin, kertomalla luvut sopivalla kvantisointikertoimella, ottamalla saaduista luvuista kokonaisosa ja lopuksi palauttamalla luvut takaisin alkuperäiseen suuruuteensa jakamalla kokonaisluvut samalla kvantisointikertoimella.

Tämän ilmiön vaikutus on esitetty kuvassa 13. Esimerkkisuodin on sadannen kertaluvun FIR-tyyppinen suodin, johon on sovellettu kaiser-ikkunaa β:n arvolla 20 ja taajuusvaste on esitetty bittimäärille 16, 24 ja liukuluku(kvantisoimaton).



Kuva 13. Kvantisointivirheen vaikutus suotimen taajuusvasteeseen

Kvantisointikertoimena on käytetty Texasin 16-bitin signaaliprosessoreille tyypillistä arvoa 2^{14} 16 bitille, tämä siksi, että ko. prosessoreiden akku on ainoastaan 32-bittinen ja näin ollen vuotaa helpommin yli. Motorolan 56002 prosessorilla 24 bitille on käytetty arvoa 2^{23}, koska se on suurin arvo, jolla lukuvälin 1- (-1) luvut mahtuvat 24 bittiin. Tämä ei aiheuta helposti ylivuotoa, koska prosessorin akuissa on 8 bittinen laajennusosa joka sallii akun ylittää lukualueensa 256 kertaa. Tuloksista havaitaan selvästi kvantisoinnin aiheuttama estokaistan vaimennuksen pieneneminen.

Koska audiotekniikassa halutaan yleensä yli 90 dB signaalikohinasuhde, on 24-bitin sanapituuden käyttö perusteltua. 16 bitin tarkkuus riittää paremmin kuin hyvin tietoliikennesovelluksiin, missä data on muutenkin yleensä vain 8-bittistä ja äänenlaadulle ei aseteta läheskään niin suuria vaatimuksia.

Kaikkein suurimpaan tarkkuuteen päästään kuitenkin käyttämällä liukulukuesitystä. Liukulukuprosessorin käyttö tässä työssä olisi kuitenkin nostanut prosessorikortin hinnan huomattavasti korkeammaksi, saavuttamatta kuitenkaan mitään oleellista hyötyä tässä sovelluksessa. Tyypillisiä sovelluksia, jotka vaativat suurta tarkkuutta ovat esim. FFT ja kuvankäsittely.

5 KORTIN RAKENNE

5.1 Rakenne

Kortti rakentuu Motorolan 40 MHz:n signaaliprosessorista DSP56002FC40, Crystalin koodekista CS4225, SRAM-muistipiireistä sekä sekalaisesta logiikkaliimasta. Tarkempi rakenne selviää liitteessä 1 olevasta kytkentäkaaviosta.

5.2 I^2C-väylä

Käytetyn koodekin ohjelmointi tapahtuu I^2C-väylän kautta. Koska 56002 ei sisällä kovolla toteutettua I^2C-väyläliitäntää, toteutin väylän DSP:n B-portin yleiskäyttöisiä pinnejä hyödyntäen. I^2C -väylä on alun perin Philipsin kehittämä synkroninen kaksisuuntainen väylä viihde-elektroniikkalaitteiden piirien väliseen liikennöintiin /4/.

Tiedonsiirto tapahtuu kahdella signaalilla: SDA (data) ja SCL (kello). Jokaisella väylään kytketyllä laitteella on oma osoitteensa, jolla sitä voidaan osoittaa. Väylän tiedonsiirto aloitetaan START-tilalla, jota seuraa osoitettavan piirin osoite ja sen jälkeen varsinaiset siirrettävät tavut.

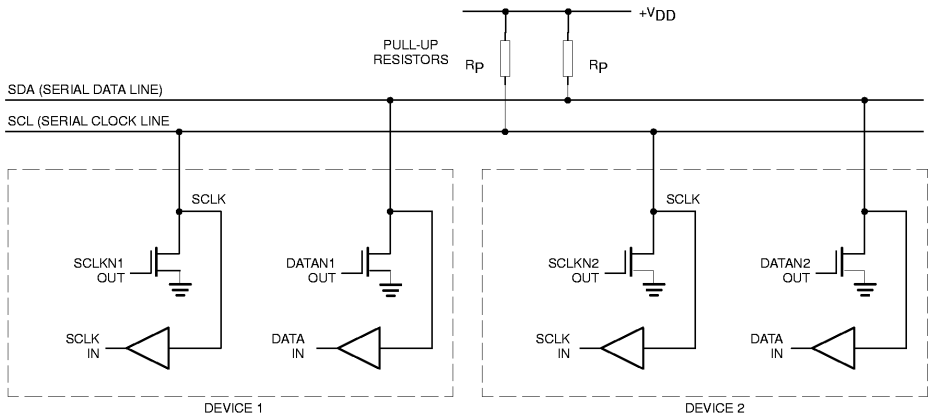
Taulukko 3. I^2C-väyläsanasto.

Termi	Selitys
Lähetin	Laite, joka lähettää tietoa väylälle.
Vastaanotin	Laite, joka vastaanottaa tietoa väylältä.
Isäntä	Laite, joka aloittaa tiedonsiirron, generoi kellosignaalit ja lopettaa tiedon siirron.
Orja	Isännän osoittama laite.
Moni-isäntä	Useampi kuin yksi isäntä voi hallita väylää yhtä aikaa, sotkematta siirrettävää tietoa.
Sovittelu	Menettelytapa, jolla taataan tiedon oikeellisuus, jos monta isäntää yrittää yhtä aikaa hallita väylää.
Synkronointi	Menettelytapa kahden tai useamman laitteen kellosignaalin tahdistamiseen.

Kummatkin väyläsignaalit, SDA ja SCL, on kytketty ylösvetovastuksilla positiiviseen käyttöjännitteeseen. Väylän ollessa vapaana molemmat linjat ovat ylhäällä. Jotta wired-

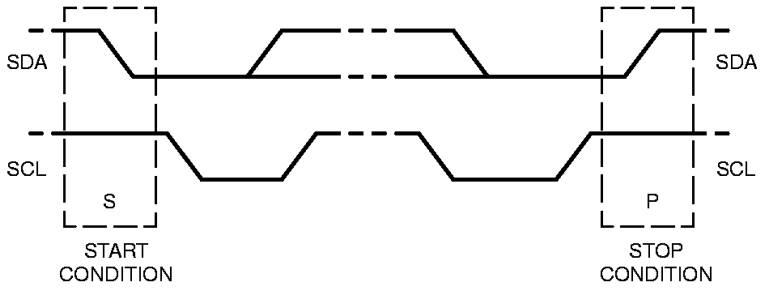
and-toiminto toteutuisi, väylään kytketyillä laitteilla tulee olla avokollektori tai avonielu-tyyppinen anto.

Tiedonsiirtonopeus väylällä on normaalisti 100 kbit/s ja nopeassa tilassa 400 kbit/s. Väylään kytkettävien laitteitten määrää rajoittaa käytännössä ainoastaan maksimikapasitanssi jolle on määritelty maksimiarvo 400 pF.



Kuva 14. Laitteiden kytkeminen I^2C-väylään.

Tiedonsiirron väylällä aloittaa aina väylän master-piiri, lähettämällä start-tilan väylälle. Start- ja stop-tilat ovat ainoat poikkeukset säännöstä, jonka mukaan SDA:n tila ei saa muuttua kun SCL on ylhäällä. Niinpä start on määritelty SDA:n muuttumisena ylhäältä alas, kun SCL on ylhäällä ja stop vastaavasti kun SDA muuttuu alhaalta ylös SCL:n ollessa ylhäällä.

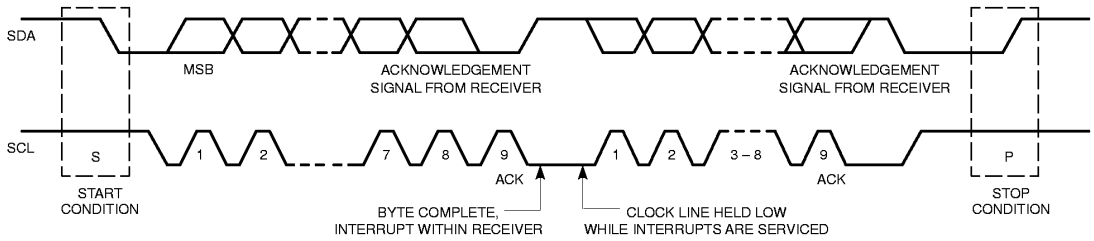


Kuva 15. Start- ja stop-tilojen määrittely.

Väylällä lähetettävä data on 8-bittistä. Bitit lähetetään eniten merkitsevä bitti edellä. Jokaisen siirrettävän tavun jälkeen tulee vastaanottavan laitteen lähettää kuittausbitti ACK, jonka arvo on 0, jos kuittaus on OK, muussa tapauksessa tieto ei ole mennyt perille.

Vastaanotettaessa tietoa orjalähettimeltä viimeistä vastaanotettua tavua ei kuitenkaan kuitata, ja siitä orjalähetin tietää tiedonsiirron päättyvän.

Täydellinen tiedonsiirto on esitetty seuraavassa kuvassa:



Kuva 16. Tiedonsiirto I^2C-väylällä.

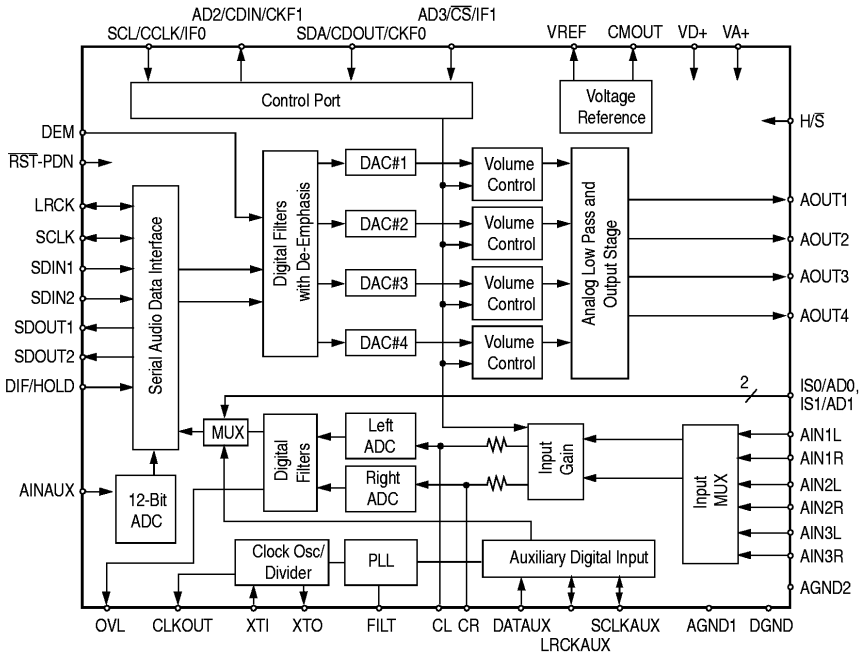
Kortilla I^2C on toteutettu yleiskäyttöisten IO-pinnien avulla porttiin B. Transistorit T1 ja T2 toimivat avokollektorisina väyläohjaimina. Koska tässä sovelluksessa ei tarvita täydellistä moni-isäntä -moodia, on kättelyt hoidettu täysin ohjelmallisesti. Työhön ohjelmoidut I^2C-väylärutiinit tukevat kuitenkin tilannetta, jossa vastaanottava piiri pitää kelloa alhaalla hidastaakseen tiedonsiirtoa. Työssä käytetyt I^2C-rutiinit ovat liitteessä 5.

5.3 CS4225 Monikanavakoodekki

Koodekki on Crystalin (Cirrus Logic Inc:n omistama tuotemerkki) valmistama 16-bittinen neljällä D/A- ja kahdella A/D-muuntimella varustettu. D/A-muuntimien signaali-kohinasuhde on yli 100 dB. Käytettävä näytteenottotaajuus voi olla 4 – 50 kHz. Koodekki sisältää ylinäytteistykseen liittyvän signaalinkäsittelyn ja D/A-muuntimen jälkeisen suotimen. Lisäksi jokaisen D/A-muuntimen lähtösignaalin voimakkuutta voidaan säätää. Piiriin voidaan tuoda myös digitaalimuotoista dataa (esim. S/PDIF vastaanotin-piirin CS8412 kautta). Piirin kellotusmahdollisuudet ovat hyvin monipuoliset. Kellolähteenä voi olla (Fs = näytteenottotaajuus):

- oma kideoskillaattori (256x, 384x tai 512x näytteenottotaajuus) XTI ja XTO pinneissä
- PLL ohjattuna ulkoisesta LRCKAUX -pinnistä Fs -taajuudella
- PLL ohjattuna ulkoisesta LRCK pinnistä Fs -taajuudella
- PLL ohjattuna XTI pinnistä Fs -taajuudella
- PLL ohjattuna SCLK pinnistä 32Fs -taajuudella
- PLL ohjattuna SCLK pinnistä 64Fs -taajuudella
- PLL ohjattuna SCLKAUX pinnistä 32Fs -taajuudella
- PLL ohjattuna SCLKAUX pinnistä 64Fs -taajuudella.

Koodekin rakenne käy tarkemmin ilmi kuvasta 17.



Kuva 17. Koodekin lohkokaavio.

Piiriä voidaan ohjata kahdella tavalla, joko pinnien kautta tai ohjelmallisesti I^2C- tai SPI-väylän kautta. Tässä työssä käytetään I^2C-väyläohjausta. Moodi valitaan H/S-pinnin avulla. I^2C-väylän kautta voidaan ohjata toimintoja seuraavalla rekisteriraken-teella:

Taulukko 4. CS4225 koodekin rekisterirakenne

Rekisteri	Tarkoitus
0	Varattu
1	Ulostulo DAC 1 vaimennus
2	Ulostulo DAC 2 vaimennus
3	Ulostulo DAC 3 vaimennus
4	Ulostulo DAC 4 vaimennus
5	Input 1 vahvistuksen säätö
6	Input 2 vahvistuksen säätö
7	Ulkoisen digitaalitulon ohjaus
8	DSP portin ohjaus
9	Kellon ohjaus
10	Ohjaus
11	Statustavu
12	Tulon valinta
13	Apuäänitulon ohjaus
14	Varattu
15	Varattu

I^2C protokollan mukaisesti lähetetään ensin osoitetavu, jonka bitti 0 kertoo halutaanko piirin rekisteriä lukea vai kirjoittaa. Jos bitti on 1=kyseessä on lukuoperaatio, jos taas 0 niin kirjoitus. Piirin osoite I^2C-väylällä määräytyy pinnien AD0-AD3 tiloista. Osoite on siis seuraavanlainen:

Bitti							
7	6	5	4	3	2	1	0
0	0	1	AD3	AD2	AD1	AD0	R/W

AD3-AD0 pinnit on kytketty piirilevyllä kiinteästi maihin, joten osoite on $20 kirjoitettaessa ja $21 luettaessa. Osoitetavun jälkeen lähetetään MAP (Memory Address Pointer) joka kertoo mitä rekisteriä halutaan lukea/kirjoittaa. Jos halutaan ennen lukua asettaa haluttu rekisteri, täytyy ensin kirjoittaa MAP piiriin, lähettää toistettu start-tila ja osoittaa koodekkia lukuosoitteeseen. MAP tavun rakenne on seuraava:

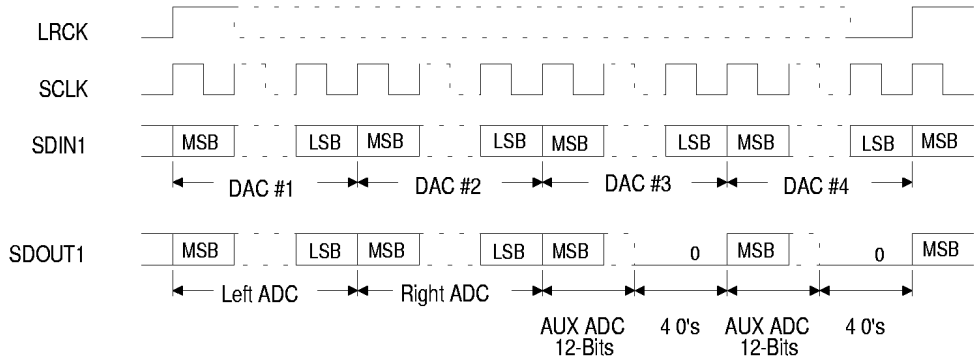
Bitti							
7	6	5	4	3	2	1	0
INCR	0	0	0	MAP3	MAP2	MAP1	MAP0

Jos INCR-bitti on 1, niin kirjoitettaessa/luettaessa peräkkäisiä tavuja osoite kasvaa aina automaattisesti yhdellä mahdollistaen näin kokonaisten rekisterilohkojen luvun/kirjoituksen.

Linjatulot kytketään piiriin AIN-pinnien kautta. Nämä nastat on sisäisesti biasoitu CMOUT-jännitteeseen (tyypillisesti 2,1 V). Kondensaattorin kautta kytketty signaalilähde sallii tulosignaalin olla nollan ympäristössä. Jotta piiriin voitaisiin kytkeä standardi 2Vrms linjatasoinen signaali, se täytyy vaimentaa tulopiirillä. 16-bittisten A/D-muuntimien tulosignaali voidaan valita IS0 ja IS1 -biteillä.

Koodekissa on myös mahdollisuus valita tulosignaalille haluttu vahvistusarvo, 0 - 46,5 dB, 1,5 dB:n askelin. Tämän vahvistuksen arvo vaihtuu häiriöiden välttämiseksi ainoastaan tulosignaalin nollakohtien ylityksien kohdalla. Jos tulosignaalissa ei ole yhtään nollan ylitystä, tapahtuu vahvistuksen vaihto joka tapauksessa 511 näytteen jälkeen.

Liitäntä DSP-prosessoriin voi tapahtua neljällä eri formaatilla, näistä käytetään formaattia, jossa kaikki kanavat saadaan peräkkäin samassa signaalissa, koska SSI-yksikössä on ainoastaan yksi datan tulo ja lähtönasta. Käytetty formaatti näkyy seuraavassa kuvassa.



Kuva 18. Käytettävä dataformaatti

Tämä dataformaatti sopii muuten oikein hyvin SSI-liitännän verkkomoodiin, mutta kehystahdistussignaaliksi LRCK-signaalin ylhäälläoloaika on liian pitkä, se pysyy ylhäällä kahden ensimmäisen sanan ajan. Jotta DSP pystyisi erottamaan nämä kaksi sanaa toisistaan lyhennetään sitä AND-portin IC23A (74HC08), D-kiikun IC22A (74HC74) ja laskurin IC21 (74HC191) avulla. Kytkennän toiminta on seuraavanlainen:

Aina LRCK-signaalin noustessa ylös, asettuu D-kiikun lähtö 1-tilaan. Tämä sallii LRCK-signaalin FSYNC-tuloon prosessorille. Laskuri IC21 laskee bittikellon tahdissa, ja kun se on laskenut 16 bittiä menee sen RC-lähtö (pinni 13) 0-tilaan. Tämä nollaa D-kiikun ja katkaisee LRCK-signaalin lopun. Näin saadaan frame sync-signaali sopivaksi DSP-prosessoria varten. Kytkentä ja sen simulointitulokset on esitetty liitteessä 3.

5.4 Parametrimuisti 24LC16

Parametrimuistina käytetään Microchipin 16 kilobitin EEPROM-muistia, joka liitetään prosessoriin I^2C-väylän kautta. Tämän väylän käyttäminen on järkevää, koska joka tapauksessa tarvittaisiin I^2C-liityntä. Tämä piiri käyttää väyläosoitetta $A0.

5.5 Watchdog MAX1232

Kortille on asennettu vahtikoirapiiri, joka estää ohjelmiston jumittumisen esim. jonkin hetkellisen käyttöjännitehäiriön takia. Piiri hoitaa myös power-on-resetin virrat kytket-

täessä. Tarvittava virkistysväli on tässä valittu 600 ms:ksi. Jos tulosignaalin tila vaihtuu tätä hitaammin, resetoi watchdog prosessorin.

Sallittu jännitteen vaihteluväli on aseteltavissa TOL-pinnin avulla joko 5 tai 10%. Tässä sovelluksessa käytetään 5% toleranssia (4.62V reset-taso). Jos käyttöjännite laskee tämän tason alapuolelle, asettaa watchdog prosessorin reset-tilaan.

Reset-kytkin on myös liitetty watchdog-piiriin, joka poistaa reset-kytkimen aiheuttamat kytkinvärähtelyt.

5.6 Ulkoiset data/ohjelmamuistit

Kortille on suunniteltu asennettavaksi 24*32k tai 24*128k kokoiset SRAM-muistit. Nämä voidaan jumppereilla asetella näkymään prosessorista joko puoliksi jaettuna X/Y-datamuistina tai neljäsosaan jolloin saadaan myös ohjelmamuistiin ulkoista RAMia. Tällöin tosin yksi lohko jää käyttämättä. Tämä olisi saatu poistettua lisäämällä osoite-koodauslogiikkaa, mutta muistin nopeusvaatimusten vuoksi se jätettiin tekemättä, ja hyväksyttiin tällainen muistin ”tuhlaus”.

Ulkoisen muistin liitäntä jota kutsutaan myös portiksi A, sisältää kaksi tehonkulutusta vähentävää ominaisuutta. Sisäistä muistia voidaan lukea siten, että vain välttämättömät ulkoiset signaalit vaihtavat tilaansa, vähentäen näin kytkentävirtoja. Muistinosoituksiin voidaan myös lisätä odotustiloja joka myös vähentää virrankulutusta ulkoista muistia osoitettaessa.

