Tampereen ammattikorkeakoulu
Tietotekniikka
Sulautetut järjestelmät ja elektroniikka
Mika Heikkinen

Opinnäytetyö

TMS320VC5505 Ezdsp-kehitysalusta

Työn ohjaaja Yliopettaja Mauri Inha
Työn tilaaja Tampereen ammattikorkeakoulu
Tampere 9/2010

Tampereen ammattikorkeakoulu
Tietotekniikka, Sulautetut järjestelmät ja elektroniikka

Tekijä	Mika Heikkinen
Työn nimi	TMS320VC5505 Ezdsp-kehitysalusta
Sivumäärä	58 sivua
Valmistumisaika	9/2010
Työn ohjaaja	Yliopettaja Mauri Inha
Työn tilaaja	Tampereen ammattikorkeakoulu

TIIVISTELMÄ

Tässä työssä tutustuttiin Texas Instruments'n digitaalisen signaalinkäsittelyn kehitysalustaan. Kehitysalustan tärkeimmät komponentit ovat TMS320VC5505 16-bittinen signaalinkäsittelyprosessori ja AIC3204 codec-piiri.

Kehitysalustan tavoite on ollut mahdollistaa halpa, helppo ja nopea kehitystyö kannettavia ja vähän virtaa kuluttavia sovelluksia varten. Kehitysalustan mukana tulee Code Composer Studio 4 -ohjelmisto, johon on integroitu kaikki toiminnot, joita kehitysalustan ohjelmoinnissa tarvitaan.

Kehitysalustalla on tehty kaksi esimerkkiprojektia, joiden tarkoitus on havainnollistaa alustan ominaisuuksia ja käyttökohteita. Esimerkeissä käydään läpi codec-piirin ohjelmointia, codec-piirin ja prosessorin välistä tiedonsiirtoa sekä AD- ja DA-kanavien signaalinkäsittelyominaisuuksia. Lisäksi on tehty ohjeet Code Composer Studion kehitysalustan asennukseen, käyttöönottoon ja käyttöön.

Avainsanat Digitaalinen signaalinkäsittely, näytteenottotaajuus, kehitysalusta

TAMK University of Applied Sciences
Information Technology, Embedded systems and electronics

Writer	Mika Heikkinen
Thesis	TMS320VC5505 development board
Pages	58 pages
Graduation time	September 2010
Thesis supervisor	Senior lecturer Mauri Inha
Co- operating company	TAMK University of Applied Sciences

ABSTRACT

The purpose of this thesis was to study Texas Instrument's TMS320VC5505 Ezdsp-development board for digital signal processing.

Objective of this development board is to allow cheap, easy, and fast way to create low power digital signal processing applications. Development package includes Code Composer Studio 4 software for programming and debugging applications.

Work includes two example project and quick start guide for new users.

Keywords Digital signal processing, sampling rate, development board

Alkusanat

Tämän työn tarkoituksena oli oppia DSP-prosessorin toimintaa ja ohjelmoimista. Työ tehtiin pääasiassa kesällä 2010 Tampereen ammattikorkeakoulun tiloissa. Työ oli mielenkiintoinen, sekä opettava.

Kiitän Tampereen ammattikorkeakoulua työskentelytiloista, sekä Mauri Inhaa hyvästä opinnäytetyön aiheesta ja opastuksesta. Kiitän myös kaikkia kirjallisen osuuden oikolukemisessa, sekä kielioppissa auttaneita.

Tampereella 24. marraskuuta 2010

Mika Heikkinen

Sisällysluettelo

LYHENTEET JA TERMISTÖ

Jtag	IEEE 1149.1, mikropiirien testauksessa käytetty portti, jonka avulla voidaan ajaa ohjelmaa käsky kerrallaan tai keskeytyspisteeseen saakka.
FFT	Fast Fourier transform, jatkuva integraalimuutos.
IFFT	Inverse fast Fourier transform, fft: n käänteisoperaatio.
BGA	Ball grid array, pintaliitoskotelo jossa ulkoiset johdinjalat on korvattu pintaliitokseen sopivilla juotospallonastoilla.
ADC	Analog to digital converter, muuntaa analogisen signaalin digitaaliseksi.
RTC	Real time clock, reaali-aika kello.
WDT	Watchdog timer, vahtikoira ajastinta käytetään piirin resetoimiseen vikatilanteessa jossa ohjelma jumiutuu.
GPT	General purpose timer, yleiskäyttöinen ajastin.
SPI	Serial Peripheral Interface, Motorolan tiedonsiirtoprotokolla
I2C	Inter-Integrated Circuit bus, Philipsin tiedonsiirtoprotokolla.
UART	Universal Asynchronous Receiver Transmitter, lähettimenä ja vastaanottimena toimiva sarjaliitäntäpiiri.
MMC	Multimedia Memory Card, muistikortti
SD	Secure Digital, muistikortti
DMA	Direct memory acces, oikosiirrolla voidaan siirtää muistista dataa esimerkiksi oheislaitteelle ilman, että sitä kuljetetaan prosessorin kautta.
PLL	Phase locked loop, vaihelukittu silmukka

1 Johdanto

TMS320VC5505 eZdsp on Texas Instruments’n julkaisema kehitystalusta digitaalisen signaalinkäsittelyn sovelluksille. Kehitysalusta perustuu Texas Instruments’n 16-bittiseen TMS320VC5505 signaalikäsittelyprosessoriin.