Prosessorista saadaan valmiina /DS (datamuistin valintasignaali), /PS (ohjelmamuistin valintasignaali) sekä X/Y (X/Y datamuistin valinta) –signaalit.

Jumppereilla J14, J15, J17,J18,J19 valitaan minkä kokoinen muisti on käytössä, sekä sen jakaminen eri muistiavaruuksien kesken.

Taulukko 5. 128k SRAM-piirien muistikonfiguraatiot

128k SRAM

J14	J15	J17	J18	J19	P-RAM	X-RAM	Y-RAM
1-2	2-3	1-2	2-3	1-2	32k	32k	32k
2-3	1-2	1-2	2-3	1-2	0	64k	64k

Taulukko 6. 32k SRAM-piirien muistikonfiguraatiot

32k SRAM							
J14	J15	J17	J18	J19	P-RAM	X-RAM	Y-RAM
1-2	2-3	2-3	1-2	2-3	8k	8k	8k
2-3	1-2	1-2	1-2	2-3	0	16k	16k

5.7 Jännitesyöttö

Kytkennän käyttämät analogia- ja digitaaliosioiden käyttöjännitteet reguloidaan kummatkin erikseen omilla lineaarisilla regulaattoreillaan. Näin saadaan minimoitua digitaaliosan aiheuttamat häiriöt analogiapuolelle. Digitaali- ja analogiaosioiden maat on kytketty toisiinsa kuristimen kautta.

5.8 PIIRILEVY

5.8.1 Suunnittelu

Piirilevy suunniteltiin kytkentäkaavion tavoin Protel 98-ohjelmistolla, joka tarjoaa tämän laajuuden työssä välttämättömän vetolistan (netlist) käyttömahdollisuuden. Kytkentäkaavioeditorista saadaan vetolista, joka voidaan ladata piirilevysuunnitteluohjelmistoon. Näin piirilevyohjelmisto tietää, mikä pinni kytkeytyy mihinkin, jolloin vältytään tahattomilta virhekytkennöiltä. Tämä helpottaa testausta, koska ei tarvitse epäillä levyn vetojen virheellisyyttä. Layout ja osasijoittelu on liitteessä 3.

5.8.2 Protovalmistus

Prototyypit valmistettiin ”kotikonstein”, käyttäen valmiiksi lakattua kaksipuolista piirilevyä. Tätä varten piirilevyn molemmat puolet tulostettiin A3-arkille 1,5 kertaiseen kokoon. Nämä tulosteet kuvattiin reprofilmille pienentäen ne alkuperäiseen kokoonsa. Näin tarkkuus saatiin paremmaksi käyttämättä mitään eksoottista tulostinta. Näin saadut reprokalvot kohdistettiin päällekkäin ja toiseen päähän liimattiin piirilevyn suikale, jotta valotettava piirilevy mahtuisi hyvin väliin.

5.8.3 Kustannukset

Kustannukset jakautuivat suunnilleen seuraavasti:

Taulukko 7. Osien ”noin” hinnat

Osa/osakokonaisuus	Noin hinta
DSP-prosessori	100,-
Koodekki	230,-
Piirilevy	80,-
Repro	100,-
128k Muistit	300,-
Piirikannat	70,-
Passiiviset komponentit	100,-
Yhteensä	980,-

6 OHJELMISTO

6.1 Yleistä

Suodinohjelmisto kirjoitettiin kokonaan assembler-kielellä, koska sillä pääsee tarvittaessa optimoimaan nopeutta/koodin kokoa mielin määrin. Apuna ohjelmistokehityksessä käytettiin DSP56002EVM-korttia, jolla testattiin ohjelman osien toimivuutta. Myös käytettävissä oleva tila (512 ohjelmamuistisanaa) asettaa omat rajoituksensa, joten suodinohjelmistoa ei todellakaan voinut tehdä tyyliin ”Microsoft”.

6.2 FIR-suotimen toteutus 56000-assembler-kielellä

Harvard-arkkitehtuurin ansiosta FIR-suodin on yksinkertainen ohjelmoida jopa assemblerilla. Suodin kokonaisuudessaan on seuraava /8/:

```
CLR         A          X0,X:(R0)+   Y:(R4)+,Y0
REP         #NTAPS-1
MAC         X0,Y0,A    X:(R0)+,X0   Y:(R4)+,Y0
MACR        X0,Y0,A    (R0)-
```

Tässä suotimessa vanhojen tuloarvojen rengaspuskuri on asetettu X-muistiavaruuteen ja kertoimet Y-muistiavaruuteen. Valittaessa suotimessa käytettäviä osoiterekistereitä, pitää ne valita siten, että toinen on alemmasta ja toinen ylemmästä osoiterekisteriryhmästä, koska osoite-ALUja on kaksi, yksi molemmille ryhmille. Suotimen tuloarvo on rekisterissä x0 ja lähtöarvo on laskennan päätyttyä akussa A.

On kuitenkin huomattava, että käytettävien osoiterekisterien modulot täytyy asettaa ennen suodatuksen laskemista arvoon NTAPS-1. Tässä siis rekisterit M0 ja M4.

Ensimmäinen käsky nollaa akun A ja samaan aikaan tallennetaan viimeisen näytteen tilalle uusi tuloarvo ja haetaan ensimmäinen suodinkerroin muistista. Sen jälkeen toistetaan seuraavaa käskyä NTAPS-1 kertaa (NTAPS on suotimen kertaluku). Mac-käsky kertoo rekisterit x0 ja y0 keskenään ja summaa tuloksen akkuun. Samaan aikaan haetaan X-muistista aina seuraava näytearvo ja Y-muistista seuraava suodinkerroin. Joka muistiosoituksella AGU suorittaa r0:n ja r4:n kasvatuksen. Kun r0 tai r4 on kasvanut modulo-rekisterillä (m0 ja m4) asetettuun loppuarvoonsa se saa automaattisesti lohkon alkuosoitteen. Lopuksi lasketaan viimeinen tulo ja pyöristys MACR-käskyllä ja pyöräy-

tetään r0 osoittamaan vanhinta näytettä X-muistissa olevaan viivelinjaan. Nyt suotimen lähtöarvo on akussa A.

6.3 Tietoliikenneprotokolla

Korttia voidaan ohjata RS-232-liitännän kautta. Tähän tarkoitukseen on suodinohjelmistoon suunniteltu protokolla, joka käyttää seitsemäntavuisia kiinteänmittaisia paketteja. RS-232 liitännän kautta voidaan vaihtaa suodinkertoimia ja ohjata koodekin toimintoja. RS-232-liitäntä käyttää parametrejä 19200 bit/s, 8 databittiä, 1 stopbitti ja ei pariteettia. Olin aikaisemmin toteuttanut vastaavantyyppisen protokollan 56002EVM-kortille, jossa tosin komennot olivat ASCII:na. Tilanpuute pakotti kuitenkin käyttämään binaarista protokollaa tiedonsiirrossa. Alun perin suunnittelin käyttäväni 38400 bit/s nopeutta, mutta valitun kellotaajuuden takia sitä ei saanut toteutettua riittävällä tarkkuudella (yhteys ei toiminut kaikkien tietokoneiden kanssa). Niinpä pudotin nopeutta puoleen.

Paketti koostuu yhdestä tavusta joka ilmaisee komennon ja kahdesta kolmetavuisesta (24 bittiä) parametristä. 24-bittisten parametrisanojen tavut ovat ’little endian’(intel)-formaatissa eli siis eniten merkitsevä tavu lähetetään ensin, johtuen siitä, että tämä on helpoin käsitellä DSP:n käskykannalla.

Protokollan tahdistus tapahtuu siten, että isäntätietokone lähettää tavua 0x41 niin monta kertaa kunnes kortti vastaa tavulla 0x41 tai riittävä määrä yrityskertoja on tullut täyteen (virhe).

Protokollaa olisi periaatteessa helppo laajentaa myös toiseen suuntaan, jolloin voitaisiin myös lukea kortilta esim. suodinkertoimia, tai muita parametrejä. Työn kuluessa siihen ei kuitenkaan ollut mitään erikoista tarvetta.

Komentopaketin rakenne on esitetty taulukossa 8.

Taulukko 8. Komentopaketin sisältö

Tavu	Merkitys
1	Komentotavu. Katso seuraava taulukko.
2	1. parametrin eniten merkitsevä tavu (MSB)
3	1. parametrin toiseksi eniten merkitsevä tavu (NSB)
4	1. parametrin vähiten merkitsevä tavu (LSB)
5	2. parametrin eniten merkitsevä tavu (MSB)
6	2. parametrin toiseksi eniten merkitsevä tavu (NSB)
7	2. parametrin vähiten merkitsevä tavu (LSB)

Taulukko 9. Protokollan komennot

Komentotavu (hex)	Selitys
0x41	NOP, no operation (ei toimintaa). Käytetään tiedonsiirron tahdistamiseen.
0x42	Aseta alipäästösuotimen kerroin. (1. sana = kertoimen numero, 2.sana = kertoimen arvo).
0x43	Aseta ylipäästösuotimen kerroin. (1. sana = kertoimen numero, 2.sana = kertoimen arvo).
0x44	Aseta koodekin rekisterin sisältö (1. sana = rekisteri, 2. sana = rekisterin arvo).
0x45	Tallenna suodinkertoimet ja koodekin konfigurointi EEPROM-muistiin.
0x46	Lataa suodinkertoimet ja koodekin konfigurointi EEPROM-muistista.

Kortin käyttöä varten on tehty Visual Basic 6.0:lla käyttöliittymä. Tällä voidaan vaihtaa suodinkertoimia sekä ohjata koodekin toimintoja.

6.4 Suodinohjelman toiminta

Suodinohjelman (kts. liite 5) aluksi asetetaan PLL kertoimelle 8 ja odotustiloiksi kaikkiin muistiavaruuksiin 0. Nollataan prosessorin HW-pino-osoitin (SP) ja alustetaan portin B suuntarekisteri. Myös koodekin RESET-signaalia pidetään aktiivisena muutama kymmenen millisekuntia, jotta sen sisäinen resetointilogiikka ehtisi tehdä DC-offsetin kalibroinnin. SSI-liitäntä konfiguroidaan verkkomoodiin, jossa 4 aikaväliä 16-bitin sanoja.

Kun I^2C-liitäntä on alustettu, luetaan tallennetut suodinkertoimet ja koodekin alustusarvot EEPROM-muistista prosessorin datamuistiin. Kortti ei siis käynnisty, ellei koodekin

oikeita alustusarvoja löydy EEPROM-muistista. Tämän jälkeen alustetaan koodekki, ja jäädään suorittamaan pääsilmukkaa.

Pääohjelma odottaa kehyksen vastaanottoa silmukassa. Kehyksen saavuttua kopioidaan vastaanottopuskurista uudet vasemman ja oikean kanavan tuloarvot talteen sekä asetetaan lähetyspuskuriin viimeiset suodatustulokset.

Sen jälkeen lasketaan ali- ja ylipäästösuotimet molemmille kanaville. Kun alipäästötulos on laskettu, viivepuskurin osoite osoittaa jälleen ensimmäiseen näytteeseen puskurissa. Vasta ylipäästösuotimen jälkeen viivepuskurin osoitin asetetaan osoittamaan vanhinta näytettä puskurissa. Ohjelma pitää muistissa ainoastaan yhdet yli- että alipäästösuotimen kertoimet ja oikean ja vasemman kanavan viivepuskurin. Viivepuskuri on toteutettu rengaspuskuriperiaatteella. Uusi näytearvo kirjoitetaan aina vanhimman näytteen päälle. Tällainen järjestely säästää huomattavasti muistia.

Aina ennen uuden kehyksen odottamista, tarkistetaan onko RS-232-liitännästä vastaanotettu uusi komento. Jos on, niin käydään suorittamassa se ennen palaamista suodinsilmukkaan. Tiukan ajoituksen takia RS-232 liitännän käyttö aiheuttaa ääneen pientä ritinää.

Watchdogin virkistys tehdään kahdessa osassa. Ennen suodatusta asetetaan signaali ylätilaan ja suodatuksen jälkeen signaali palautetaan takaisin alas. Näin sen vuoksi, jos normaali suotimen toiminta estyy pitkäksi ajaksi, resetoi watchdog kortin, eikä ohjelma jumiudu ikuisiksi ajoiksi.

Kun RS-232-liitännästä on vastaanotettu 7 tavua, asettaa SCI-liitännän vastaanottokeskeytys lipun, josta pääohjelma tietää komentopaketin olevan odottamassa. Vastaanotetut 7 tavua kootaan `makecmd`-aliohjelmassa kolmeksi sanaksi, joista ensimmäiseen tulee ainoastaan ensimmäinen vastaanotettu tavu (ID), toiseen ja kolmanteen seuraavat 6 tavua. Kaksi viimeistä ovat komennon parametrit (kts. taulukko 8).

RS-232-liitännästä saatujen komentojen suoritus tehdään vertaamalla komennon 1. tavua `cmdtable`-taulukossa oleviin komentoihin. Jos komennon ID täsmää, kutsutaan seuraavan sanan osoittamaa aliohjelmaa ohjelmamuistissa. Komentotaulukko sisältää kaksi sanaa komentoa kohti. Ensimmäinen sana on komennon ID ja toinen sana suoritettavan aliohjelman osoite.

Datamuistin lukua/kirjoitusta varten lasketaan halutun 24-bittisen sanan tavujen osoitteet `form_mem_addr`-aliohjelmalla `DataWRBuffer`-puskuriin. Käytetty EEPROM sisältää ominaisuuden, joka kasvattaa piirin sisäistä osoitetta aina tavun lukutapahtuman jälkeen. Näin luettaessa tarvitaan ainoastaan sanan ensimmäisen tavun osoite, ja EEPROMin sisäinen laskuri hoitaa loput.

Kirjoitettaessa tilanne on hiukka monimutkaisempi. Nyt sisäinen osoitelaskuri toimii ainoastaan 16 tavun lohkoissa, eikä osoite kasva lohkojen rajojen yli. Siispä kirjoitettaessa kirjoitetaan jokainen tavu erikseen. Tätä varten `form_mem_addr` laskee jokaiselle tavulle omat osoitteet. Tilannetta mutkistaa vielä se, että osa osoitteesta sisältyy I^2C-väylän osoitebitteihin. Tallennettava sana pilkotaan kolmeen tavuun jotka sijoitetaan kirjoitusta varten myös `DataWRBuffer`-puskuriin osoitetavujen jatkoksi.

7 KAIUTTIMET

7.1 Yleistä

Käyn tässä läpi lyhyesti eri kaiutinperiaatteet, täydellisyyden vuoksi. Tarkempaa tietoa löytyy esim. Hifi-lehdestä ja akustiikan oppikirjoista asiasta kiinnostuneille.

7.2 Kotelorakenteet

7.2.1 Suljettu kotelo

Suljettu kotelo on nimensä mukaisesti suljettu tila, johon elementti on asennettu. Tämän tilan sisältämä ilma toimii kaiutinelementille jousena, joka vastustaa elementin liikettä. Matalien taajuuksien vaihekäyttäytyminen on suljetulla kotelolla maltillisin tässä käsiteltäviltä kotelotyypeiltä. Taajuusvaste saa vastaavanlaisia muotoja kuin toisen asteen ylipäästösuodin. Alarajataajuuden alapuolella vaste siis laskee 12dB/oktaavi.

7.2.2 Refleksikotelo

Refleksikotelo on helmholz-resonaattori, joka vahvistaa toistoa oman resonanssitaajuutensa ympäristössä. Taajuusvasteeltaan refleksikotelo käyttäytyy kuten 4. asteen ylipäästösuodin. Vasteen lasku on alarajataajuuden alapuolella 24dB/oktaavi.

Ensimmäisenä teoreettisen analyysin elementin käyttäytymisestä refleksikotelossa tekivät N. Thiele ja R. H. Small, joiden mukaan bassoelementtien parametrit on nimetty Thiele-Small-parametreiksi.

Refleksikotelossa viritystaajuuden ympäristössä ääni tulee enimmäkseen putkesta. Tällä taajuudella elementti näkee hyvin pienen akustisen impedanssin, joten kartion liikepoikkeama jää pieneksi. Pieni liikepoikkeama taas merkitsee pientä säröä.

Viritystaajuuden alapuolella alkaa kotelon tuoma vaimennus pienentyä, ja oktaavia viritystaajuutta alempana, toisto refleksiputkesta on vastakkaisvaiheinen elementin ääneen verrattuna. Tämä aiheuttaa toiston nopean vaimenemisen sekä liikepoikkeaman kasvun että mekaanisen tehonsiedon heikkenemisen. Refleksikotelo on mitoitusvirheille huomattavasti herkempi kuin suljettu kotelo.

7.2.3 Kaistanpäästökotelo

Kaistanpäästökotelo on rakenne, jossa elementti on kotelon sisällä ja ääni tulee yleensä yhdestä tai kahdesta putkesta. Rakenne soveltuu hyvin suurien äänenvoimakkuuksien tuottamiseen rajatulla taajuuskaistalla. Rakenteella on huonompi vaihelineaarisuus kuin suljetulla tai refleksikotelolla.

7.2.4 Transmissiolinja

Transmissiolinja on periaatteessa aaltoputkirakenne, jossa alin seisova aalto toistuu putken toisessa päässä voimakkaammin. Periaatteessa putki mitoitetaan halutulla resonanssitaajuudella neljännesaallon mittaiseksi, jolloin saadaan voimakkain vahvistus.

Tämän kotelotyypin ongelmia ovat monimutkainen kotelorakenne sekä hankalasti ennustettava kokonaisvaste, joten transmissiolinjan käyttö on jäänyt vähäiseksi. Lisäksi putki pitää vaimentaa voimakkaasti ei-toivottujen resonanssien välttämiseksi. Tämä valitettavasti vaimentaa transmissiolinjasta saatavaa hyötyä. Ongelmana onkin että kaikki neljännesaallonpituutta vastaavan taajuuden parittomat kerrannaiset aiheuttavat periaatteessa samanlaisen (ei-toivotun) resonanssin.

7.3 Elementtityypit

7.3.1 Dynaaminen elementti

Dynaamisessa elementissä kalvoa liikuttaa voimakkaan magneetin kentässä oleva käämikela. Tässä kelassa kulkeva virta synnyttää voiman F, joka virran suunnasta riippuen siirtää kalvoa ylös tai alaspäin.

7.3.2 Elektrostaattinen elementti

Elektrostaattisessa elementissä on kaksi ns. staattorilevyä joiden väliin varataan yleensä muutaman kilovoltin jännite. Staattorilevyjen välissä on sitten johtavalla aineella pinnoitettu kalvo, johon tuodaan sitten signaalin tahdissa muuttuva jännite, jolloin sähköstaattiset vetovoimat poikkeuttavat kalvoa. Kalvon liike synnyttää kuultavan äänen.

Koska elektrostaattinen kaiutin ei tarvitse jakosuodinta ja ääni synnytetään vain yhdellä elementillä on siinä erittäin hyvä vaihelineaarisuus. Koska ääntä säteilevä pinta on iso, on myös suuntaavuus voimakasta. Tämä vähentää huoneesta johtuvaa äänen värittymistä.

Oikeastaan ainut ongelma sähköstaattisessa elementissä on sen heikko bassotoisto johtuen matalien äänien vaatimasta suuresta tilavuuspoikkeamasta. Tällaisen kaiuttimen säteilykuvio on muodoltaan dipoli, joten kovin lähelle seinää sitä ei voi sijoittaa.

7.3.3 Magnetostaattinen elementti

Magnetostaattinen elementti on analoginen sähköstaattisen elementin kanssa. Tässä kalvoon on tehty ohut liuskajohdin. Kalvo on asetettu voimakkaaseen magneettikenttään, joka poikkeuttaa kalvoa, kun johtimeen johdetaan virta. Muut ominaisuudet vastaavat suunnilleen sähköstaattista elementtiä.

7.4 JAKOSUOTIMISTA

Jakosuotimen tehtävä on jakaa koko äänitaajuuskaista kahteen tai useampaan taajuusalueeseen, jotka sitten toistetaan kukin tarkoitukseensa parhaiten sopivalla elementillä. Jakosuotimissa käytettävillä suotimilla pitäisi olla seuraavat ominaisuudet /7/:

- Käytettävien suotimien vaimennuksen pitäisi olla 6 dB jakotaajuudella
- Suotimien pitäisi olla mahdollisimman jyrkät, tämä vähentää elementtien mahdollisia ei-toivottuja resonansseja toiminta-alueen ulkopuolella, sekä lisää diskanttielementin tehonsietoa.
- Vakio ryhmäviive taajuuden funktiona sekä elementtien vaiheet samat jakotaajuudella.

Näistä vaatimuksista kaksi viimeistä on yleensä hankala ellei mahdotonta toteuttaa käytettäessä analogisia suodattimia.

Vakio ryhmäviive edellyttäisi symmetristä suotimen impulssivastetta, jota on vaikea saada aikaan analogisesti. Muita ongelmia lähinnä passiivisissa jakosuotimissa ovat kelojen kyllästyminen suurilla tehoilla (isoilla kuunteluvoimakkuuksilla) ja toleranssiongelmat (toteutettaessa jyrkkäreunaisia suotimia). Myös elementin impedanssin muuttuminen taajuuden mukana on otettava huomioon jakosuodinta suunnitellessa.

Lisäksi laadukkaat jakosuodinkelat ovat yleensä huomattavan kalliita. Osa passiivisten suotimien ongelmista voidaan ratkaista käyttämällä aktiivista suodatusta. Tällöin päästään eroon keloista, koska ne voidaan simuloida käyttämällä operaatiovahvistinta.

Tällöin tarvitaan jokaiselle elementille oma vahvistin.

Soveltaen DSP-prosessoria aktiivisena jakosuotimena, saadaan samanaikaisesti erittäin hyvä vaihelineaarisuus sekä jyrkkäreunainen suodin. Teoriassa DSP:llä voidaan korjata kaikki kaiuttimen taajuus- että aikavirheet, mutta kuitenkin vain yhdessä pisteessä kerrallaan. DSP:n hyvät ominaisuudet eivät kuitenkaan pelasta kaiutinta siinä tapauksessa, jos siinä on suuntaavuuksiltaan epäyhteensopivat elementit. Jos elementtien suuntakuvioissa on kovin suuria eroja, on kaiuttimen toistaman äänen tasapaino kaikesta huolimatta erilainen. Tämä johtuu siitä, että käytännön kaiutin säteilee ääntä myös muihin suuntiin kuin ainoastaan eteenpäin.

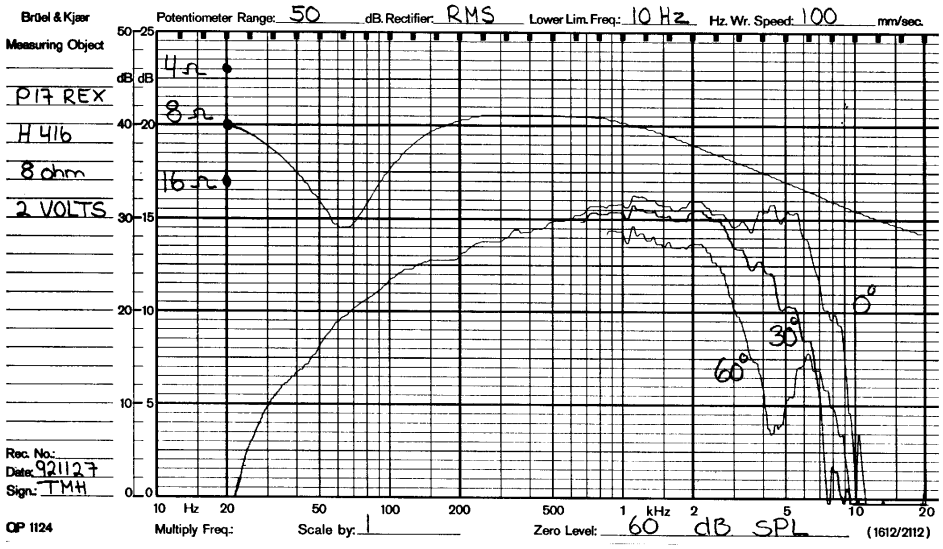
Kun taajuus kasvaa, keskittyy äänen säteily yhä voimakkaammin eteenpäin. Tämä aiheuttaa kokonaisäänenvoimakkuuden laskemisen. Tätä suuntaavuutta voidaan mitata ns. kaiuntahuoneessa mittaamalla tehovaste. Se kuvaa kaiuttimen kaikkiin suuntiin säteilemän äänen summavoimakkuutta eri taajuuksilla. Tehovasteen toivottava muoto on loivasti laskeva suora, jossa ei ole jyrkkiä muutoskohtia.

Jos siis suuntaavuus muuttuu voimakkaasti jakotaajuudella, kaiutin ei voi soida hyvin. Siispä kaiutin täytyy suunnitella hyvin myös perinteisessä akustisessa mielessä.

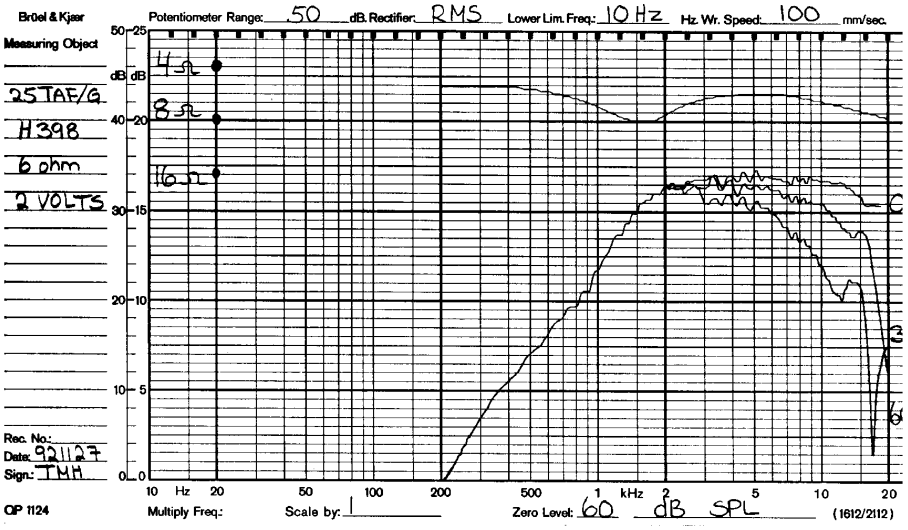
8 KOEKAIUTTIMET

8.1 Yleistä

Suunniteltua suodinta piti tietenkin koekuunnella. Esitän seuraavaksi kuvauksen käytetyistä kaiuttimista, jälleen täydellisyyden vuoksi. Koekaiuttimet rakennettiin 12 litran refleksikoteloon, joka viritettiin 40 Hz:n taajuudelle. Bassoelementiksi valittiin Seasin P17REX-elementti ja diskantiksi Seasin 25TAF/G. Tehtaan mittaamat elementtien vasteet on esitetty seuraavissa kuvissa.



Kuva 19. Bassoelementin Seas P17REX vaste



Kuva 20. Diskanttielementin 25TAF/G vaste

8.2 Jakosuodinsuunnittelusta

Bassoelementin vasteesta /3/ näkyy, että elementin vaste nousee n. 10 dB välillä 100 Hz - 1kHz, joten tämä nousu täytyy kompensoida basson suotimessa vaimentamalla vastaavasti suotimen vastetta. Jos tätä ei tehdä, kuulostaa kaiutin liian keskialuevoittoiselta, lisäksi bassot jäävät turhan vaimeiksi. Vasteet on mitattu vakiojännitteellä kaiuttomassa huoneessa. Bassoelementti on 12 litran suljetussa kotelossa.

Digitaalisuotimien suunnitteluohjelmistona käytettiin MathWorksin MATLABia, jossa on erittäin hyvät työkalut FIR- ja IIR-tyyppisten suotimien suunnittelemiseksi. Erityisen hyvin jakosuotimen suunnitteluun soveltuu fir1-funktio, johon annettava rajataajuus on samalla –6 dB vaimennusta vastaava taajuus. Näin ollen funktiolle voidaan antaa parametriksi suoraan haluttu jakotaajuus ali- että ylipäästöosalle ja kokonaisvaste on suora. Tässä työssä käytetään FIR-tyyppistä suodinta, koska pyritään mahdollisimman korkealuokkaiseen suodatukseen, myös aikatasolla. Seuraavaksi esittelen lyhyen esimerkin, jossa on selostettu lyhyesti suotimen suunnittelun vaiheet.

Ylipäästösuodin voidaan laskea alipäästösuotimen impulssivasteesta seuraavasti: Ensin otetaan lähtökohdaksi alipäästösuotimen impulssivaste, joka invertoidaan. Tämän jälkeen lasketaan alkuperäisestä alipäästösuotimen impulssivasteesta DC-vaste (summataan kaikki kertoimet yhteen). Vähennetään keskimmäisen kertoimen arvo äsken lasketusta DC-vasteesta ja sijoitetaan saatu arvo keskimmäisen suodinkertoimen arvoksi. Nyt saatu ylipäästösuotimen impulssivaste ja alipäästösuodin muodostavat täydellisen parin, jonka summattu taajuusvaste on täysin suora.

Tällaisen ylipäästösuotimen suunnittelumenetelmän voi myös johtaa siten, että kuvitellaan kaksi fir-tyyppistä suodinta kytketyksi lähdöstään summaimeen. Alustetaan vanhat tuloarvot nolliksi ja syötetään sisään molempiin suotimiin impulssi. Nyt halutaan, että summaimen lähdöstä täytyy myös saada kaikilla muilla hetkillä 0, paitsi keskimmäisen tapin kohdalla 1. Näin siis saadaan kokonaisimpulssivasteeksi sama impulssi, joka suotimeen syötettiin, paitsi viivästyneenä puolen suotimen pituuden verran.

Kenties paras ikkunointifunktio lienee kaiser-ikkuna, jonka beta-parametrilla voidaan tehokkaasti säädellä estokaistan vaimennusta. Jos tiedetään haluttu estokaistan vaimennus desibeleinä, betan arvo saadaan kaavasta

$$b = 0{,}1102(vaim - 8{,}7). \qquad (3)$$

Jakotaajuudeksi elementtien akustisten vasteiden perusteella päätettiin ottaa 3,5 kHz. Suotimen kertaluku valittiin käytettävissä olevan muistitilan huomioiden 63:ksi. Tämän jälkeen tarvittavat suotimet laskettiin Matlabilla.

3,5 kHz alipäästösuotimen kertoimet saadaan helposti Matlabilla komennolla

```
h_lp = fir1(63-1,3500*2/48000,kaiser(63,10));
```

Jostakin kumman syystä matlabin signal processing toolboxin fir1-funktion parametrinä annettava suotimen pituus täytyy olla yhtä pienempi kuin haluttu suotimen pituus, mutta ikkunointifunktiota ilmoitettaessa se täytyy olla sama kuin suotimen todellinen pituuskin. Tämän jälkeen käännetään impulssivaste ja nollataan kertoimien summa, josta saadaan siis täydellinen pari alipäästön impulssivasteelle. Tähän ei matlabissa ole valmista funktiota, mutta onneksi sellaisen ohjelmointi ei ole kovin vaikeaa. Ylipäästösuotimen impulssivaste saadaan siis funktion mkhigh()-avulla. Tämän funktion listaus on liitteessä 7.

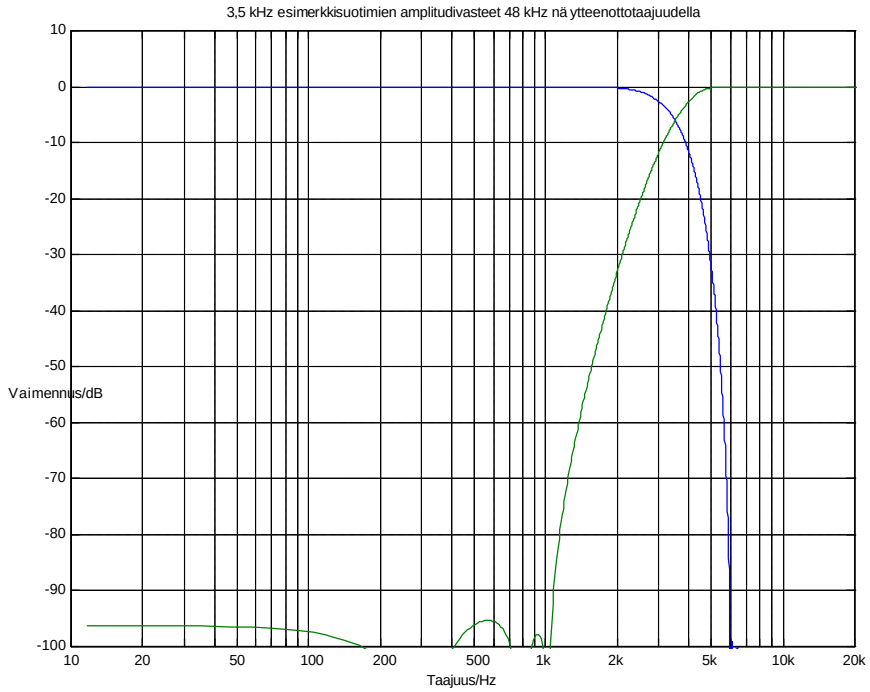
```
h_hp=mkhigh(h_lp);
```

Tämän jälkeen voidaan kertoimet (h_lp sisältää siis alipäästön impulssivasteen ja h_hp ylipäästön impulssivasteen) listata tiedostoon kortille lataamista varten. Suotimen latausohjelma lukee puhtaita ASCII-tiedostoja, joissa on yksi kerroin aina rivillään desimaalisena.

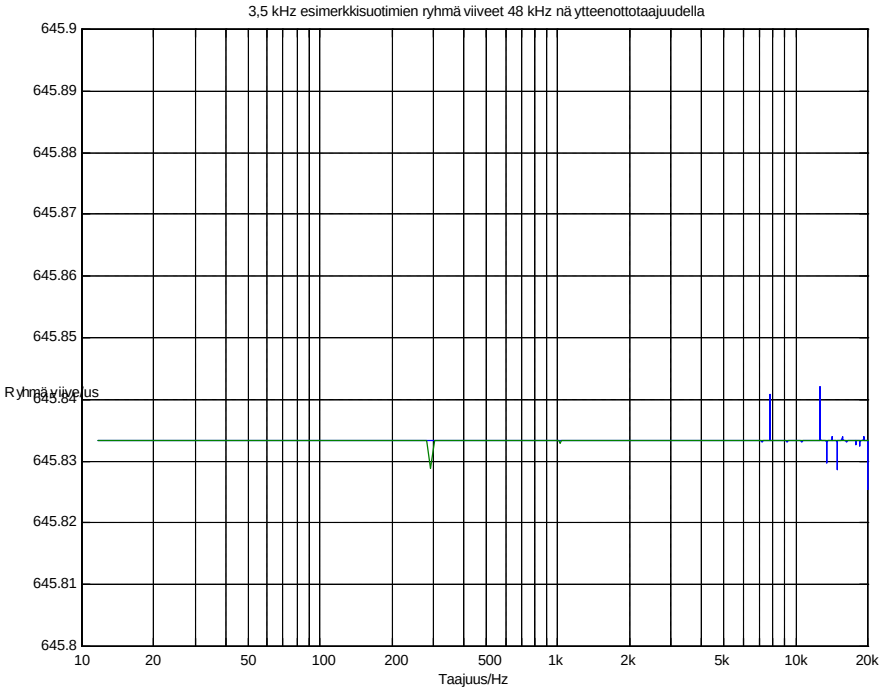
Tässä yhteydessä on huomattava, että kortilla oleva suodinohjelma ei millään tavalla kohtele yli- tai alipäästökanavaa eri tavalla. Tästä seuraa, että molempiin suotimiin voidaan ladata toistensa kertoimet, joka tietenkin aiheuttaa yli- ja alipäästökanavien vaihtumisen keskenään.

Myös signaalin vaiheenkääntö on mahdollista kääntämällä impulssivasteen vaihe.

Laskettujen suotimien taajuus- ja impulssivasteet on esitetty seuraavissa kuvissa.



Kuva 21. Esimerkkisuotimien amplitudivasteet



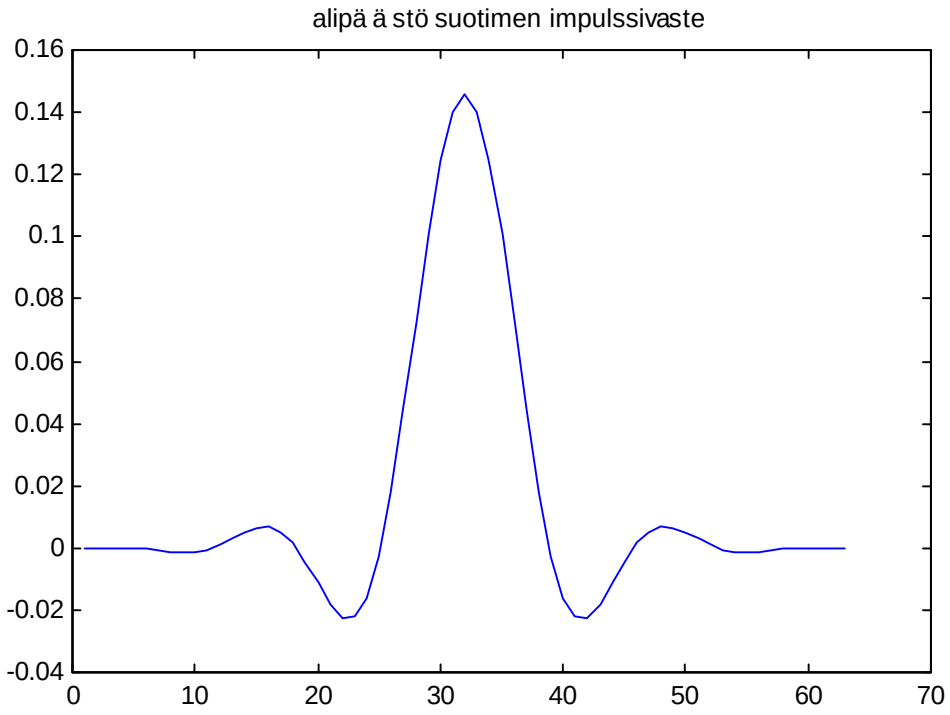
Kuva 22. Esimerkkisuotimien ryhmäviiveet. Ryhmäviive on täysin tasainen lukuun ottamatta laskennan epätarkkuutta, joka näkyy suotimen estokaistalla.

Taajuusvasteesta kuvassa 21 voi hieman arvioida, kuinka jyrkkiä lasketut suotimet ovat. Suotimet vaimentavat noin 75 dB oktaaville. Vastaavaan analogisen suotimen vaimennukseen tarvittaisiin 75 dB/6 dB/kertaluku = 13. kertaluvun analoginen suodatin. Vastaavassa analogisessa suodattimessa ei kuitenkaan päästä lähellekään digitaalisen FIR-suotimen erinomaista vaihelineaarisuutta. Yleisesti pätee analogisiin suotimiin se tosiseikka, että suotimen aikatason ominaisuudet huononevat mitä paremmaksi taajuustason ominaisuudet tulevat. Aivan ongelmaton ei digitaalinen suodinkaan ole, matalien rajataajuuksien realisointi on nimittäin suhteellisen hankalaa, impulssivasteen pitenemisen vuoksi.

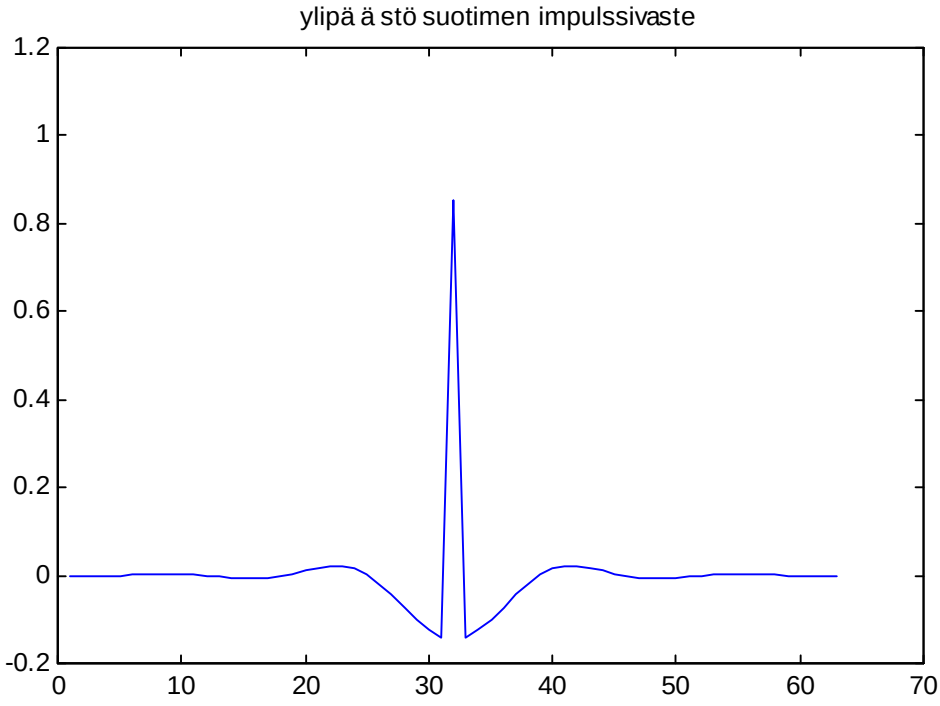
Ryhmäviive kuvassa 22 on laskentatarkkuus huomioon ottaen täysin tasainen. Matemaattisesti on mahdollista osoittaa, että symmetrisen impulssivasteen omaavan digitaalisuotimen siirtofunktio on mahdollista saattaa muotoon, jossa on sekä kompleksinen viive- että reaalinen amplitudivasteen määräävä osa. Tämä viive riippuu ainoastaan suotimen pituudesta ja on tarkalleen puolet suotimen pituudesta. Näin voidaan siis todeta, että FIR-suodin on täysin vaihelineaarinen. Matlab laskee vaiheen ja sen perusteella ryhmäviiveen. Tässä on kuitenkin se vaikeus, että jos kompleksisen taajuusvasteen itseisarvo sattuu olemaan hyvin lähellä nollaa tai peräti tasan 0, kyseisen taajuuden vaihetta ei voida päätellä ja niinpä siitä seuraa ryhmäviivekuvaajaan ”ryppyjä”.

Suotimien impulssivasteet ovat aivan odotetun kaltaiset. Alipäästösuotimen impulssivaste on sinc-funktion pätkä, tosin ikkunoituna kaiser-ikkunalla.

Lopuksi lienee syytä huomauttaa, että mitä matalammalle jakotaajuus asetetaan, sitä huonommin fir1- tai fir2-funktiot suunnittelevat ylipäästösuotimen. Ei siis kannata käyttää ko. funktioita ylipäästösuotimen suunnitteluun. Sen sijaan kannattaa ylipäästösuodin tehdä alipäästösuotimen impulssivasteesta edellä esitetyllä tavalla.