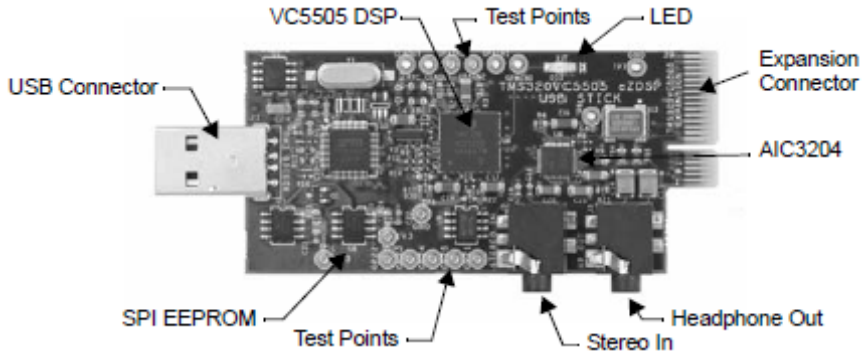
Kehitysalustan ulkoiset mitat ovat 80x381mm ja se on varustettu USB-liitännällä, jonka kautta prosessorin ohjelmointi tapahtuu, käyttöjännitteet kortille saadaan myös USB-liitännän kautta, jolloin ei ole tarvetta ylimääräisille kaapeleille. Piirin ohjelmointia varten kehityspaketin mukana tulee Code Composer Studio 4 kehitysympäristö.

Tämän työn tarkoitus on tutustua kehitysalustaan, oppia sen ominaisuuksia ja kartoittaa käyttömahdollisuuksia. Lisäksi on tarkoitus luoda ohjeet CCS4:n asennukseen, kehityskortin käyttöönottoon ja käyttöön.

Työssä tehdään kaksi esimerkkiprojektia, joiden on tarkoitus demonstroida kehitysalustan ominaisuuksia ja ohjelmointia.

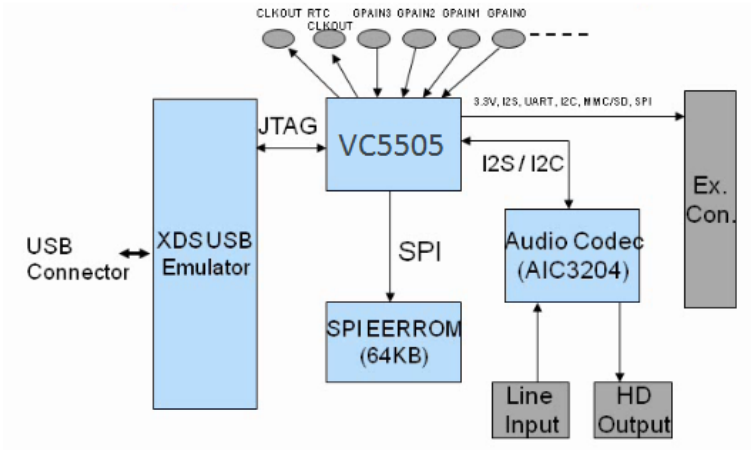
2 TMS320VC5505 eZdsp-kehitysalusta

TMS320VC5505 eZdsp on Texas Instruments'n julkaisema kehitysalusta digitaalisen signaalinkäsittelyn sovelluksiin. Tarkoituksena on ollut mahdollistaa helppo, halpa ja nopea tapa kehittää, sekä testata uusia sovelluksia. Kehitysalustan mukana toimitetaan CCS4-ohjelma DVD-levyllä. Korttiin on valittu komponentit jotka soveltuvat kannettaviin, sekä vähän virtaa kuluttaviin sovelluksiin. Kuviossa 1 näkyy kehitysalusta ja sen tärkeimmät osat.



Kuvio 1: TMS320VC5505 eZdsp kehitysalusta (TMS320VC5505 eZdsp Technical Reference 2009, 8)

Kehitysalustaan kuuluu TMS320VC5505 DSP-prosessori, XDS100 jtag-emulaattori, 64KB SPI EEPROM-muistipiiri, mittauspisteitä, ledi, stereo sisään/ulostulo, AIC3204 codec-piiri ja kytkentänastat sovelluksien laajennusta varten. Laitteiden yhteys toisiinsa näkyy lohkokaaviosta (kuvio 2). (TMS320VC5505 eZdsp Technical Reference 2009, 8)



Kuvio 2: Kehitysalustan lohkokaavio (TMS320VC5505 eZdsp USB Stick Development Tool Hands-on Training)

3 Code composer studio 4-ohjelmisto

Code Composer Studio on Eclipsen ohjelmointiympäristöön perustuva ohjelmisto, joka on tarkoitettu Texas Instruments'n signaalinkäsittelyprosessoreille ja mikrokontrollereille. CCS4 soveltuu C, C++ sekä assembly kielillä tapahtuvaan ohjelmistojen kehitykseen.

3.1 Ohjelman ominaisuudet

Kehitysympäristöstä on pyritty rakentamaan mahdollisimman kattava ohjelmointityökalu, jossa on integroituna kaikki tarvittavat ominaisuudet samassa paketissa. Seuraavaksi esitellään CCS4:n tärkeimmät ominaisuudet ja niiden käyttökohteet.

Debuggeri on työkalu, jota käytetään virheiden etsimiseen koodista. Koodin keskelle voidaan sijoittaa keskeytyksiä, mikäli halutaan tarkastella muuttujien, muistipaikkojen tai rekistereiden arvoja kesken ohjelman suorituksen. Vaihtoehtoisesti koodia voidaan ajaa käsky kerrallaan ja tutkia käskyjen vaikutusta sovellukseen.

Profiloija mittaa kirjoitettujen funktioiden suorituskykyä ja varmistaa, että käytetyn prosessorin resurssit riittävät suorittamaan halutut toimenpiteet. Näin koodista voidaan jo kehitysvaiheessa tehdä riittävän suorituskykyistä.

Testausta varten CCS4:ä on mahdollisuus automatisoida tiettyjä yleisimpiä tehtäviä, jotta pitkiä testisessioita voidaan suorittaa ilman, että käyttäjän tarvitsee puuttua ohjelman suoritukseen.

Kääntäjän tarkoitus on kääntää ohjelmoijan kirjoittama koodi konekieliseksi, CCS4:n kääntäjä on optimoitu digitaalisen signaalinkäsittelyn sovelluksiin.

Simulaattorilla on mahdollisuus testata ohjelmia ilman, että varsinaiselle kehitysalustalle tarvitsee kirjoittaa mitään.

DSP/BIOS on reaaliaikaydin, jolla voidaan siirtää osa prosessorin resursseja vievistä toimenpiteistä isäntäkoneelle. (Texas Instruments. Code Composer Studio Ide. Viitattu 20.7.2010)

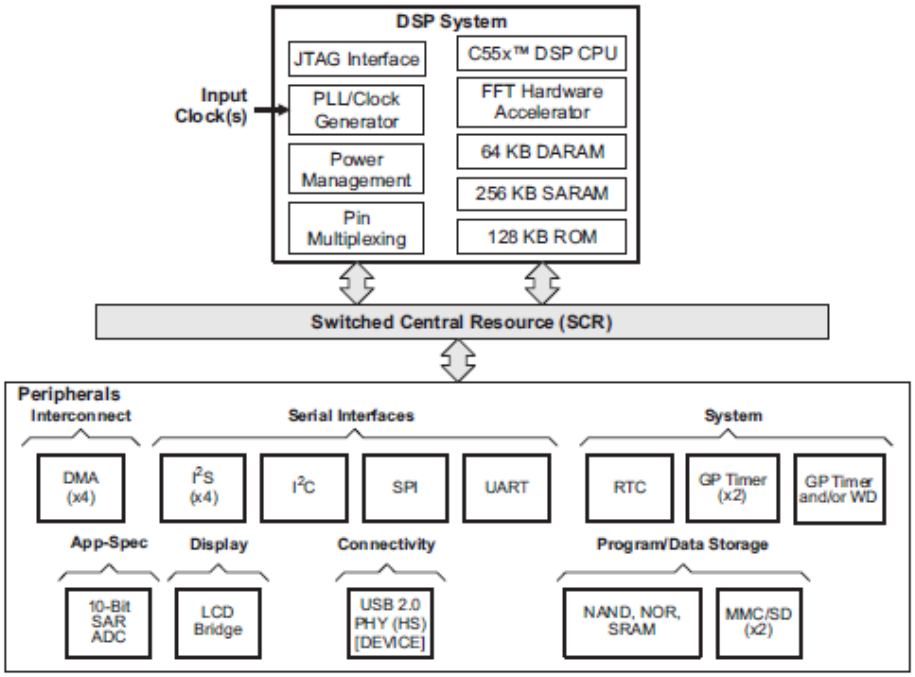
4 TMS320VC5505-prosessori

TMS320VC5505 on 16-bittinen kiintolukuprosessori, joka kuuluu C5000 piiriperheeseen. TMS320VC5505 perustuu C55x -arkkitehtuuriin jonka päätavoitteena on ollut hyvän suorituskyvyn saavuttaminen todella pienellä virrankulutuksella ja se on erityisesti tarkoitettu matalan tehonkulutuksen sovelluksiin, kuten kannettaviin audio-, lääkintä- ja kommunikaatiolaitteisiin. (TMS320VC5505 Fixed-Point Digital Signal Processor 2009, 2)