Kuva 23. Alipäästösuotimen impulssivaste.



Kuva 24. Ylipäästösuotimen impulssivaste

9 EMC-mittaukset

EMC-mittaukset suoritettiin Pohjois-Savon Ammattikorkeakoulun elektroniikan tuotekehityslaboratoriossa. Rohde&Schwarzin EMC-mittapaikalla (TEM-kammio). Nämä mittaukset tehtiin lähinnä mielenkiinnosta, koska kaksikerroksisella levyllä ei juuri ole mahdollisuuksia optimoida layoutia EMC:n suhteen. Mittaukset suoritettiin paljaalle piirilevylle, joten erikoisen hyviä tuloksia ei siksikään ollut odotettavissa.

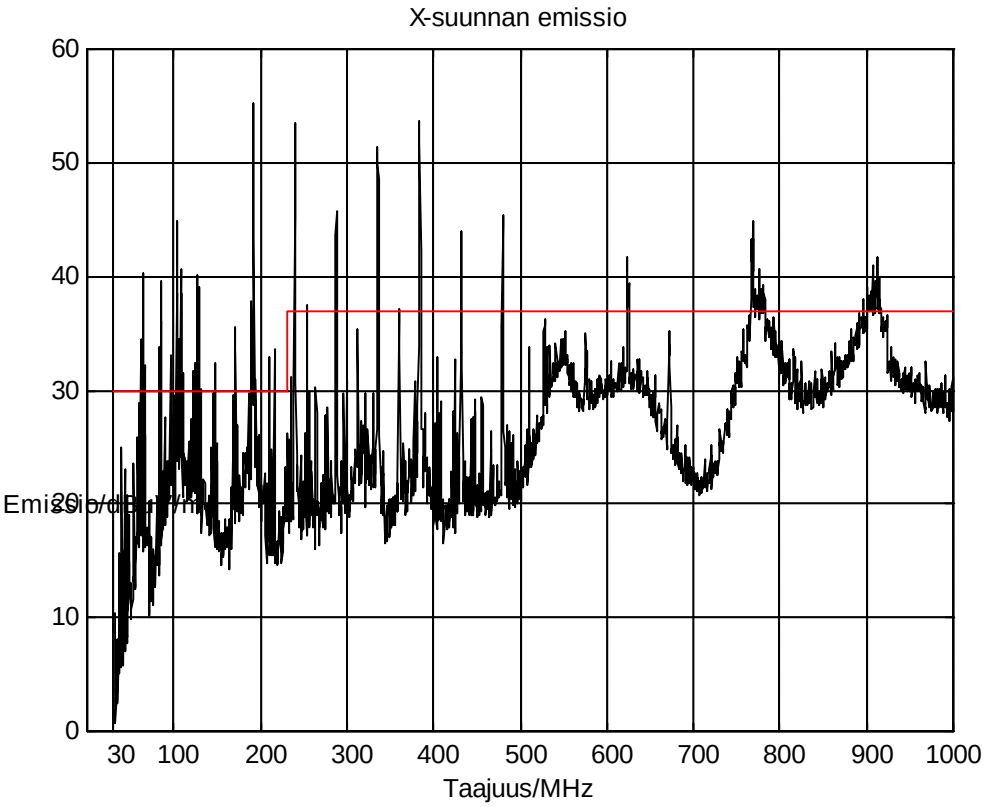
EMC-vastaanotin on periaatteessa aivan tavallinen radiovastaanotin, jossa on tietynlainen standardissa määritelty ilmaisin, sekä välitaajuuskaistanleveys 120kHz (-6dB).

Suunniteltu laite kuuluisi periaatteessa kotilaitekategoriaan (Luokka B), joten sitä koskee standardin EN55022 mukainen tiukempi 30/37 dBuV:n raja (koti-, toimisto- ja kevytteollisuuslaitteet). Asianmukainen metallikoteloon sulkeminen vähentäisi emissioita huomattavasti.

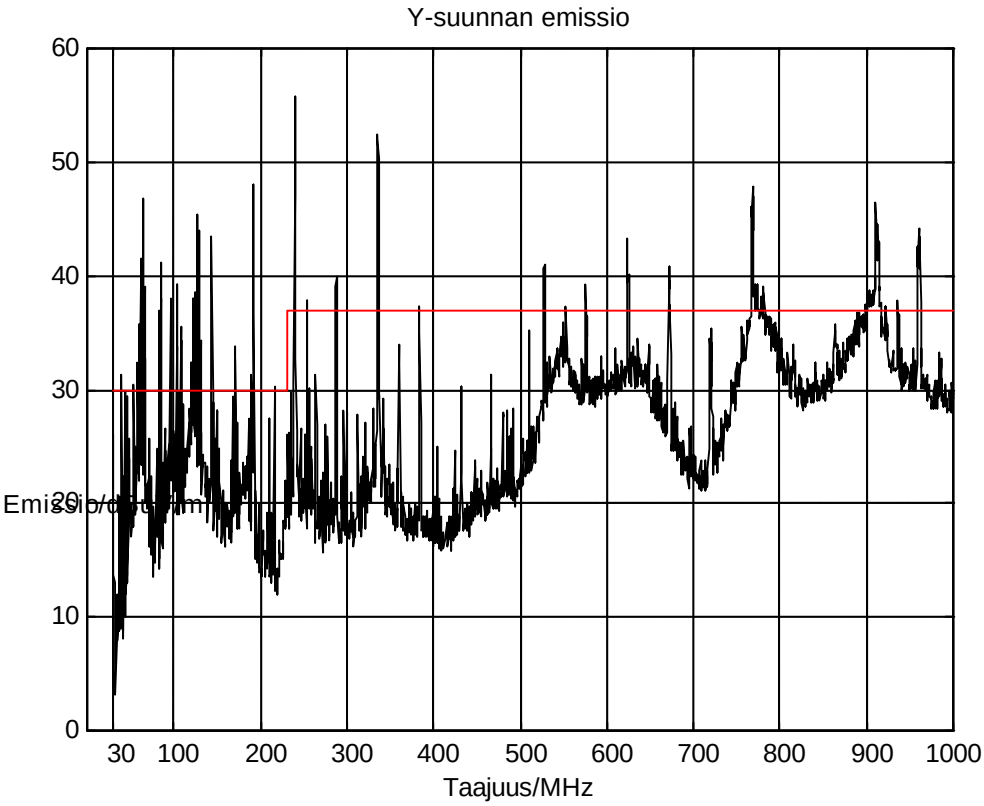
Mittaustuloksiin on piirretty rajaviiva osoittamaan standardin vaatimusta.

Taulukko 10. EN55022 1987 EMC-standardin mukaiset emission raja-arvot.

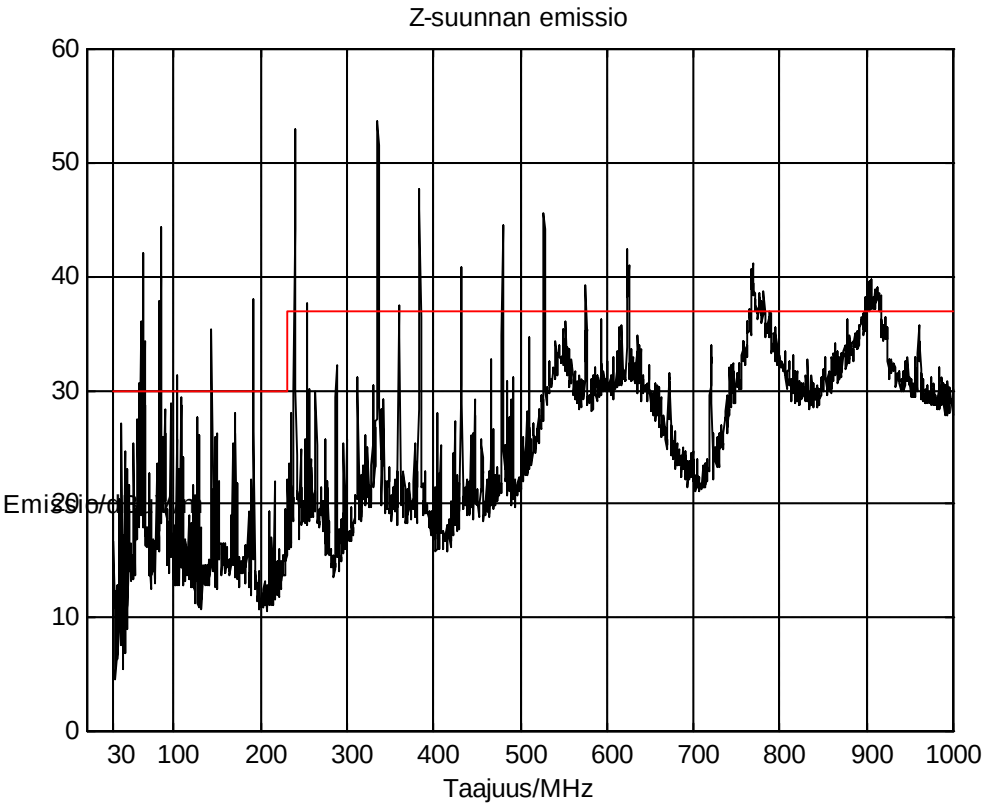
Taajuus/MHz	Testausetäisyys/m		Kvasi-huippuarvoilmaistun signaalin raja-arvo dBµV/m		
	Luokka A	Luokka B	Luokka A	Luokka A@10m	Luokka B
30-230	30	10	30	39,5	30
230-1000	30	10	37	46,5	37



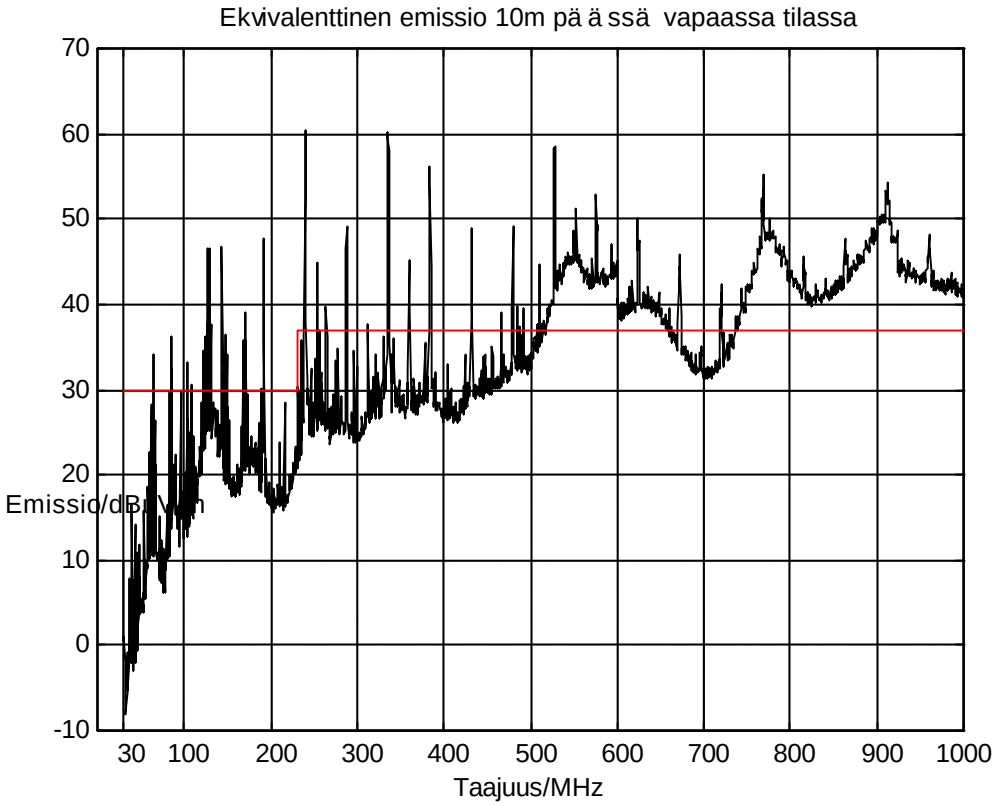
Kuva 25. Kortin X-suuntainen emissio



Kuva 26. Kortin Y-suuntaiset emissiot



Kuva 27. Kortin Z-suuntaiset emissiot



Kuva 28. Ekvivalenttinen kokonaisemissio 10m päässä

Kortin ekvivalenttinen emissio kuvassa 28 on laskettu kolmen mittauksen perusteella.

Voimakkaimmat kortin säteilemät häiriöt keskittyvät 200-400MHz välille. Tämä johtunee käyttöjännitevedoissa olevien induktanssien sekä hajakapasitanssien välisistä resonansseista. Koska kellotaajuus on 48MHz, pitäisi voimakkaimman signaalin näkyä ko. taajuuden ympäristössä. Kuitenkin ko. taajuusalueen emissiot ovat pienemmät kuin ylempien taajuusalueiden emissiot, joten niiden täytyy aiheutua em. resonansseista. Myös prosessorin käyttöjännitteen erottaminen omaksi ”saarekkeekseen” kondensaattorien ja kelojen avulla saattaisi tuoda ainakin osittaista parannusta tilanteeseen.

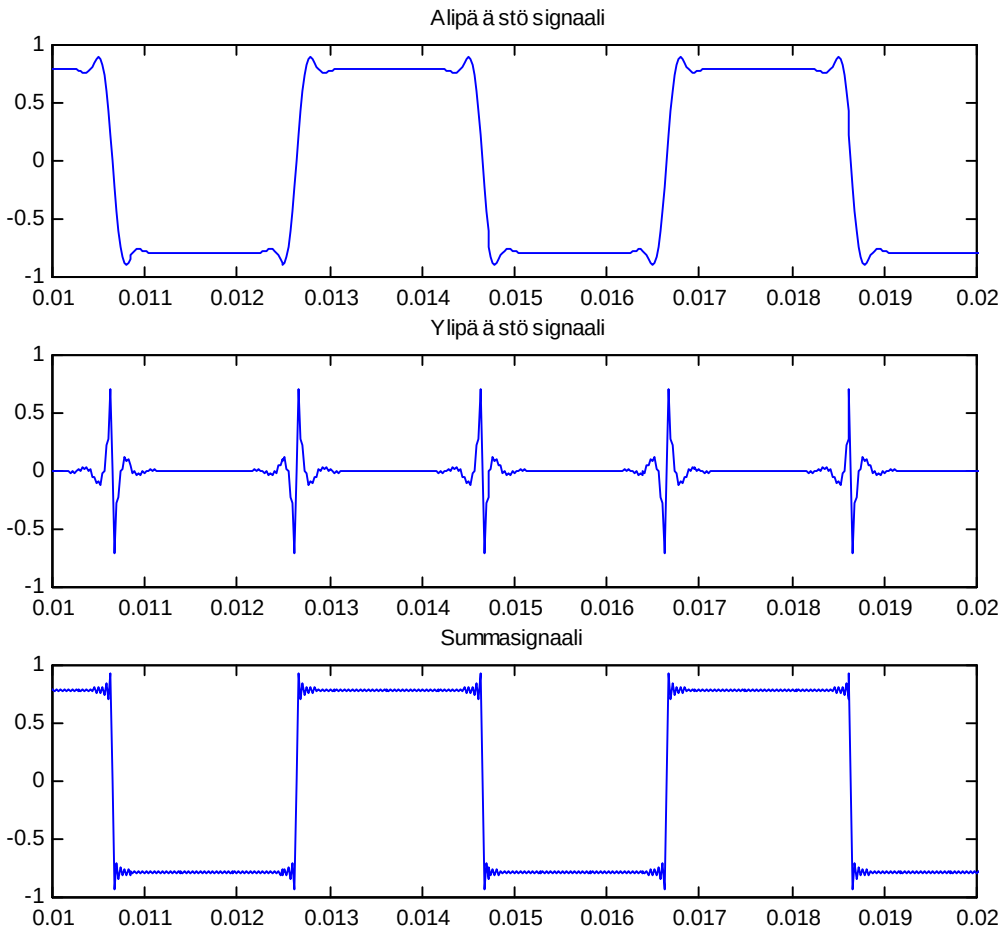
Näistä eroon pääsemiseksi täytyisi kortti rakentaa 4-kerroksiselle piirilevylle, jossa olisi omat tasot käyttöjännitteelle. Kuitenkin analogia- ja digitaaliosalle täytyisi jättää omat erotetut maa-alueet, koska muuten niiden väliin syntyy voimakas häiriön kytkeytyminen kapasitiivisesti.

Viimeisenä, mutta ei suinkaan vähäisimpänä muutoksena, koko kytkentä täytyisi koteloida tiiviiseen metallikoteloon. Tämä olisi luultavasti helpoin vaihtoehto emission vähentämiseksi, vaikka toki muutkin em. keinot olisivat luultavasti melko tehokkaita.

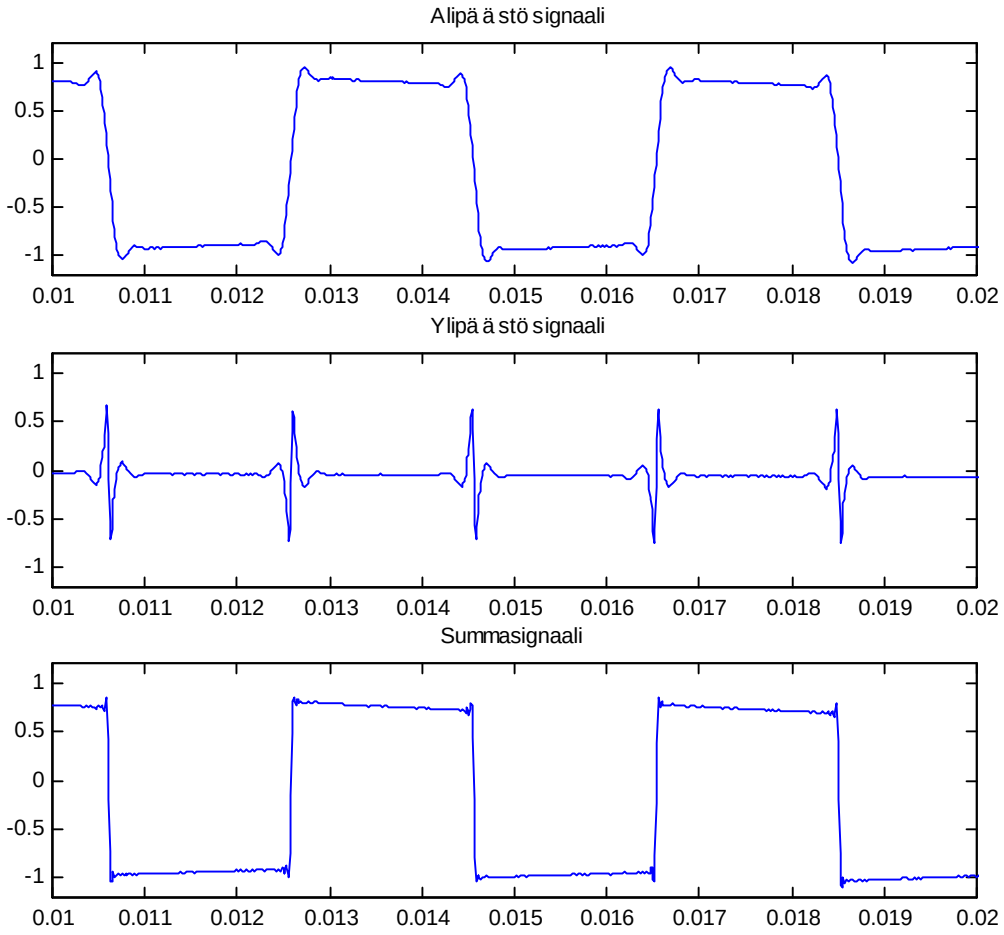
10 Suodintyyppien vertailu

Digitaaliset suotimet ovat erittäin hyvin realisoituvia ja niiden ominaisuudet voidaan hyvin pitkälle ennustaa simuloinnin avulla. Tämän varmistamiseksi alla on esitetty yli- ja alipäästösignaalit sekä mitattuna kortilta että simuloituna Matlabilla.

Suotimen tulosignaalina on 250 Hz:n taajuinen kanttiaalto ja suodatuksen rajataajuutena 3,5 kHz. Kanttiaalto on valittu tulosignaaliksi sen vuoksi, että vaihelineaarisuuden havaitseminen on erittäin helppoa signaalista, joka sisältää runsaasti harmonisia taajuuksia. Pienikin kulkuaikavirhe vääristää signaalia selvästi. Summasignaali on yli- ja alipäästösignaalin summa.



Kuva 29. Simuloitu vaste yli- ja alipäästösuotimelle (3,5 kHz) 250 Hz:n kanttiaaltosignaalilla.