Prosessori kykenee toimimaan 60MHz taajuudella 1,05 voltin jännitteellä ja 100MHz taajuudella 1,3 voltin jännitteellä. Tehonkulutus on aktiivisena 0,15mW/MHZ 1,05 voltin jännitteellä ja 0,22mW/MHz 1,3 voltin jännitteellä. Lepotilassa kulutus on 1,05 voltin jännitteellä 0,15mW ja 1,3 voltin jännitteellä 0,28mW.(TMS320VC5505 Fixed-Point Digital Signal Processor 2009, 7)

Prosessoriin kuuluu FFT-kiihdytin joka kommunikoi prosessorin kanssa C-kielellä kutsuttavien funktioiden avulla. Kiihdytin tukee FFT-ja IFFT-muunnoksia 8 pisteestä 1024 pisteeseen saakka. Muistia piirillä on 320KB RAM:a josta 64KB DARAM:a ja 256 SARAM:a, ROM-muistia on 128KB.

Prosessorin oheislaitteita ovat lcd ohjain, ADC, RTC, WDT, GPT(2x), USB2.0, I2S(4x), SPI, I2C, UART, MMC/SD ja DMA. Kotelo on 196 nastainen 10x10mm BGA-kotelo. Prosessorin sisäinen rakenne ja laitteiden yhteydet toisiinsa näkyvät lohkokaaviosta (kuvio 3).



Kuvio 3: Prosessorin lohkokaavio (TMS320VC5505 DSP System user's guide 2009, 12)

4.1 Virrankulutuksen hallinta

Monissa paristokäyttöisissä sovelluksissa on erityisen suuri tarve minimoida virrankulutus, pienikin säästö saattaa tuoda merkittävän lisän laitteen käyttöaikaan. Virrankulutusta voidaan jakaa kahteen eri osaan, aktiiviseen virrankulutukseen ja vuotovirtaan. (TMS320VC5505 DSP System user's guide 2009, 49)

Aktiivinen kulutus määräytyy suoritettujen toimintojen ja käytetyn kellotaajuuden mukaan. Aktiivista virrankulutusta voidaan useasti pienentää joko laskemalla kellotaajuutta mahdollisimman paljon siten, että tarvittavat tehtävät voidaan vielä suorittaa. Toinen vaihtoehto on suorittaa tehtävät suurella kellotaajuudella ja tiputtaa taajuutta sen jälkeen, kunnes suuremmalle taajuudelle tulee taas tarvetta. Vuotovirran suuruus taas on täysin kellotaajuudesta riippumatonta ja sitä tapahtuu aina kun laite on päällä. Ainoa tapa välttyä vuotovirralta on sammuttaa laite tai joku laitteen osa kokonaan. (TMS320VC5505 DSP System user's guide 2009, 49)

Virransäästötoimenpiteet TMS320VC5505 prosessorissa:

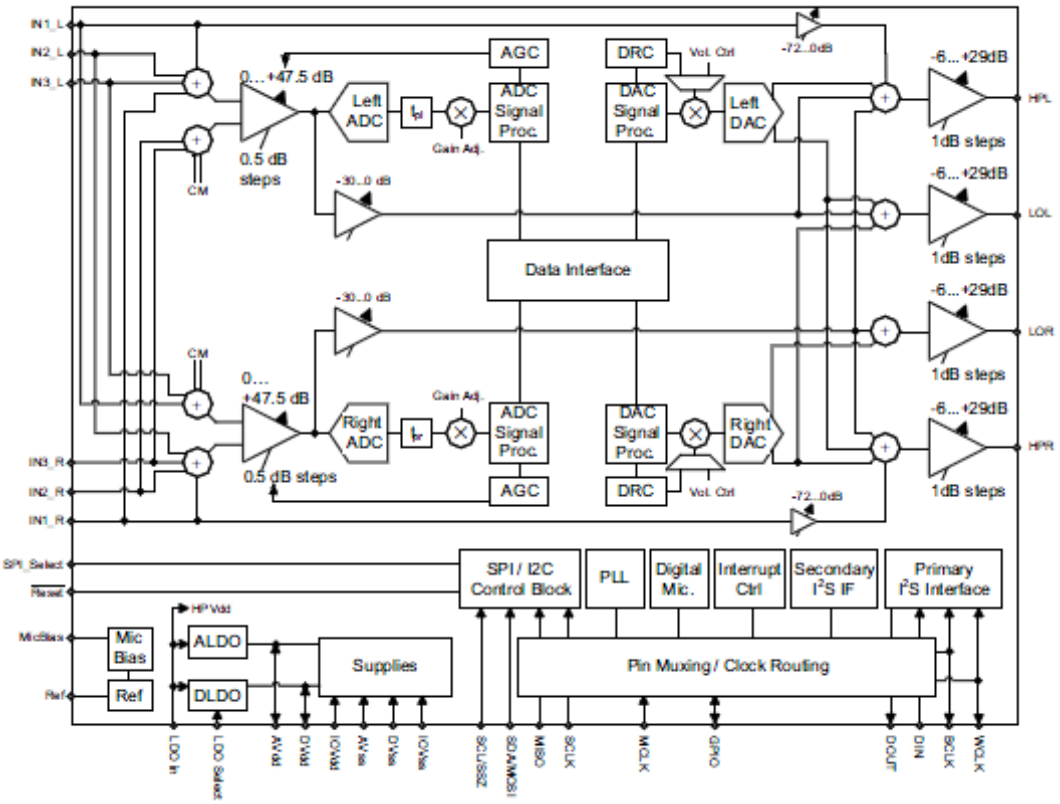
- PLL voidaan kytkeä pois päältä, mikäli sitä ei käytetä.
- Ytimen jännitettä voidaan säätää ajon aikana pienemmäksi tai suuremmaksi riippuen tarvittavasta suorituskyvystä.