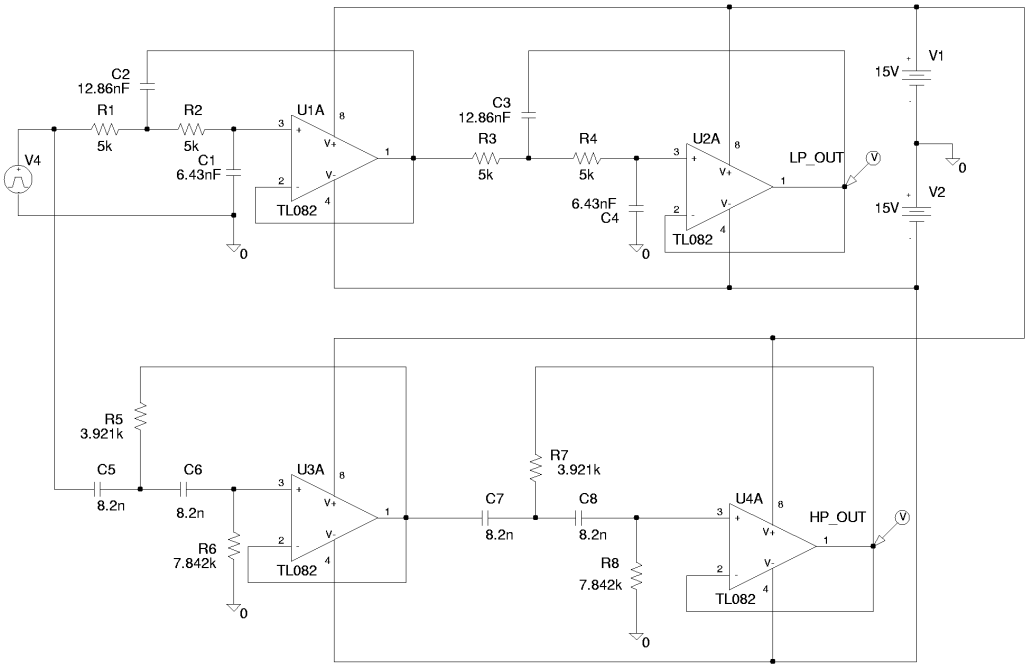


Kuva 30. Todellisesta kortista mitattu 250Hz kanttiaaltosignaalin vaste.

Mitattu signaali on käytännössä täysin sama kuin simuloitukin. Suurin poikkeama teoreettisesta aiheutuu alipäästökanavassa hiukan liian pienistä DC-erotuskondensaattoreista, jotka leikkaavat matalimpia taajuuksia hiukan pois aiheuttaen signaaliin ylä- ja alapuolen ”painumista notkolle”.

Analogisista suotimista tämän suotimen lähin vastine on lähinnä Linkwitz-Riley-tyyppinen suodin /7/, jolle on tyypillistä tasainen yli- ja alipäästön taajuusvaste (summattu ali- ja ylipäästösuotimen signaali). Tämä suodin ei kuitenkaan, kuten muutkaan analogiset suotimet, ole vaihelineaarinen.

Linkwitz-Riley-suodin koostuu kahdesta peräkkäin kytketystä butterworth-tyyppisestä 2. asteen suotimesta, joilla on sama -3 dB rajataajuus. Näin saadaan haluttu -6 dB:n vaimennus jakotaajuudelle.



Kuva 31. Linkwitz-Riley suotimen kytkentäkaavio. Suodin koostuu kahdesta peräkkäin kytketystä butterworth-tyyppisestä aktiivisuotimesta.

Linkwitz-Riley-suotimen alipäästöosio mitoitetaan valitsemalla vastukset R_1 - R_4 väliltä 4,7 kΩ - 10 kΩ. Tällöin saadaan kondensaattoreiden C_1 ja C_4 arvot kaavasta

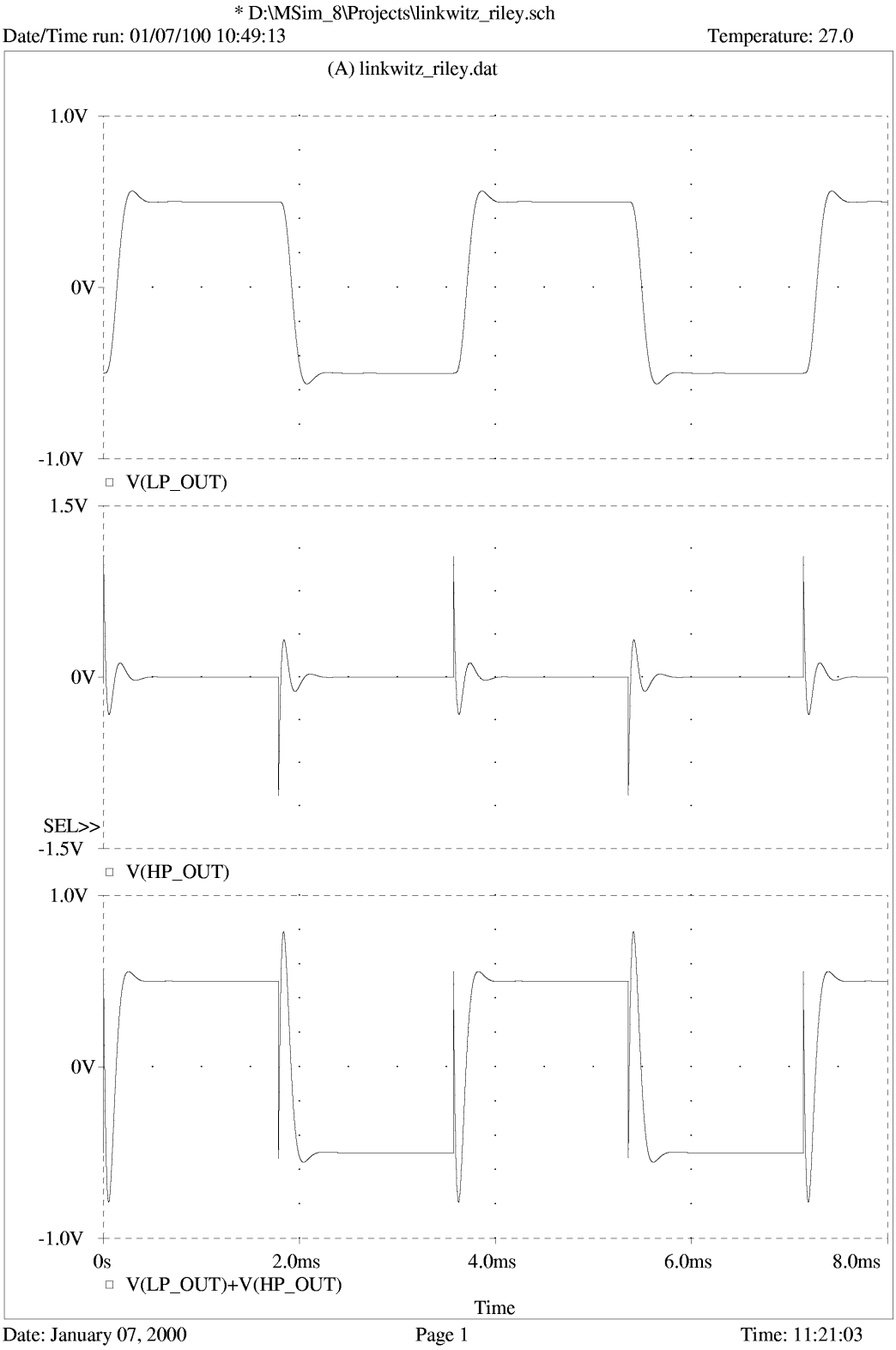
$$C_1 = C_4 = \frac{1}{2 \cdot \sqrt{2} \cdot p \cdot f_c \cdot R}, \tag{3}$$

missä R on äsken valittu vastusarvo (R_1 - R_4) ja f_c on haluttu suotimen rajataajuus hertseinä. Arvot kondensaattoreille C_2 ja C_3 saadaan kertomalla äsken laskettu kondensaattorin arvo kahdella, siis $C_2 = C_3 = 2C_1 = 2C_4$.

Ylipäästöosio mitoitetaan vastaavasti siten, että valitaan kondensaattoreiden arvoksi jokin standardiarvo väliltä 4,7 - 10 nF. Sitten lasketaan vastusten R_5 ja R_7 arvot kaavalla

$$R_5 = R_7 = \frac{1}{2 \cdot \sqrt{2} \cdot p \cdot f_c \cdot C}, \tag{4}$$

jossa C on äsken valittu kondensaattorin arvo ja f_c jälleen haluttu suotimen rajataajuus. Jälleen arvot vastuksille R_6 ja R_8 saadaan kertomalla äsken laskettu vastusten R_5 ja R_7 arvo kahdella. Ali- ja ylipäästöosioille tulee sama rajataajuus, jolloin summattu taajuusvaste on suora.



Kuva 32. Linkwitz-Riley suotimen vaste kanttiaallolle. Kuvasta näkyy selvästi suotimen kulkuaikavääristymä. Kannattaa myös huomata impulssivasteen epäsymmetrisyys.

Myös Linkwitz-Riley -suotimen rajataajuus on 3,5 kHz, joten se on täysin vertailukelpoinen aiemmin olevien digitaalisuotimen mittaustulosten kanssa. Signaalista näkyy se tyypillinen analogiasuotimen ominaisuus, että viive pienenee voimakkaasti taajuuden kasvaessa ja on suurimmillaan juuri rajataajuuden kohdalla. Tulos on simuloitu Microsim 8:lla. Käytännössä tilanne voisi olla vielä huonompi, koska käytännön komponenteilla on äärelliset tarkkuudet ja saatavissa olevia arvoja on rajallinen määrä.

Sallen-Key-kytkentä on melko herkkä komponenttiarvojen suhteen, joten pienetkin virheet komponenttiarvoissa vääristävät taajuusvastetta. Mitä korkeamman kertaluvun suodin realisoidaan yhdellä opampilla, sitä suuremmaksi toleranssivaatimukset tulevat.

Passiivisuotimissa on lisäksi vielä otettava huomioon elementtien impedanssien muuttuminen taajuuden mukana, mikä vaikuttaa oleellisesti taajuusvasteeseen.

Nämä asiat huomioiden FIR-suodin on vaihelineaarisuudeltaan aivan ylivoimainen analogiseen suotimeen nähden. Lisäksi digitaalisuodinta on erittäin helppo muokata kuuntelun aikana.

11 Jatkokehitysideoita

Jatkokehitysideoita voisivat olla esim. debuggeriliitännän (OnCE) lisäys kortille, jolloin ohjelmankehityksessä voisi käyttää apuna DSP56002 EVM:n mukana tulevaa debuggeria. Tällöin uuden ohjelmiston testaus olisi huomattavasti helpompaa. Tämä vaatisi erillisen sovitinprosessorin lisäämistä (esim. Microchipin PIC-sarjasta). Tämä ei periaatteessa olisi kovin työlästä, koska Motorolan DSP56002EVM:n mukana tulevilla levykkeillä on mukana kortilla olevan 68HC705 mikro-ohjaimen sisäinen assemblerlistaus.

Kortille voisi myös integroida jonkinlaisen käyttöliittymäprosessorin ja LCD-näytön josta sitä voisi ohjata ilman tietokonetta, tai sitten rakentaa pienen lisälaitteen, joka liittyisi RS-232-portin kautta kortille.

Tärkein suoritusarvoja parantava muutos olisi nelikerrospiirilevyn käyttöönotto, jolloin käyttöjännitteiden laatu paranisi vielä hiukan ja täten myös analogiasignaalien laatu. Tämä olisi kaupallisen valmistuksen tapauksessa välttämätöntä EMC-vaatimusten täyttämiseksi.

Mielenkiintoisia sovellusalueita kortille olisivat taajuusvastemittari MLS-signaalia käyttäen, jolloin kortilla voisi mitata kaiuttimen toistovasteen tietokoneen avulla normaalissa huoneessa tietystä alarajataajuudesta ylöspäin. Alarajaa rajoittaa vain huonekaikujen eliminointiin tarvittavan aikaikkunan lyhyys.

12 Loppupäätelmät

Työn tavoitteena oli suunnitella audiosignaalien käsittelyyn sopiva DSP-kortti. Työ osoittautui erittäin laajaksi. Eniten aikaa vieviä osia olivat piirilevyn layoutin ja ohjelmistojen suunnittelu.

Kortin skema ja piirilevy onnistuivat erittäin virheettömästi, ottaen huomioon aikataulun ja käytetyn valmistustekniikan. Ainoastaan yksi pinni koodekilta oli jäänyt kytkemättä, tämäkin virhe oli jo kytkentäkaaviotasolla.

Kortin ensimmäinen proto saatiin kuuntelukuntoon syyslomalla lokakuun puolivälissä 1999. Tällöin testattiin suodinta koemielessä edellä esitetyillä koekaiuttimilla. Suotimen todettiin toimivan määriteltyjen parametrien mukaisesti, joskin vasteita kannattaisi vielä hiukan viilailla. Olisi myös erittäin mielenkiintoista päästä mittaamaan kaiuttimien akustiset vasteet kaiuttomassa huoneessa, jolloin saisi lisätietoa mahdollisista puutteista.

Suunniteltu suodinkortti toimii aktiivisuotimena täysin suunnitellulla tavalla. Kortti on lisäksi sovellusalueiltaan huomattavasti pelkkää digitaalisuodatettua kaiutinta laajempi. Digitaalisesti voidaan tehdä erittäin jyrkkiä suotimia, joiden vaihekäyttäytyminen on moitteeton toisin kuten vastaavilla analogisilla suotimilla.

Suurin ongelma FIR-suotimen käytössä on matalien jakotaajuuksien realisoinnin ongelma, johtuen pitenevästä impulssivasteesta. Tämän ongelman poistamiseksi pitäisi lisätä suotimen kertalukua, mutta tämä lisäisi myös laskentanopeusvaatimusta.

13 LÄHTEET

1 Motorola Inc., DSP56000 Digital Signal Processor Family Manual, USA 1992, DSP56KFAMUM/AD, Internet: http://www.motorola-dsp.com

2 Motorola Inc., DSP56002 Digital Signal Processor User's Manual, USA 1993, DSP56002UM/AD REV 1, Internet: http://www.motorola-dsp.com

3 Seas, Elementtien P17REX ja 25TAF/G datalehdet, Internet: http://www.seas.no

4 Philips semiconductors, I^2C-bus and how to use it (including specifications), 1995, Internet: http://www.semiconductors.philips.com

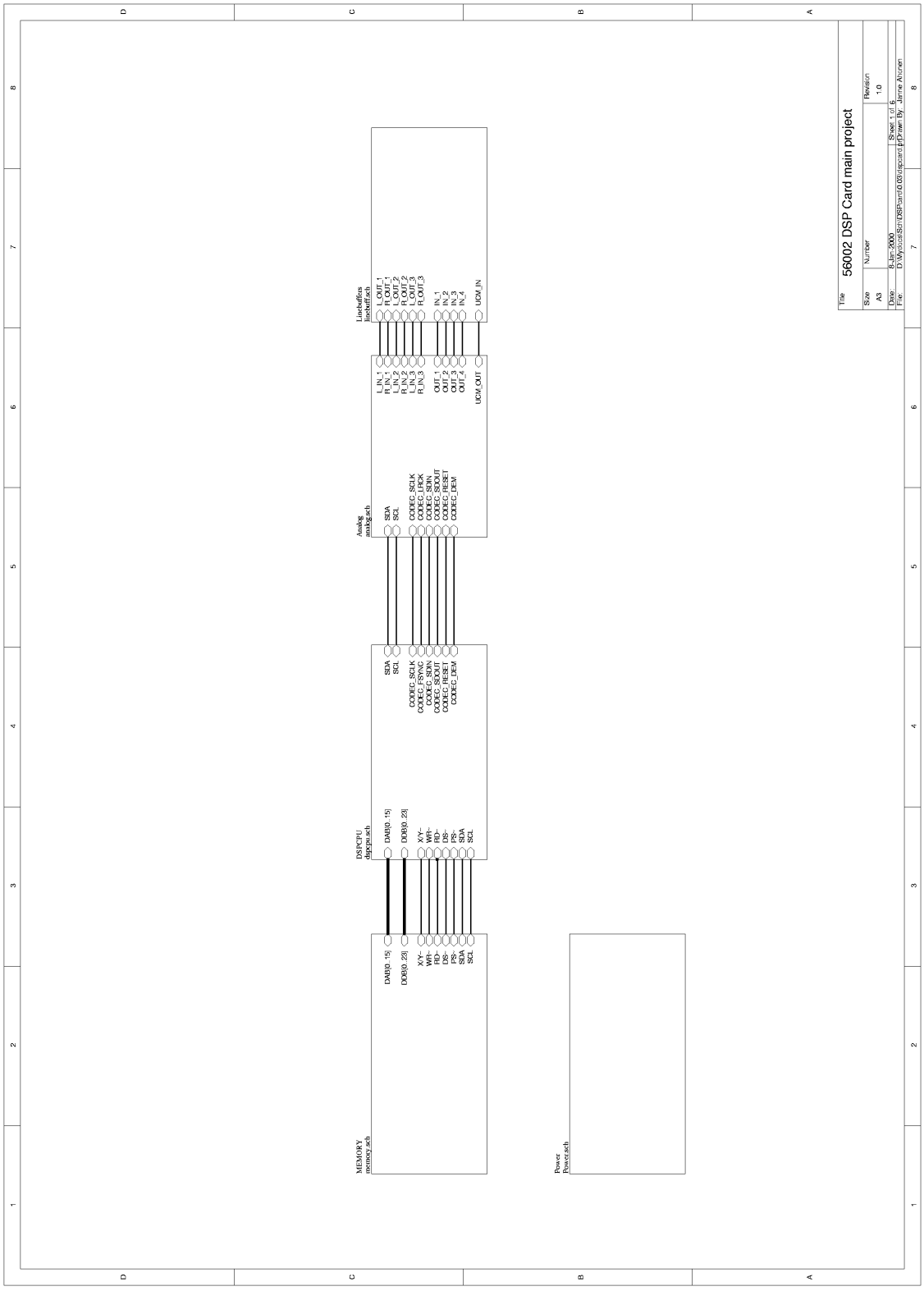
5 Cirrus Logic/Crystal, CS4225 Datalehti, Internet: http://www.crystal.com

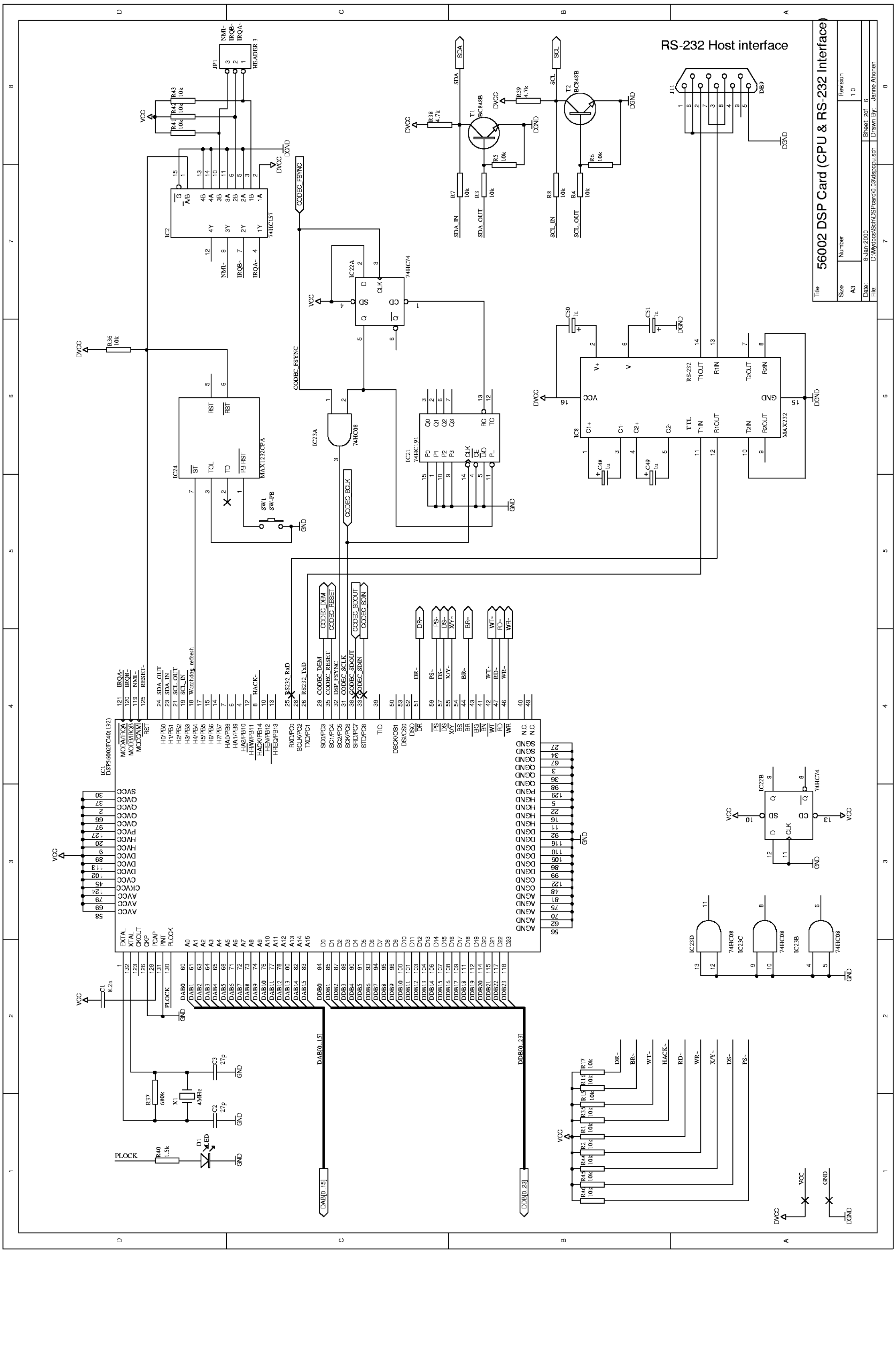
6 Erickson, Grant M., A fundamental introduction to the compact disc player. WWW: http://www.tc.umn.edu/~erick205/Papers/3011Paper.pdf