- DARAM/SARAM-muistit voidaan asettaa matalatehoiseen tilaan, mikäli ei ole tarvetta lukea tai kirjoittaa muistiin.
- Prosessori on jaettu sisäisiin tehonhallintalohkoihin, joista voidaan sammuttaa sellaiset joita ei tarvita.
- I/O porttien jännitetasoja ja asettumisnopeutta voidaan laskea suorituskyvyn kustannuksella.
- USB-lisälaitteet voidaan sammuttaa, mikäli niitä ei käytetä
(TMS320VC5505 DSP System user's guide 2009, 49)

5 TLV320AIC3204-piiri

TLV320 AIC3204 on Texas Instruments'n ohjelmoitava codec-piiri joka on suunniteltu matalajännitteisiin sovelluksiin, kuten paristokäyttöisiin ääni- ja puhelinlaitteisiin. AIC3204 ohjelmoidaan sen rekistereiden avulla, joihin voidaan kirjoittaa joko I2C tai SPI tiedonsiirtoprotokollaa käyttäen.

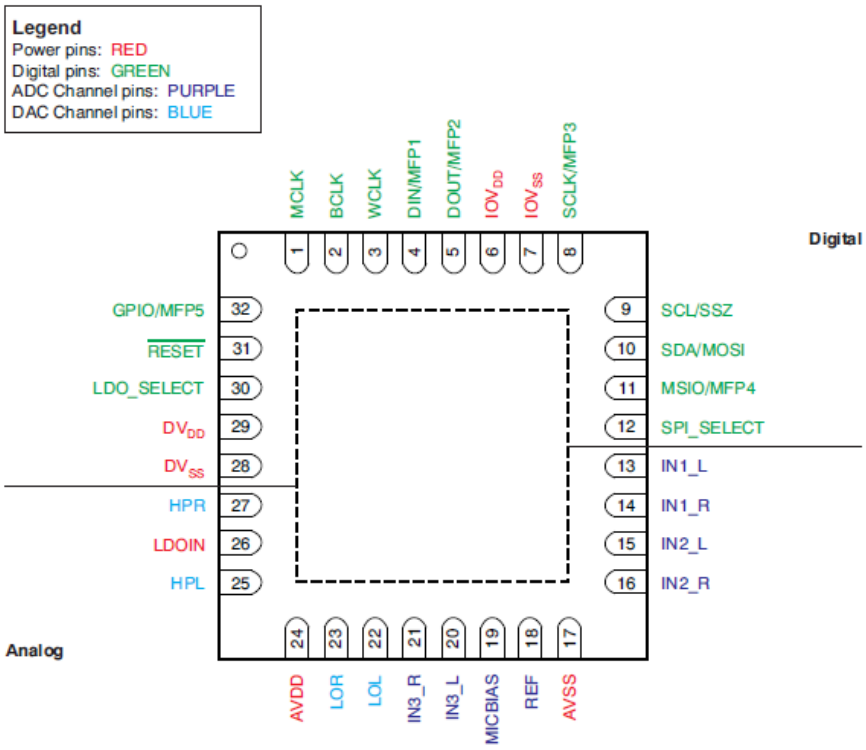
Tärkeimmät komponentit ovat AD-muunnin, DA-muunnin sekä dataväylä jolla siirretään muunnettua dataa codecin ja prosessorin välillä. Codec-piirin sisäinen kytkentä näkyy lohkokaaviosta (kuvio 4).



Kuvio 4: AIC3204 codecin lohkokaavio(Ultra Low Power Stereo Audio Codec 2008, 1)

5.1 Piirin nastajärjestys

Nastat on jaettu neljään eri ryhmään käyttötarkoituksen mukaan: käyttöjännite, AD-kanava ja DA-kanavan nastat. Suurin osa nastoista on ohjelmoitavissa useisiin eri käyttötarkoituksiin. Kuviossa 5 on esitetty piirin nastajärjestys sekä ryhmittely.



Kuvio 5: Codec-piirin nastajärjestys (Design and Configuration Guide for the TLV320AIC3204 2010, 3)

5.2 Powertune-teknologia

Piirille integroidun powertune teknologian tarkoitus on mahdollistaa teho-suorituskyvyn mahdollisimman helppo säätäminen. Ohjelmoija voi valita 4 erilaista powertune tilaa joilla säädetään AD- ja DA- muuntimen tehoresursseja.