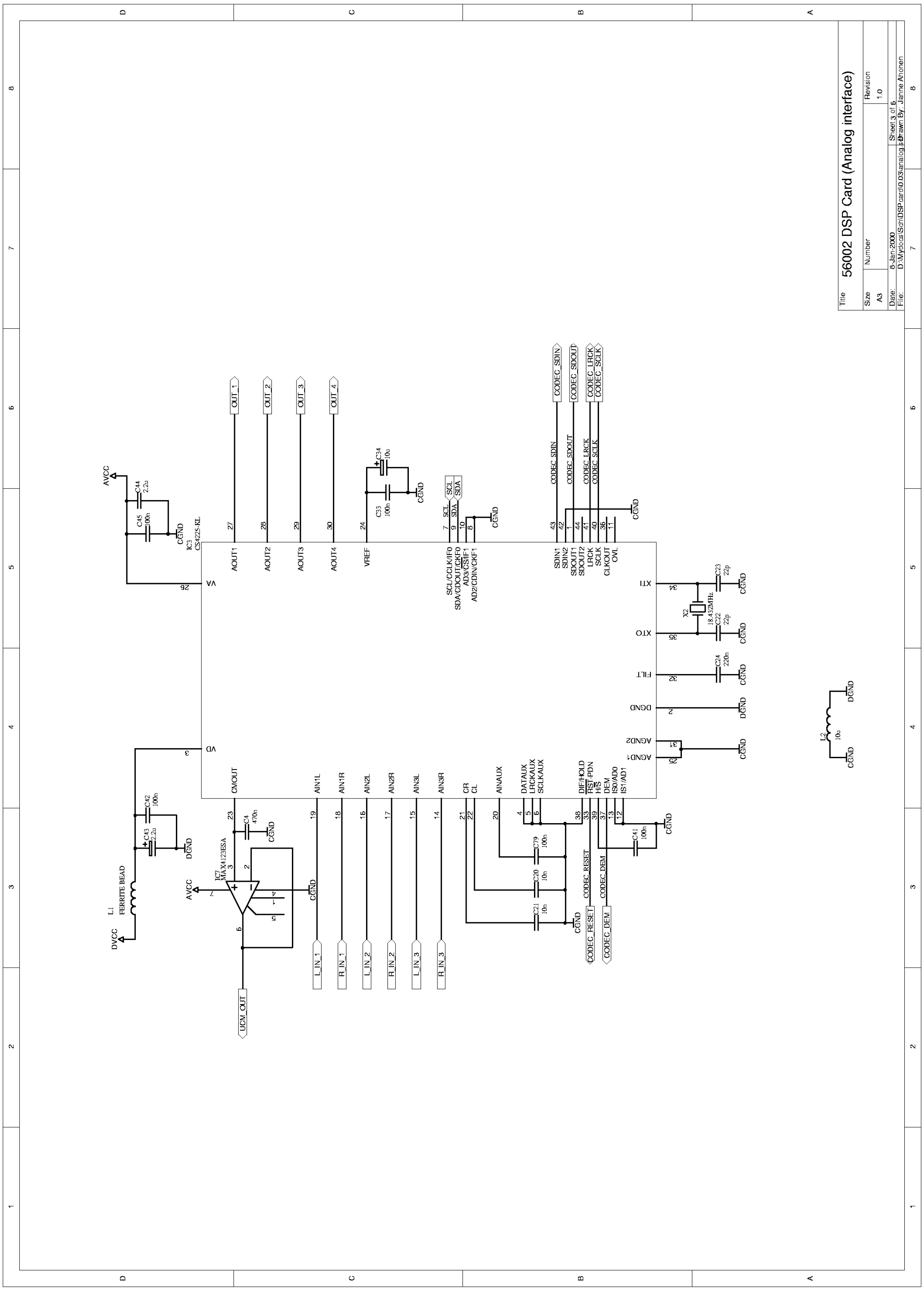
7 Linkwitz S.H., Active crossover networks for non-coincident drivers, JAES, vol. 24, January 1976, s. 2

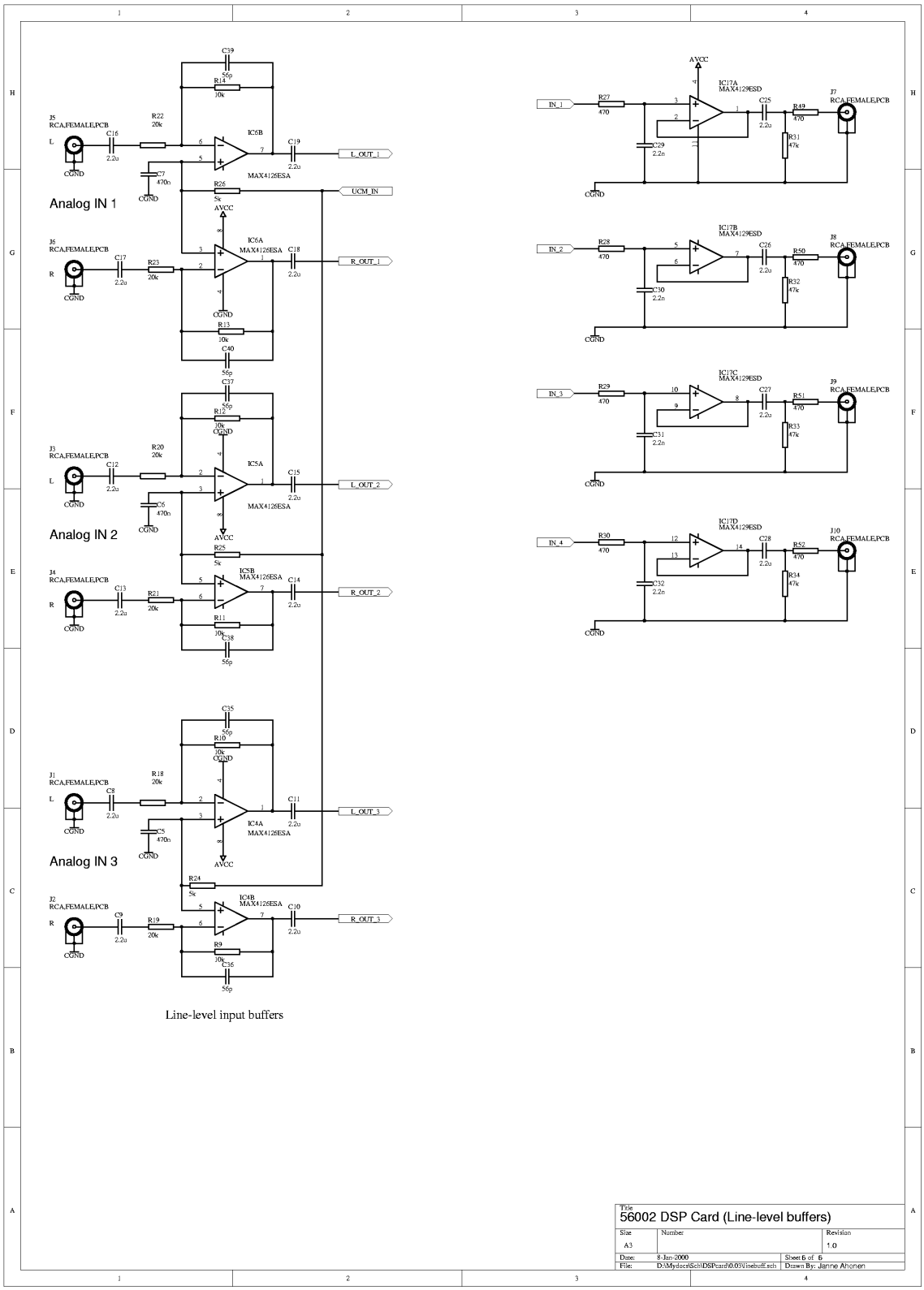
8 APR7/D Motorola Inc., Implementing IIR/FIR Filters with Motorola's DSP56000/SPS/DSP56001, Internet: http://www.motorola-dsp.com

Liite 1. Kortin kytkentäkaavio.

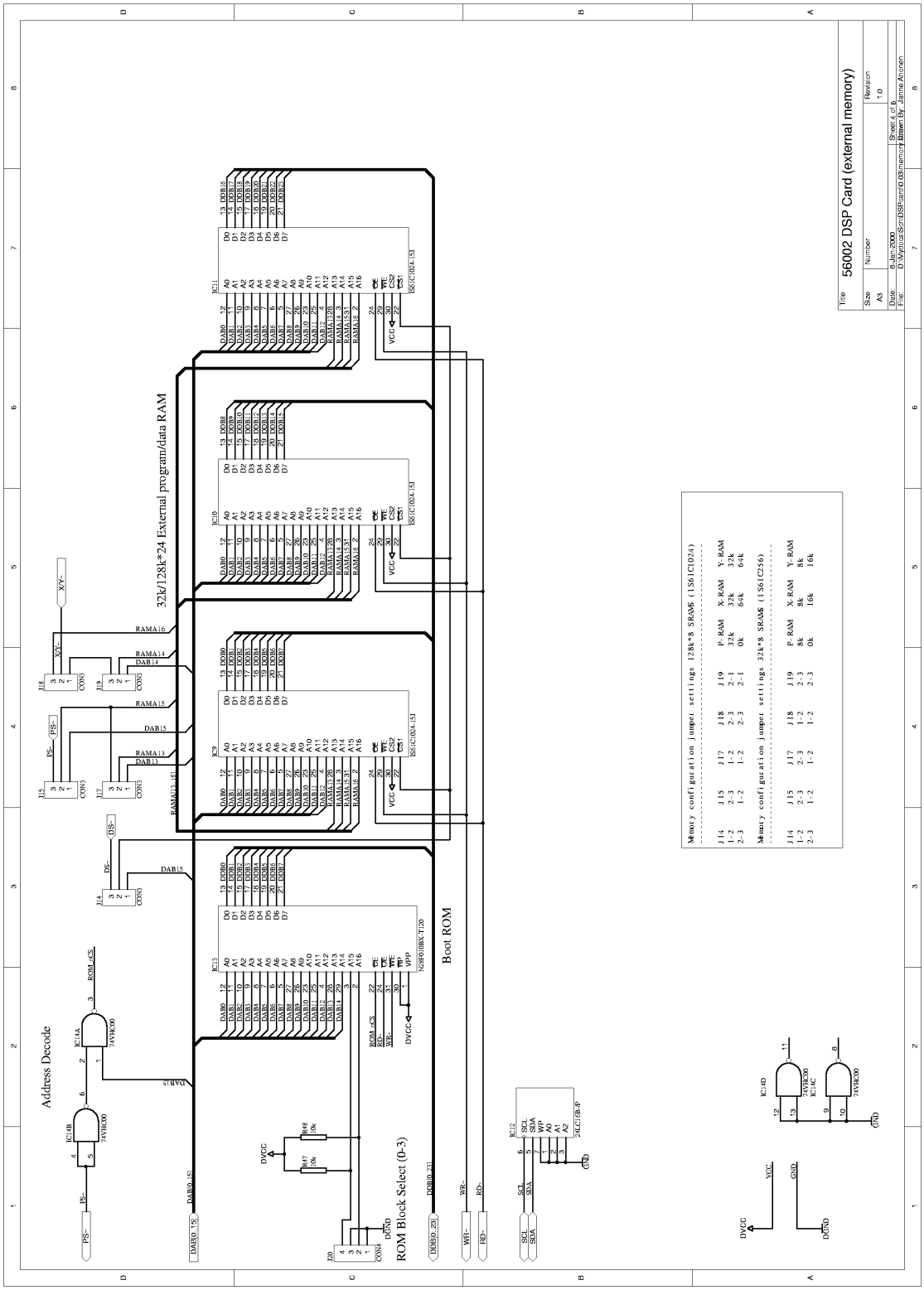


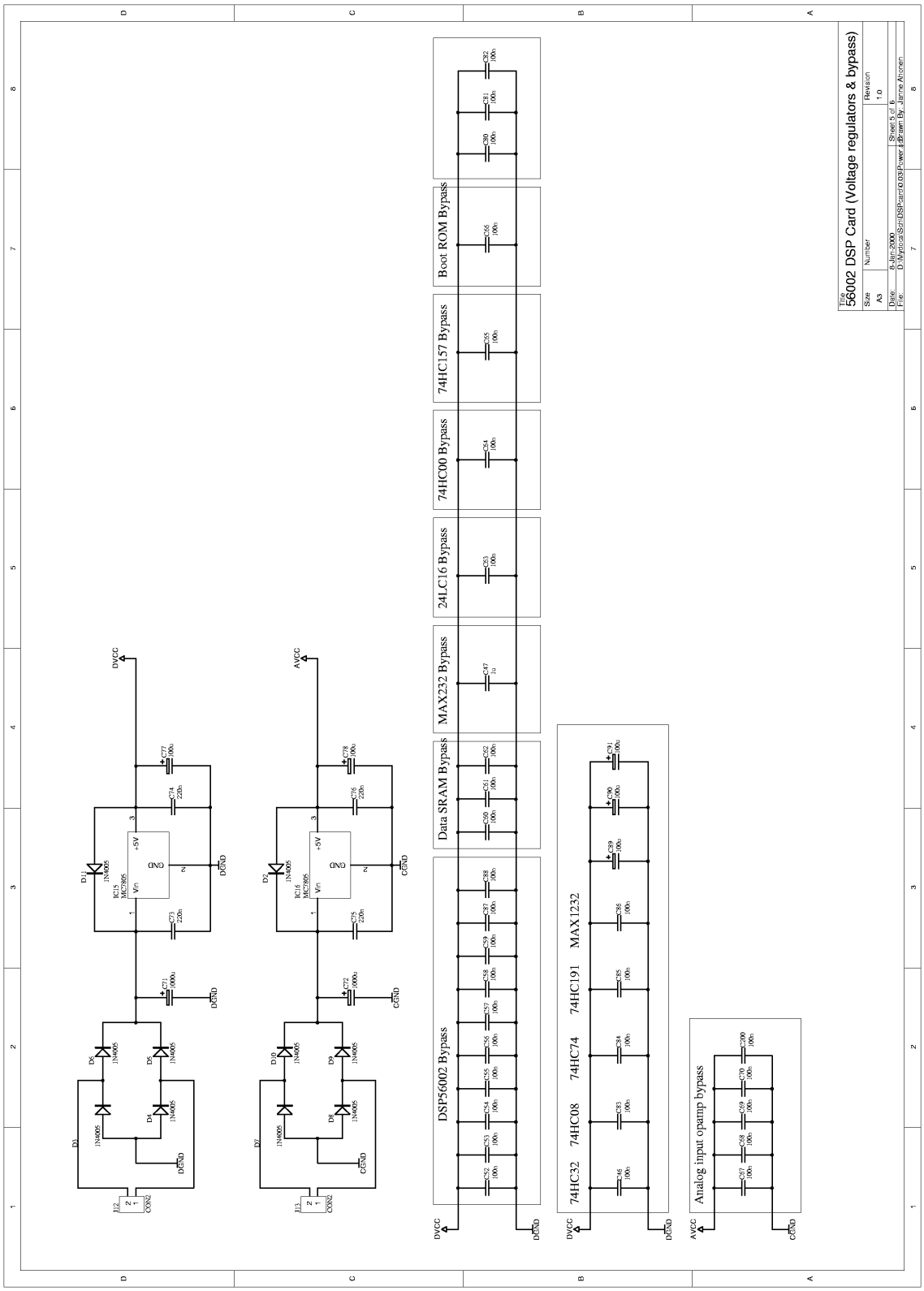

RS-232 Host interface
56002 DSP Card (CPU & RS-232 Interface)


56002 DSP Card (Analog interface)
Size A3
Revision 1.0
Date 8-Jan-2000
Sheet 3 of 6
Drawn By: Janne Ahonen
IC3 CS4225-KL
IC7 MAX4123ESA
X2 18.432MHz
L1 FERRITE BEAD
CODEC_SDIN
CODEC_SDOUT
CODEC_LRCK
CODEC_SCLK
CODEC_RESET
CODEC_DEM
OUT_1
OUT_2
OUT_3
OUT_4
L_IN_1
R_IN_1
L_IN_2
R_IN_2
L_IN_3
R_IN_3
UCM_OUT
AVCC
DVCC
CGND
DGND



Line-level input buffers


56002 DSP Card (external memory)
Address Decode
32k/128k*24 External program/data RAM
Boot ROM
ROM Block Select (0-3)
Memory configuration jumper settings 128k*8 SRAMS (1S61C1024)
Memory configuration jumper settings 32k*8 SRAMS (1S61C256)


56002 DSP Card (Voltage regulators & bypass)
DSP56002 Bypass
Data SRAM Bypass
MAX232 Bypass
24LC16 Bypass
74HC00 Bypass
74HC157 Bypass
Boot ROM Bypass
74HC32
74HC08
74HC74
74HC191
MAX1232
Analog input opamp bypass
DVCC
DGND
AVCC
AGND

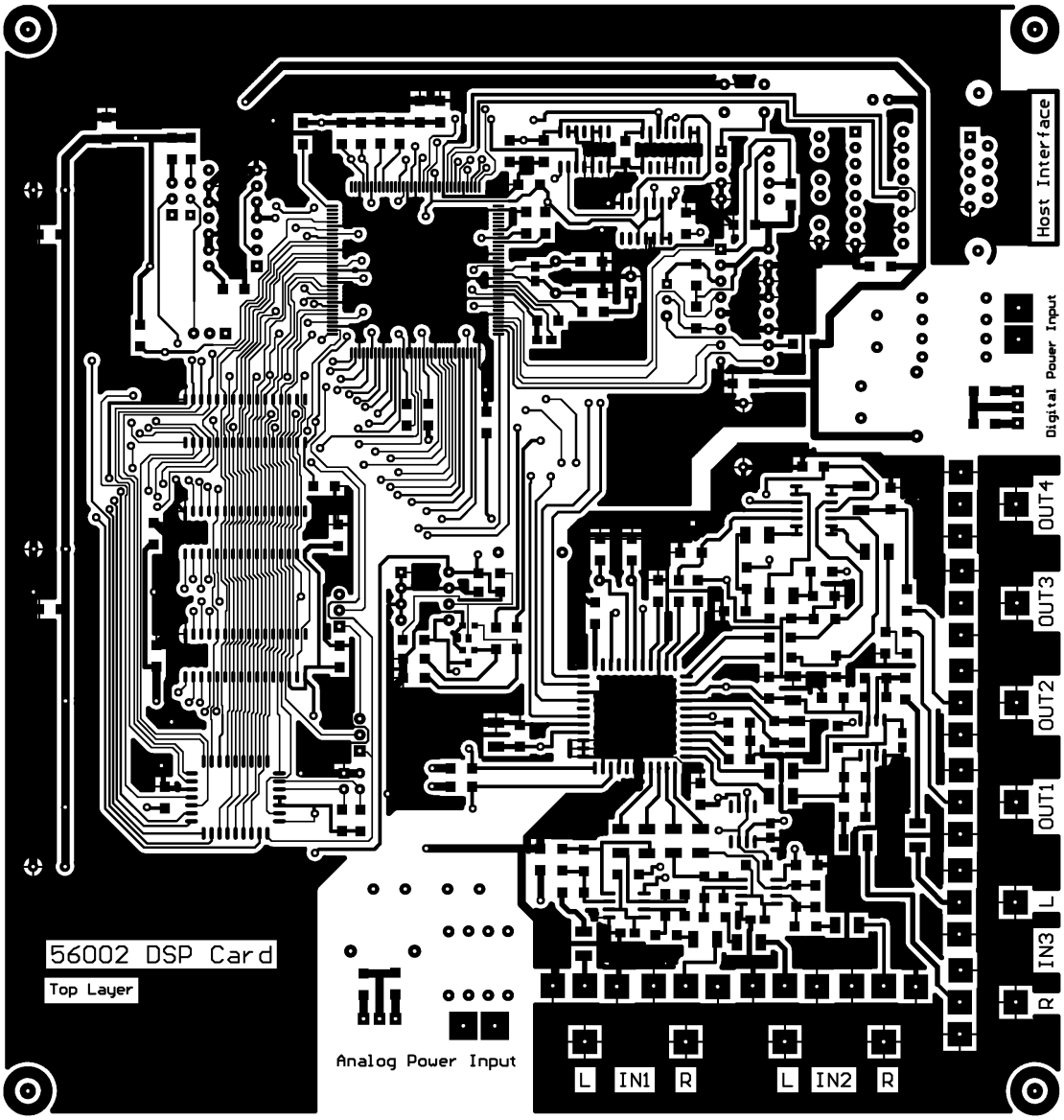
Liite 2. Osaluettelo.

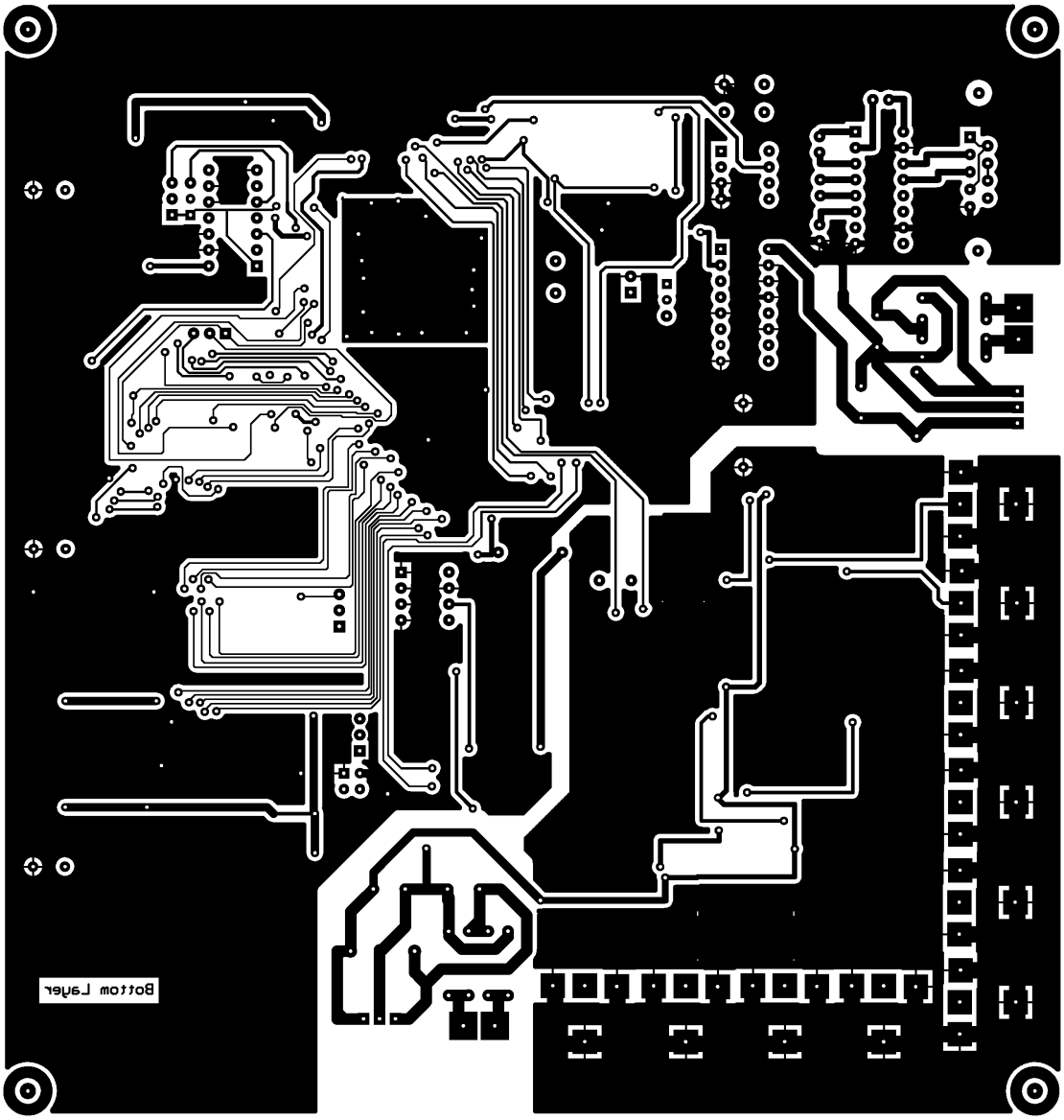
Bill of Material for D:\Mydocs\Sch\DSPcard\0.Prj

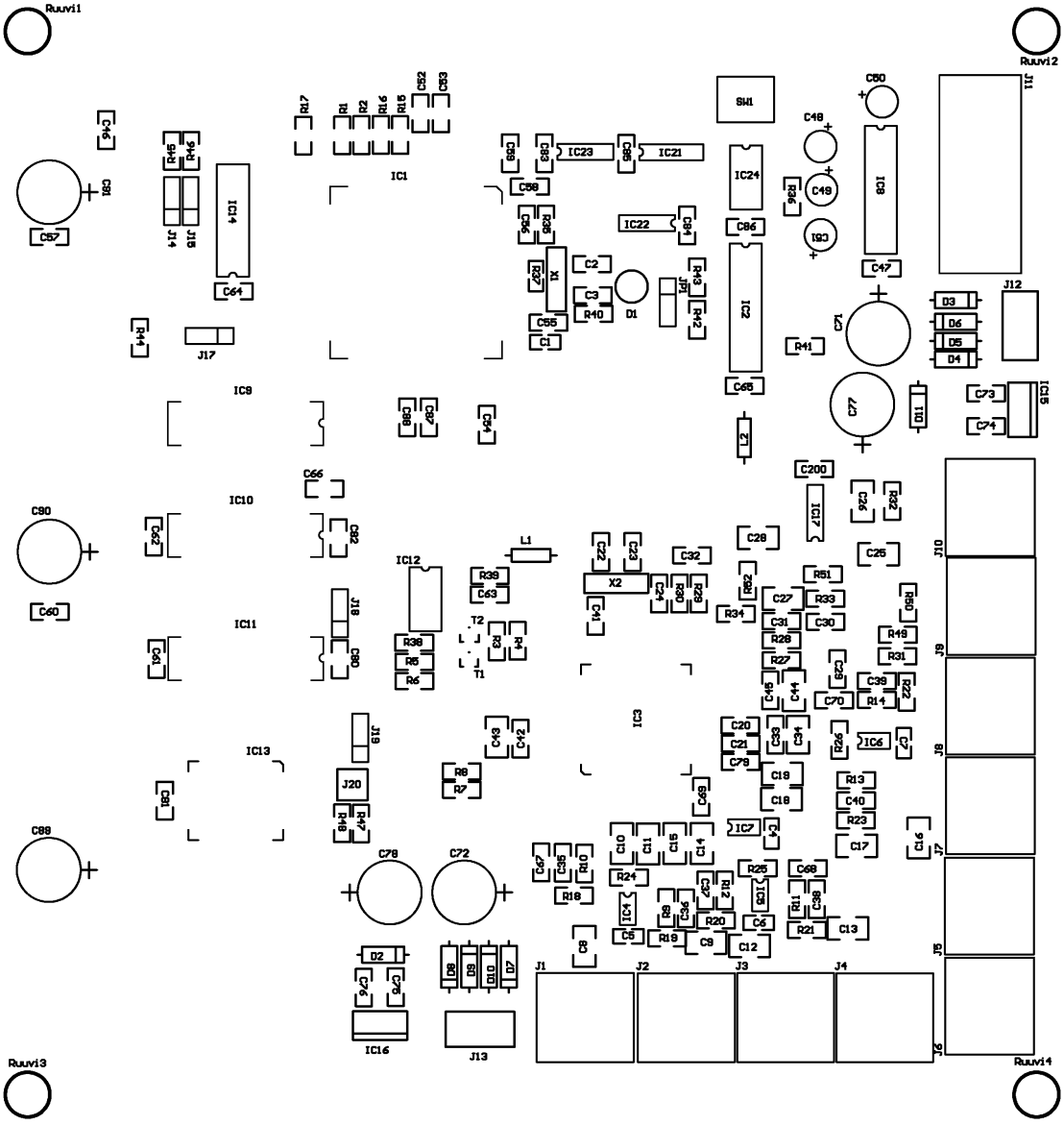
Used	Part Type	Designator	Footprint
1	1.5k	R40	1206
2	1000u	C71 C72	RB.2/.4
35	100n	C200 C33 C41 C42 C45 C46 C52 C53 C54 C55 C56 C57 C58 C59 C60 C61 C62 C63 C64 C65 C66 C67 C68 C69 C70 C79 C80 C81 C82 C83 C84 C85 C86 C87 C88	1206
5	100u	C77 C78 C89 C90 C91	RB.2/.4
27	10k	R1 R10 R11 R12 R13 R14 R15 R16 R17 R2 R3 R35 R36 R4 R41 R42 R43 R44 R45 R46 R47 R48 R5 R6 R7 R8 R9	1206
2	10n	C20 C21	1206
2	10u	C34 L2	1210
1	18.432MHz	X2	XTAL1
10	1N4005	D10 D11 D2 D3 D4 D5 D6 D7 D8 D9	DIODE0.4
5	1u	C47 C48 C49 C50 C51	RB.1/.2
4	2.2n	C29 C30 C31 C32	1206
18	2.2u	C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C25 C26 C27 C28 C43 C44 C8 C9	1310
6	20k	R18 R19 R20 R21 R22 R23	1206
5	220n	C24 C73 C74 C75 C76	1206
2	22p	C22 C23	1206
1	24LC16B-/P	IC12	DIP8
2	27p	C2 C3	1206
2	4.7k	R38 R39	1206
8	470	R27 R28 R29 R30 R49 R50 R51 R52	1206
4	470n	C4 C5 C6 C7	0805
4	47k	R31 R32 R33 R34	1206
1	4MHz	X1	XTAL1
6	56p	C35 C36 C37 C38 C39 C40	1206
3	5k	R24 R25 R26	1206
1	680k	R37	0805
1	74HC08	IC23	SO-14
1	74HC157	IC2	DIP16
1	74HC191	IC21	SO-16
1	74HC74	IC22	SO-14
1	74VHC00	IC14	DIP14
1	8.2n	C1	0805
2	BC848B	T1 T2	SOT-23-1
2	CON2	J12 J13	SCREWCON2
5	CON3	J14 J15 J17 J18 J19	SIP3
1	CON4	J20	JUMPERFIELD2
1	CS4225-KL	IC3	PLCC44
1	DB9	J11	DB9/F
1	DSP56002FC40(132)	IC1	PQFP132(T)
1	FERRITE BEAD	L1	AXIAL0.4
1	HEADER 3	JP1	SIP3
3	IS61C1024-15J	IC10 IC11 IC9	SOJ32(300)
1	LED	D1	LED1
1	MAX1232CPA	IC24	DIP8

1	MAX232	IC8	DIP16
1	MAX4123ESA	IC7	SO-8
3	MAX4126ESA	IC4 IC5 IC6	SO-8
1	MAX4129ESD	IC17	SO-14
2	MC7805	IC15 IC16	TO-220V
1	N28F010BX-T120	IC13	PLCC32
10	RCA,FEMALE,PCB	J1 J10 J2 J3 J4 J5 J6 J7 J8 J9	RCA1
1	SW-PB	SW1	BUTTONSW1

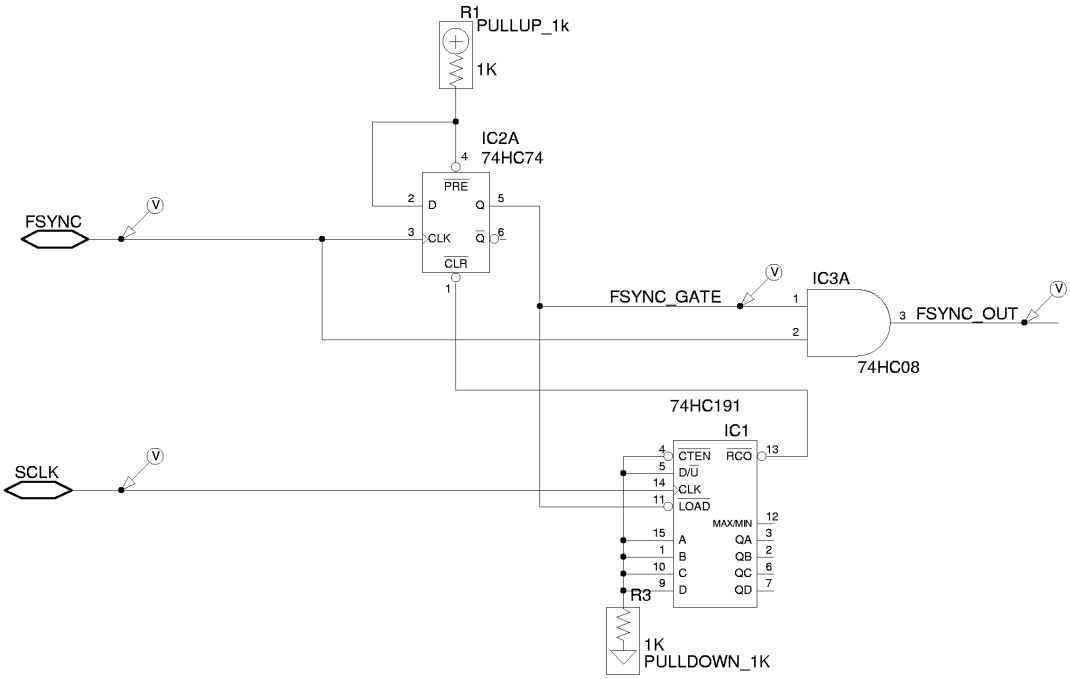
Liite 3. Kortin layout ja osasijoittelu komponenttipuolelta.



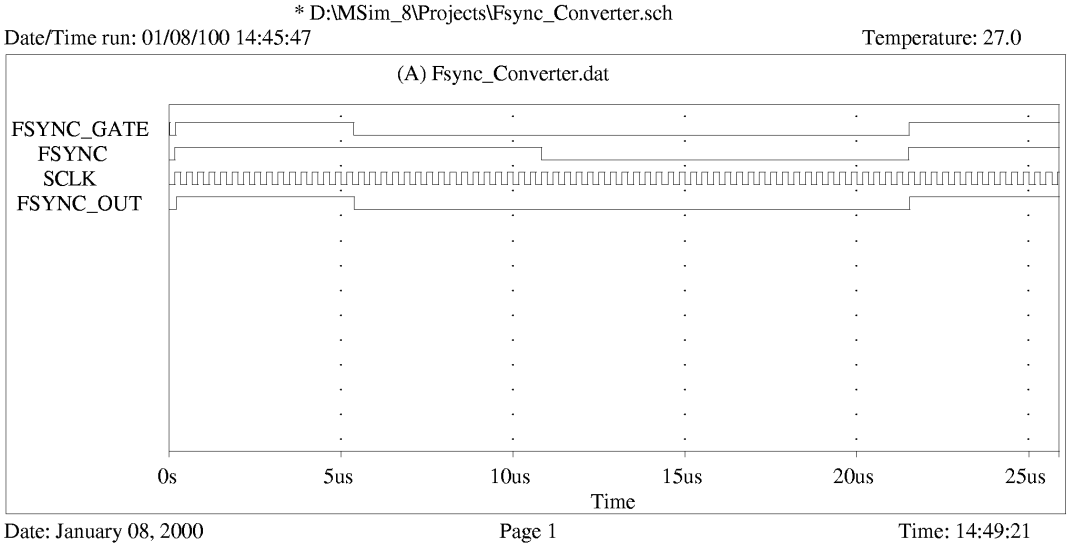

Bottom Layer


Ruuvi1
Ruuvi2
Ruuvi3
Ruuvi4
IC1
IC3
IC6
IC7
IC9
IC10
IC11
IC12
IC13
IC14
IC15
IC16
IC17
IC21
IC22
IC23
IC24
SW1
X1
X2
L1
L2
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
T1
T2
J1
J2
J3
J4
J5
J6
J7
J8
J9
J10
J11
J12
J13
J14
J15
J16
J17
J18
J19
J20
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C39
C40
C41
C42
C43
C44
C45
C46
C47
C48
C49
C50
C51
C52
C53
C54
C55
C56
C57
C58
C59
C60
C61
C62
C63
C64
C65
C66
C67
C68
C69
C70
C71
C72
C73
C74
C75
C76
C77
C78
C79
C80
C81
C82
C83
C84
C85
C86
C87
C88
C89
C90
C91
C92
C93
C94
C95
C96
C97
C98
C99
C100
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
R37
R38
R39
R40
R41
R42
R43
R44
R45
R46
R47
R48
R49
R50
R51
R52
R53
R54
R55
R56
R57
R58
R59
R60
R61
R62
R63
R64
R65
R66
R67
R68
R69
R70
R71
R72
R73
R74
R75
R76
R77
R78
R79
R80
R81
R82
R83
R84
R85
R86
R87
R88
R89
R90
R91
R92
R93
R94
R95
R96
R97
R98
R99
R100

Liite 4. Frame sync -signaalin sovituskytkentä ja ajoitussimulaatio.



Kuva 33. Word-sync-signaalin sovituskytkentä



Kuva 34. Kytkennän ajoitussimulaatio.