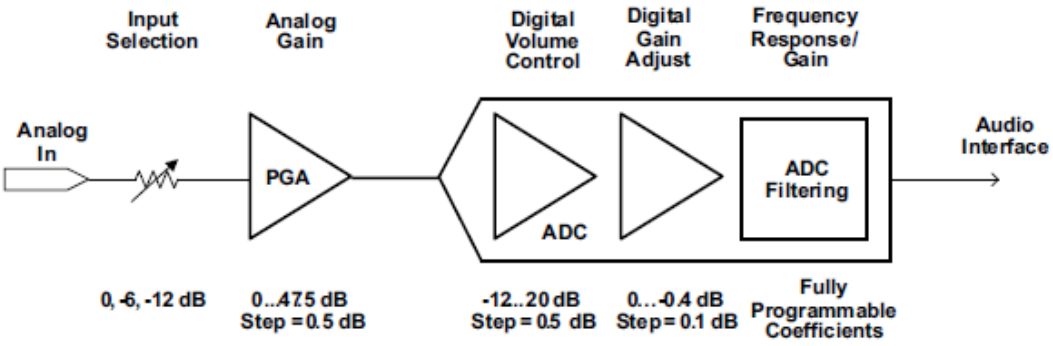
Powertune strategiaan kuuluu myös mahdollisuus valita mitä signaalinkäsittelyominaisuuksia käytetään ja mitkä sammutetaan tehon säästämiseksi. Ohjelmoinnin helpottamiseksi on kasattu valmiita signaalinkäsittelypaketteja, joista ohjelmoija voi valita sovellukseen sopivimman.

5.3 AD-muunnin

TLV320AIC3204 sisältää AD-muuntimen delta-sigma modulaattorilla ja desimointisuotimella. AD-muuntaja tukee näytteenottotaajuuksia 8kHz ja 192kHz väliltä. (Ultra Low Power Stereo Audio Codec 2008, 39)

AD-muuntimessa on kuusi analogisisääntuloa jotka ovat ohjelmoitavissa differentiaali tai single ended tilaan, sisääntulojen impedanssi on ohjelmoitavissa joko 10k, 20k tai 40k ohmiin. Jokainen kanava on myös erikseen ohjelmoitavissa päälle tai pois päältä sovelluksen tarpeen mukaan. (Ultra Low Power Stereo Audio Codec 2008, 39)

Piirillä on sisäänrakennettu ohjelmoitava vahvistuksen säätömahdollisuus(PGA), jolla on tarkoitus vahvistaa matalatasoisia signaaleja kuten mikrofonin signaalia. PGA:n avulla saadaan sisääntuloa vahvistettua impedanssista riippuen, single ended tilassa 0dB-47.5dB ja differentiaalisessa tilassa 6dB-53.5dB. Piirillä on myös erillinen äänenvoimakkuudensäätö joka toimii -12dB ja 20dB välillä 0,5dB porrastuksella. Jokaisen AD-kanavan vahvistusta voidaan lisäksi hienosäätää -0,5dB ja 0,5dB välillä 0,1dB porrastuksella, näin saadaan jokaiselle kanavalle tarkalleen samanlainen vahvistus. Kuviossa 5 esitetään AD-kanavan vahvistuksen säätömahdollisuudet. (Ultra Low Power Stereo Audio Codec 2008, 40-41)



Kuvio 6: AD kanavan vahvistuksen säätö (Ultra Low Power Stereo Audio Codec 2008, 40)

TLV320AIC3204-piiriin kuuluu automaattinen vahvistuksensäätöalgoritmi, jolla voidaan pitää kanavalta ulos tulevan äänenvoimakkuuden taso jatkuvasti samana, vaikka sisääntulevan signaalin voimakkuus muuttuisi. Kun käytetään AGC(automatic gain control)-algoritmia, käyttäjä määrittelee sen toimintaa parametrien avulla. (Ultra Low Power Stereo Audio Codec 2008, 41-42)

AD-kanavaan kuuluu 3 erityyppistä(a,b ja c) sisäänrakennettua desimointisuodinta, joilla lasketaan delta-sigma modulaattorilta tulevan datan näytteenottotaajuutta. Desimointisuodin määräytyy halutun taajuusvasteen, viiveen ja näytteenottotaajuuden mukaan, käyttäjä ei siis voi valita haluamaansa suodinta, vaan se määräytyy

automaattisesti. AD-muuntimessa on myös erilaisia suotimia datan muokkaamista varten. (Ultra Low Power Stereo Audio Codec 2008, 50)

Ensimmäisen asteen IIR-suotimella voidaan helposti poistaa signaalista mahdollinen DC-komponentti. IIR-suotimen lisäksi voidaan käyttää vaihtoehtoisesti joko maksimissaan viittä biquad suodinta, tai maksimissaan 25 asteista FIR-suodinta. (Ultra Low Power Stereo Audio Codec 2008, 48)