Liite 5. Suodinohjelmiston assembler-listaus.

```
                page    132,60
;*************************************************************************
;
; firxover.asm V1.3
;
; Active digital 2-way crossover filter for DSP56002 & CS4225 Codec board
;
; Created: Janne Ahonen 18101999
;
; Notes:
;
; 1) The WDT pin high is used to monitor filter computing time.
;
; 2) The DSP is slightly overclocked in this application (48 MHz)
;
; 3) The SCI Parameters are currently 19200, 8 databits, 1 stopbits, no parity
;
; Modification history:
;
; - 18101999 by JA  ->  - Created.
;                       - Doesn't work. (surprise!)
;
; - 19101999 by JA  ->  - Fixed interrupt register saving.
;                         (stack pointer is in r7, you fool!)
;                       - Fixed bug in jump in SSI Tx isr.
;                         Now DEM pin works ok.
;                       - Now filtering works ok.
;                       - Some cleanup done.
;
; - 22121999 by JA  ->  - Now adding the serial control support for this.
;                       - Basics converted from simptest.asm
;
; - 23121999 by JA  ->  - Now the serial interface should be reasonably
;                         behaving.
;                       - Fixed modulo setting when reading parameters from
;                         the eeprom.
;                       - Added toggling of the wdt refresh pin, to prevent
;                         wdt reset during eeprom writing.
;
; - 25121999 by JA ->   - Added support for writing into codec registers,
;                         to config input, change output attenuation etc.
;                         These settings are saved into the EEPROM,
;                         whenever COMMAND_STORE is issued.
;                       - Input channel ordering fixed, left and right are
;                         now as they should be.
;
; - 29121999 by JA ->   - Changed SCI baud rate to 19200 because the actual
;                         baud rate at 38400 is too much off. Now it should
;                         work ok, albeit slower though.
;
; - 30121999 by JA ->   - Fixed coefficient saving routine. Didn't write
;                         all bytes to EEPROM.

;-------------------------------------------------------------------------
;
; Include files
;

                nolist
                include 'ioequ.asm'
                include 'i2c.i'
                list

enable          macro
```

```
                andi    #$fc,mr
                endm

disable         macro
                ori     #$02,mr
                endm

;------------------------------------------------------------------------
;
; Filter EQUates
;

NTAPS           equ     63              ;Number of filter taps
lpcoeff         equ     $40             ;Pointer to LP coeffs in Y mem
hpcoeff         equ     $80             ;Pointer to HP coeffs in Y mem
rdelay          equ     $40             ;pointer to R Ch delay buffer
ldelay          equ     $80             ;pointer to L Ch delay buffer

;------------------------------------------------------------------------
;
;Port B allocations & EQUates
;
;                                        F T T
;                                        ENUNU
;                                        RIOIO
;
;                                        TLLAA
;                                        DCCDD
;                                        WSSSS
;                        -------------------------
;                        2222111111111100000000000
;                        3210987654321098765432100
PB_CFG          equ     %00000000001111111110101

SDA_OUT         equ     0
SDA_IN          equ     1
SCL_OUT         equ     2
SCL_IN          equ     3
WDT_REF         equ     4

;------------------------------------------------------------------------
;
;Interrupt priority register
;

IPR_CFG         equ     $7000

;------------------------------------------------------------------------
;
;Port C allocation & pin usage
;
;                                           K
;                                       DDK210LDD
;                                       TRCCCCCXX
;                                       SSSSSSSTR
;                        -------------------------
PCC_CFG         equ     %000000000000000111100011
PCDDR_CFG       equ     %000000000000000000011000

CODEC_DEM       equ     3
CODEC_nRESET    equ     4


;------------------------------------------------------------------------
;
; Serial control configuration

                                ;1111110000000000
                                ;5432109876543210
```

```
                                ;----------------
SCR_CFG         equ     %00000000000101100000010


SCI_2400        equ     311
SCI_9600        equ     77
SCI_19200       equ     38
SCI_38400       equ     19

;------------------------------------------------------------------------
;
; SSI Configuration
;
;
;                        -------------------------
;                        2222111111111100000000000
;                        3210987654321098765432l0
CRA_CFG         equ     %00000000010000l100000011
CRB_CFG         equ     %00000000111111010000000000

;------------------------------------------------------------------------
;
; X data memory variables

                org     x:$0

;------------------------------------------------------------------------
;
; CS4225 Codec register values (this data is actually fetched from the EEPROM)
;

CODEC_ADDR      equ     $20                 ;I2C address for CS4225
INCR            equ     $80                 ;Increment bit in MAP

CS4225initseq   dc      $00,$00,$00,$00     ;Output Attenuators (0x00-0x7F)
                dc      $02,$02             ;Input gain (0x00-0x1F)
                dc      $10                 ;AUX port slave (not used)
                dc      $04                 ;DSP Port master, Format 4
                dc      $68                 ;Clock mode (crystal osc)
                dc      $00                 ;Control byte
                dc      $00                 ;Input select
                dc      $00                 ;Aux control byte

;------------------------------------------------------------------------
;
; SSI Buffers
;

RX_BUFF_BASE    equ     *                   ;SSI RX ISR buffer
RX_left_adc     ds      1
RX_right_adc    ds      1
RX_aux_adc      ds      1
RX_aux_adc_2    ds      1

TX_BUFF_BASE    equ     *                   ;SSI TX ISR buffer
TX_dac_1        ds      1
TX_dac_2        ds      1
TX_dac_3        ds      1
TX_dac_4        ds      1

RX_ptr          ds      1                   ;Buffer pointers for SSI ISRs
TX_ptr          ds      1

;------------------------------------------------------------------------
;
; Temporary variables for input values
;

INPUT_L         dc      0
INPUT_R         dc      0
```

```
;------------------------------------------------------------------------
;
; I2C control & status
;

i2c_control     dc      0                   ;I2C routine control words
i2c_status      dc      0

;------------------------------------------------------------------------
;
; Parameter memory handling
;

DataWRBuffer    dc      $A0,$00,$00         ;Buffer to write parameter memory
                dc      $A0,$00,$00         ;words
                dc      $A0,$00,$00

MEM24C16ADDR    equ     $A0                 ;I2C memory address

;------------------------------------------------------------------------
;
; Serial interface buffers
;

SERBUF_SIZE         equ     7

serbuf              ds      SERBUF_SIZE
parsebuf            dc      0,0,0
sercount            dc      SERBUF_SIZE
status              dc      0
serbufptr           dc      0

STATUS_CMD_RCVD     equ     0
STATUS_CMD_OK       equ     1

COMMAND_OK          equ     'A'
COMMAND_FAILED      equ     'F'

CMDID_NOP           equ     'A'
CMDID_M_LPC         equ     'B'
CMDID_M_HPC         equ     'C'
CMDID_M_CODEC       equ     'D'
CMDID_STORE         equ     'E'
CMDID_RECALL        equ     'F'

;------------------------------------------------------------------------
;
;Y variables
;

                org     y:0

R_LP_OUT        dc      0               ;Filter outputs
R_HP_OUT        dc      0
L_LP_OUT        dc      0
L_HP_OUT        dc      0

r0st_l          dc      0               ;Temporary storage for FIR pointers
r0st_r          dc      0

                org     y:$40

                include 'lpc.i'

                org     y:$80

                include 'hpc.i'

;------------------------------------------------------------------------
```

```
;
;Interrupt vectors
;

                org     p:$0000
                jmp     main                    ;Reset vector

                org     p:$000C                 ;SSI interrupt routines
                jsr     ssi_rx_isr
                jsr     ssi_rx_isr
                jsr     ssi_tx_isr
                jsr     ssi_tx_isr
                jsr     sci_rx_int              ;SCI interrupt routines
                jsr     sci_rx_int

                org     p:$001E                 ;NMI interrupt handling
                rti

;---------------------------------------------------------------------
;
; Main program here (begins @ p:$40)
;

                org     p:$40

main            movep   #$260007,x:M_PCTL       ;set PLL for MPY of 8x
                movep   #0,x:M_BCR              ;0 wait states for all RAM

                move    #0,sp                   ;initialize hardware SP

                move    #$c0,r7                 ;Establish r7 as stack pointer
                move    #-1,m7                  ;Linear addressing

                movep   #$0000,X:M_PBC          ;Initialize port B (gpio)
                movep   #PB_CFG,X:M_PBDDR

                movep    #PCDDR_CFG,x:M_PCDDR      ;Initialize port C (SSI &
gpio)

                movep   #0,x:M_PCC
                movep   #0,x:M_PCD              ;Assert reset

                do      #1000,codec_res_loop
                rep     #1000
                bchg    #WDT_REF,x:M_PBD
codec_res_loop

                bset    #CODEC_nRESET,x:M_PCD   ;Deassert codec reset

                movep   #IPR_CFG,x:M_IPR

                movep   #CRA_CFG,x:M_CRA        ;Initialize SSI
                movep   #CRB_CFG,x:M_CRB
                movep   #PCC_CFG,x:M_PCC
                movep    #SCR_CFG,X:M_SCR           ;SCI control setup (8,n,1
format)
                movep   #SCI_19200,X:M_SCCR     ;Baud rate 19200 @ 48 MHz

                jsr     <i2c_init               ;Initialize I2C interface
                jsr     <recall_coeff           ;recall filter coefficients
                                                ;from 24C16 I2C eeprom memory

                jsr     <initsercmd             ;initialize serial interface
                jsr     <init_codec             ;Initialize codec

;---------------------------------------------------------------------
;
; Filter loop setup
;
```

```
                enable                          ;Enable interrupts

                move    #rdelay,r0              ;initialize delay buffer ptrs
                move    r0,y:r0st_r

                move    #ldelay,r0
                move    r0,y:r0st_l

;------------------------------------------------------------------------
;
; Main loop here
;


main_loop       jclr    #STATUS_CMD_RCVD,x:status,do_filter
                jsr     <makecmd
                jsr     <exec_command
                move    #COMMAND_OK,x0
                jset    #STATUS_CMD_OK,x:status,main_cmd_ok
                move    #COMMAND_FAILED,x0
main_cmd_ok     movep   x0,X:M_STXH
                jsr     <initsercmd

do_filter       jset    #2,x:M_SR,*             ;wait for next sample
                jclr    #2,x:M_SR,*

                move    #NTAPS-1,m0             ;Set up modulos
                move    #NTAPS-1,m4             ;for circular buffers

                bset    #WDT_REF,x:M_PBD        ;refresh WDT, part 1

                move    x:RX_left_adc,x0        ;Save new input values to
                move    x:RX_right_adc,x1       ;Temporary locations
                move    x0,x:INPUT_L
                move    x1,x:INPUT_R

                move    Y:R_LP_OUT,x0           ;Get last filter outputs
                move    x0,x:TX_dac_1

                move    y:R_HP_OUT,x0
                move    x0,x:TX_dac_2

                move    y:L_LP_OUT,x0
                move    x0,x:TX_dac_3

                move    y:L_HP_OUT,x0
                move    x0,x:TX_dac_4

                move    #lpcoeff,r4             ;Init LP filtering coeffs
                move    y:r0st_r,r0

; Filtering done "cleverly". Since the input signal for both filters are
; the same, the same delay storage can be used. This reduces memory usage.
; With two channels and 2-way filtering, we need only two impulse responses
; and two delays, although there are four filterings to do
; (r low, r high, l low and l high)

; Right channel first

                move     #lpcoeff,r4                ;Lowpass filtering coeffi-
cients
                move    y:r0st_r,r0

                move    x:INPUT_R,x0            ;----- "" ----- (R)
                move    x:INPUT_L,x1            ;get new sample (L)

;Right Lowpass
                clr     a               x0,X:(r0)+      y:(r4)+,y0
                rep     #NTAPS-1
                mac     x0,y0,a         X:(r0)+,x0      y:(r4)+,y0
```

```
                macr    x0,y0,a         #hpcoeff,r4
                move    a,y:R_LP_OUT

;Right Highpass

                clr     a               x:(r0)+,x0      y:(r4)+,y0
                rep     #NTAPS-1
                mac     x0,y0,a         X:(r0)+,x0      y:(r4)+,y0
                macr    x0,y0,a         (r0)-
                move    a,y:R_HP_OUT

                move    r0,y:r0st_r

;Now, process left channel similarly

                move    y:r0st_l,r0
                move    #lpcoeff,r4
                move    x1,x0

                clr     a               x0,X:(r0)+  y:(r4)+,y0
                rep     #NTAPS-1
                mac     x0,y0,a         X:(r0)+,x0  y:(r4)+,y0
                macr    x0,y0,a         #hpcoeff,r4
                move    a,y:L_LP_OUT

                clr     a               x:(r0)+,x0  y:(r4)+,y0
                rep     #NTAPS-1
                mac     x0,y0,a         X:(r0)+,x0  y:(r4)+,y0
                macr    x0,y0,a         (R0)-
                move    a,y:L_HP_OUT

                move    r0,y:r0st_l

                bclr    #WDT_REF,x:M_PBD        ;refresh WDT, part 2

                jmp     <main_loop              ;wait for next sample period

;------------------------------------------------------------------------
;
; Codec initialization
;

init_codec      move    #>CODEC_ADDR,a1
                jsr     <i2c_send_start
                move    #>INCR+1,a1
                jsr     <i2c_send
                move    #CS4225initseq,r0
                do      #10,init_codec_lp1
                move    x:(r0)+,a1
                jsr     i2c_send
init_codec_lp1

                move    #>CODEC_ADDR,a1
                jsr     <i2c_send_start
                move    #>INCR+12,a1
                jsr     <i2c_send
                move    x:(r0)+,a1
                jsr     <i2c_send
                move    x:(r0)+,a1
                jsr     <i2c_send
                jsr     <i2c_send_stop
                rts

;------------------------------------------------------------------------
;
; Coefficient store & recall routines
;

store_coeff
                move    #$40,r2
```

```
                move    #0,r0
                move    #-1,m0                  ;r0 uses linear addressing!!!
                do      #128,sc_loop
                move    y:(r2)+,a1
                jsr     write_coeff
                bchg    #WDT_REF,x:M_PBD
                move    (r0)+
sc_loop
                move    #$0,r2
                do      #16,st_codec_loop
                move    x:(r2)+,a1
                jsr     write_coeff
                bchg    #WDT_REF,x:M_PBD
                move    (r0)+
st_codec_loop
                rts

recall_coeff    move    #$40,r2
                move    #0,r0
                move    #-1,m0                  ;r0 uses linear addressing!!!
                do      #128,recall_y_loop
                jsr     read_coeff
                move    a1,y:(r2)+
                move    (r0)+
recall_y_loop
                move    #$0,r2
                do      #16,recall_x_loop
                jsr     read_coeff
                move    a1,x:(r2)+
                move    (r0)+
recall_x_loop
                rts

COEFFRW_LEN     equ     *-store_coeff

;---------------------------------------------------------------------------
; Reads one coefficient from the EEPROM memory
; The coefficient address @ r0, data in A1
;

read_coeff      jsr     form_mem_addr

                move    x:DataWRBuffer,a1
                jsr     <i2c_send_start

                move    x:DataWRBuffer+1,a1
                jsr     <i2c_send

                bset    #0,x:DataWRBuffer

                move    x:DataWRBuffer,a1
                jsr     <i2c_send_start

                clr     a

                jsr     <i2c_recv
                jsr     <i2c_recv
                bclr    #I2CC_RACK,x:i2c_control
                jsr     <i2c_recv

                rts

COEFFRD_LEN     equ     *-read_coeff

;---------------------------------------------------------------------------
; Writes one coefficient to the EEPROM memory
; The coefficient in A1, coefficient address @ r0
;

write_coeff     move    r5,x:(r7)+              ;save r5, n5 to stack
```

```
                move    n5,x:(r7)+

                move    #DataWRBuffer+2,r5      ;pointer to data 1. byte
                move     #3,n5                     ;Offset to get to next data
byte

                do      #3,wr_coeffd_lp         ;repeat 3 times
                rep     #8                      ;Shift accu a left 8 times
                asl     a
                move    a2,x:(r5)+n5            ;Store data byte
wr_coeffd_lp
                jsr     <form_mem_addr          ;convert r0 to mem address

                move    #DataWRBuffer,r5        ;Write buffer

                do      #3,WRMemLoop            ;Send 3 bytes

                move    x:(r5)+,a1              ;Send device address
                jsr     <i2c_send_start

                move    x:(r5)+,a1              ;Send memory address
                jsr     <i2c_send

                move    x:(r5)+,a1              ;Send data
                jsr     <i2c_send

                jsr     i2c_send_stop           ;Send stop condition
WRMemLoop

                move    x:-(r7),n5
                move    x:-(r7),r5

                rts

COEFFWR_LEN     equ     *-write_coeff

;-------------------------------------------------------------------------
; Address formatting subroutine for I2C bus memory 24C16
;

form_mem_addr   move    #DataWRBuffer,r6

                clr     a       r0,x0           ;Clear accu, copy addr
                move    r0,b1                   ;address to accu
                asl     b                       ;multiply addr by 2
                add     x0,b
                move    b1,b0

                do      #3,mkaddr_loop
                move    b0,a1

                rep     #7
                lsr     a
                move    #>%00001110,x0
                and     x0,a
                move    #>MEM24C16ADDR,x0
                add     x0,a
                move    a1,x:(r6)+

                move    b0,a1
                move    #>$ff,x0
                and     x0,a
                move    a1,x:(r6)+
                inc     B
                move    (r6)+
mkaddr_loop
                rts

FMEMADDR_LEN    equ     *-form_mem_addr
```

```
;------------------------------------------------------------------------
;
; SSI Receive ISR
;

ssi_rx_isr      move    r0,x:(r7)+
                move    m0,x:(r7)+
                move    #3,m0
                move    x:RX_ptr,r0
                jclr    #3,x:M_SR,next_rx
                move    #RX_BUFF_BASE,r0
                nop

next_rx         movep   x:M_RX,x:(r0)+
                move    r0,x:RX_ptr
                move    x:-(r7),m0
                move    x:-(r7),r0
                rti

;------------------------------------------------------------------------
;
; SSI Transmit ISR
;

ssi_tx_isr      move    r0,x:(r7)+
                move    m0,x:(r7)+
                move    #3,m0
                move    x:TX_ptr,r0
                jclr    #2,x:M_SR,next_tx
                move    #TX_BUFF_BASE+1,r0
                nop

next_tx         movep   x:(r0)+,x:M_TX
                move    r0,x:TX_ptr
                move    x:-(r7),m0
                move    x:-(r7),r0
                rti

SERIAL_STUFF_BEGIN      equ     *

;************************************************************************
;
; Command table
;

cmdtable        dc      CMDID_NOP,cmd_nop
                dc      CMDID_M_LPC,cmd_m_lpc
                dc      CMDID_M_HPC,cmd_m_hpc
                dc      CMDID_M_CODEC,cmd_m_codec
                dc      CMDID_STORE,store_coeff
                dc      CMDID_RECALL,recall_coeff

CMDTBL_SIZE     equ     6

;************************************************************************
;
; Commands to execute
;

cmd_nop         rts

cmd_m_hpc       move    #hpcoeff,r0
                jmp     <modify_filter
cmd_m_lpc       move    #lpcoeff,r0
modify_filter   move    x:parsebuf+1,n0
                move    x:parsebuf+2,a1
                move    a1,y:(r0+n0)
                rts

cmd_m_codec     move    #CS4225initseq,r0
```

```
                move    x:parsebuf+1,n0
                move    x:parsebuf+2,a1
                move    a1,x:(r0+n0)
                jsr     <init_codec
                rts

;*************************************************************************
;
; Execute command, if found in table
;

exec_command    move    x:parsebuf,a
                move    #cmdtable,r0
                do      #CMDTBL_SIZE,exec_cmd_loop
                movem   p:(r0)+,x0
                movem   p:(r0)+,r1
                cmp     x0,a
                jne     <exec_no_match
                jsr     (r1)
                bset    #STATUS_CMD_OK,x:status
                enddo
exec_no_match   nop
exec_cmd_loop   rts

;*************************************************************************
;
; initialize serial interface
;

initsercmd      move    #SERBUF_SIZE,r0
                move    r0,x:sercount
                move    #serbuf,r0
                move    r0,x:serbufptr
                move    #0,r0
                move    r0,x:status
                rts

;*************************************************************************
;
; Make processable words out of serial buffer
;


makecmd         move    x:serbuf,x0
                move    x0,x:parsebuf
                move    #serbuf+1,r0
                move    #parsebuf+1,r1
                do      #3,mkcmd_word_loop
                do      #3,mkcmd_byte_loop
                move    x:(r0)+,a2
                rep     #8
                asr     a
mkcmd_byte_loop
                move    a1,x:(r1)+
mkcmd_word_loop
                rts

;*************************************************************************
;
; SCI receiving interrupt
;

sci_rx_int      move    r0,x:(r7)+
                clr     B               x:serbufptr,r0
                move    x:sercount,b0

                movep   x:M_SRXL,x0
                jset    #STATUS_CMD_RCVD,x:status,scirx_quit

                dec     B
```

```
                jne     <scirx_storechar
                bset    #STATUS_CMD_RCVD,x:status
scirx_storechar move    x0,x:(r0)+

scirx_quit      move    r0,x:serbufptr
                move    b0,x:sercount
                move    x:-(r7),r0
                rti

SERIAL_STUFF_SIZE       equ     *-SERIAL_STUFF_BEGIN

;------------------------------------------------------------------------
;
; Include my I2C Library, using R6 as scratch reg.
;

                i2clib  6

PROGRAM_SIZE    equ     *

                end
```

Liite 6. I^2C-kirjaston listaus.

```
;*****************************************************************************
; I2C.I
;
; IIC single master routines for DSP56002 using port B pins
;
; Created: Janne Ahonen 17101999
;
;
; To use these routines, define following symbols:
;
; Symbol        purpose
; -------------------------------
; SDA_IN        SDA input pin
; SDA_OUT       SDA output pin
; SCL_IN        SCL input pin
; SCL_OUT       SCL output pin
;
; also define following memory locations in x memory space:
;
; Symbol        purpose
; -------------------------------
; i2c_status    i2c status bits, see definitions below
; i2c_control   i2c control bits, see definitions below
;
; Note:
;
; These routines use r7 as scratch
;
;*****************************************************************************

;I2C routine control word definitions

I2CC_START      equ     0           ;Set to begin transmit with start cond.
I2CC_STOP       equ     1           ;Set to send stop after transmit
I2CC_RACK       equ     2           ;Clear disable Acknowledge after recv.

;I2C routine status word definitions

I2CS_ACK        equ     0

                nolist

i2clib          macro   reg

;-----------------------------------------------------------------------------
;
;I2C routines begin here
;
;Define some macros

SDA_LOW         macro
                bset    #SDA_OUT,x:(r\reg)
                endm

SDA_HIGH        macro
                bclr    #SDA_OUT,x:(r\reg)
                endm

SCL_LOW         macro
                bset    #SCL_OUT,x:(r\reg)
                endm

SCL_HIGH        macro
                bclr    #SCL_OUT,x:(r\reg)
                endm
```

```
WAIT_SCL        macro
                jclr    #SCL_IN,X:(r\reg),*
                endm

GET_SDA_BIT     macro
                btst    #SDA_IN,X:(r\reg)
                endm

;------------------------------------------------------------------------
;
; I2C interface initialization routine
;
; note:
;
; The port b should be initialized before this routine is called

i2c_init        clr     a   #M_PBD,r\reg

                move    a1,x:i2c_control
                bset    #I2CC_RACK,x:i2c_control
                SDA_HIGH
                SCL_HIGH

                 SDA_LOW                      ;Send start condition followed by
stop
                jsr     <i2c_delay
                SDA_HIGH
                rts

I2CINIT_LEN     equ     *-i2c_init

;------------------------------------------------------------------------
;
; I2C send routine, send start and repeat until ack
;
; Data in A1

i2c_send_start  bset    #I2CC_START,x:i2c_control
                jsr     <i2c_send
                jclr    #I2CS_ACK,x:i2c_status,i2c_send_start
                rts

I2CSENDSTART_LEN    equ     *-i2c_send_start

;------------------------------------------------------------------------
;
; I2C send routine
;
; Data in A1

i2c_send        move    #M_PBD,r\reg

                jclr    #I2CC_START,x:i2c_control,i2c_s_nostart ;Start req?

                SCL_HIGH
                jsr     <i2c_delay
                SDA_LOW                          ;Send start condition
                jsr     <i2c_delay
                SCL_LOW
                jsr     <i2c_delay

                bclr    #I2CC_START,x:i2c_control

i2c_s_nostart   move    a1,a0
                rep     #16
                rol     A

                do      #8,i2c_send_loop        ;Send 8 bits
```

```
                rol     A1

                jcs     <i2c_send_1
                SDA_LOW
                jmp     <i2c_wait_bit

i2c_send_1      SDA_HIGH

i2c_wait_bit    jsr     <i2c_delay
                SCL_HIGH
                WAIT_SCL
                jsr     <i2c_delay
                SCL_LOW
i2c_send_loop

                SDA_HIGH                            ;SDA => high
                jsr     <i2c_delay
                SCL_HIGH                            ;clock to read ack bit
                jsr     <i2c_delay
                WAIT_SCL

                bset    #I2CS_ACK,x:i2c_status
                GET_SDA_BIT                         ;get ack bit
                jcc     i2c_send_ack_ok
                bclr    #I2CS_ACK,x:i2c_status

i2c_send_ack_ok SCL_LOW                             ;make clock low
                jsr     <i2c_delay                  ;wait half bit period

                move    a0,a1
                rts

I2CSEND_LEN     equ     *-i2c_send

;---------------------------------------------------------------------------
;
; I2C receive routine
;


i2c_recv        move    #M_PBD,r\reg
                nop
                SDA_HIGH
                jsr     <i2c_delay

                do      #8,i2c_recv_loop        ;Send 8 bits

                SCL_HIGH
                jsr     <i2c_delay

                WAIT_SCL

                GET_SDA_BIT
                rol     A1

                SCL_LOW

                jsr     <i2c_delay
                nop

i2c_recv_loop

                btst    #I2CC_RACK,x:i2c_control
                jcc     i2c_recv_nack

                SDA_LOW
                jsr     <i2c_delay
                SCL_HIGH
                jsr     <i2c_delay
                WAIT_SCL
```

```
                SCL_LOW
                jsr     <i2c_delay
                SDA_HIGH
                rts

i2c_recv_nack   SDA_HIGH
                jsr     <i2c_delay
                SCL_HIGH
                jsr     <i2c_delay
                WAIT_SCL

                SCL_LOW
                jsr     <i2c_delay
                jsr     <i2c_send_stop

                bset    #I2CC_RACK,x:i2c_control

                rts

I2CRECV_LEN     equ     *-i2c_recv

;------------------------------------------------------------------------
;
; I2C delay routine
;

i2c_delay       rep     #100                    ;wait 5 us
                nop
                rts

I2CDELAY_LEN    equ     *-i2c_delay

;------------------------------------------------------------------------
;
; I2C delay routine
;


i2c_send_stop   SDA_LOW
                jsr     <i2c_delay
                SCL_HIGH
                jsr     <i2c_delay
                SDA_HIGH
                jsr     <i2c_delay
                rts

I2CSTOP_LEN     equ     *-i2c_send_stop

I2CSIZE_EQU                                                             equ
I2CINIT_LEN+I2CSEND_LEN+I2CRECV_LEN+I2CDELAY_LEN+I2CSTOP_LEN

                endm

                list
```

Liite 7. mkhigh.m-funktion listaus

```
function h_hp = mkhigh(h_lp)
%
% h_hp = mkhigh(h_lp)
%
% Converts specified low-pass impulse response to complementary
% high-pass, so that combined frequency response is perfectly flat.
% (in terms of computational accuracy)
%
% Janne Ahonen 9.1.2000
%

filt_len=length(h_lp);

if rem(length(h_lp),2)==0
    error('Order of input impulse response must be odd');
end

middle=fix(filt_len/2+0.5);     % get index of center tap
h_hp=-h_lp;                     % invert lowpass impulse response
dc=sum(h_hp);                   % get dc amplitude response
h_hp(middle)=h_hp(middle)-dc;   % adjust center tap to cancel it
```