5.4 DA-muunnin

TLV320AIC3204 codec-piiriin kuuluu DA-muunnin joka tukee näytteenotto taajuuksia 8kHz ja 192kHz välillä. Jokainen DA-kanava sisältää digitaalisen interpolointi suotimen, delta-sigma modulaattorin ja analogisen signaalin muodostus suotimen. (Ultra Low Power Stereo Audio Codec 2008, 60)

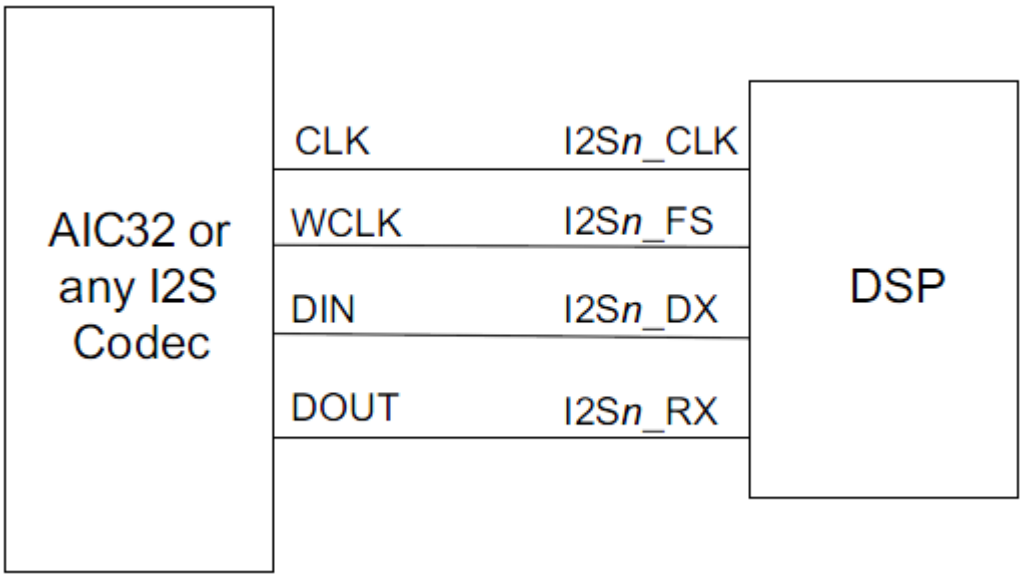
DA-muuntimessa on signaalinkäsittelyä varten AD-muuntimen tavoin IIR-suodin ja biquadsuotimia, lisäksi on mahdollisuus luoda signaaliin 3D-efekti tai tuottaa ääntä piippaus generaattorin avulla. (Ultra Low Power Stereo Audio Codec 2008, 61)

AD-kanavan tavoin myös DA-kanavassa on mahdollista automaattisesti säätää signaalin voimakkuutta käyttäen DRC(dynamic gain control) algoritmia joka jatkuvasti tarkkailee ulostulevan signaalin voimakkuutta ja tarvittaessa joko vahvistaa tai heikentää sitä käyttäjän toiveiden mukaisesti. (Ultra Low Power Stereo Audio Codec 2008, 74)

DA-muunnin sisältää äänenvoimakkuuden säätimen jolla voidaan säätää sekä oikean että vasemman ulostulo signaalin voimakkuutta -63,5 dB ja +24dB välillä 0,5dB askelein, joko yhdessä tai erikseen. (Ultra Low Power Stereo Audio Codec 2008, 73-74)

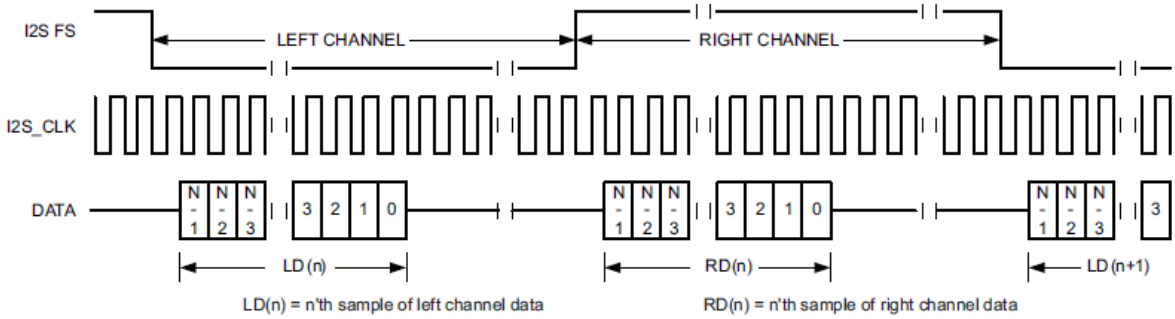
5.5 I2S-tiedonsiirtoväylä

I2S on Philipsin sarjamuotoinen full dublex tiedonsiirtoväylä, joka on erityisesti tarkoitettu audiodatan suoratoistoon, signaalinkäsittely prosessorin ja I2S lisälaitteen välillä. I2S-väylä koostuu kahdesta kellopulssista sekä kahdesta datalinjasta (kuvio7). (TMS320VC5505 DSP Inter-Integrated Circuit 2009, 9)



Kuvio 7: I2S-väylän kytkentä (TMS320VC5505 DSP Inter-Integrated Circuit 2009, 13)

Toinen kello (bit clock) antaa pulssin jokaisen lähetetyn databitin kohdalla, jolloin kellotaajuuden tulee olla näytteenottotaajuus kerrottuna siirretyn datan sananpituudella. Koska I2S mahdollistaa kahden kanavan lähetyksen yhdellä linjalla, tarvitaan toinen kello (word clock) määräämään kumman kanavan dataa lähetetään milloinkin. Word clockin taajuus määräytyy suoraan näytteenotto-taajuuden mukaan. Kuviossa 8 näkyy väylän ajoituskaavio. (TMS320VC5505 DSP Inter-Integrated Circuit 2009, 15)



Kuvio 8: I2S-kanavan ajoitus kaavio (TMS320VC5505 DSP Inter-Integrated Circuit 2009, 16)

6 Esimerkkiprojektit

Kehitysalustan käytön demonstroimiseksi tehtiin kaksi esimerkkiprojektia, tässä kappaleessa kerrotaan projektien taustat ja selvitetään sovellusten toimintaa. Lähdekoodista käydään läpi vain oleellisia kohtia, koko lähdekoodi on luettavissa liitteistä.

Kummassakin projektissa käytettiin apuna asennus-dvdllä olevia Ezdsp-kortille suunniteltuja kirjastoja. Lisäksi apuna käytettiin kahta Texas Instruments'n esimerkeissä käytettyä funktiota.

```
Int16 AIC3204_rset( Uint16 regnum, Uint16 regval )
{
    Uint8 cmd[2];
    cmd[0] = regnum & 0x007F;
    cmd[1] = regval;

    return USBSTK5505_I2C_write( AIC3204_I2C_ADDR, cmd, 2 );
}
```

Piirin ohjelmoinnissa käytetylle funtiolla lähetetään parametreinä sen rekisterin numero johon halutaan kirjoittaa ja arvo joka rekisteriin kirjoitetaan. Funktion sisällä muodostetaan viesti, jossa ilmoitetaan codecin osoite, rekisterin numero, arvo joka kirjoitetaan sekä kirjoitus käsky jotta codec tietää että kyseessä on kirjoitus toimenpide. Nämä tiedot lähetetään codecille valmiilla I2C-kirjoitusfunktiolla.

```
Int16 AIC3204_rget(  Uint16 regnum, Uint16* regval )
{
    Int16 retcode = 0;
    Uint8 cmd[2];

    cmd[0] = regnum & 0x007F;
    cmd[1] = 0;

    retcode |= USBSTK5505_I2C_write( AIC3204_I2C_ADDR, cmd, 1 );
    retcode |= USBSTK5505_I2C_read( AIC3204_I2C_ADDR, cmd, 1 );

    *regval = cmd[0];
    USBSTK5505_wait( 10 );
    return retcode;}
```

Toista valmisfunktiota käytettiin codecin rekistereiden lukemiseen. Funktiolle annetaan kaksi parametria, rekisteri jonka sisältö halutaan tietää ja sen muuttujan osoite mihin rekisterin arvo sijoitettaan. Funktio sijoittaa rekisterin sisällön haluttuun muuttujaan. Lippujen nollaamisesta ei tarvitse huolehtia, sillä codec suorittaa nollauksen automaattisesti aina sen jälkeen kun lipun arvo luetaan.

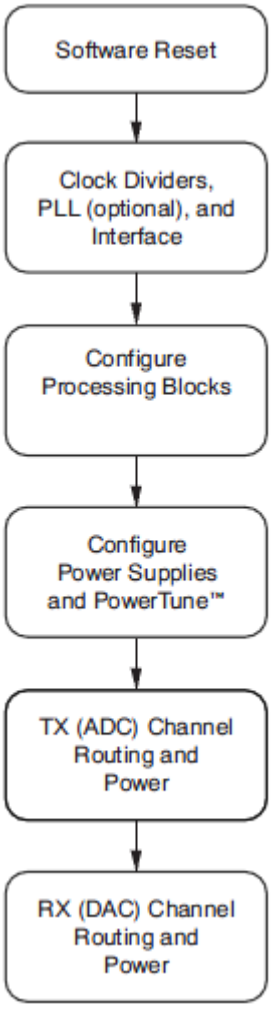
6.1 Näytteenottotaajuuden vaikutus äänen laatuun

Työssä havainnollistetaan näytteenottotaajuuden vaikutusta äänen laatuun. Aluksi syötetään musiikkia codec-piirin sisääntuloon josta tehdään AD-muunnos 48kHz näytteenottotaajuutta käyttäen. AD-muuntimelta saatu data lähetetään prosessorille I2S-väylää pitkin. Prosessori lähettää digitaalisen datan takaisin codec-piirille, mikä tekee DA-muunnoksen käyttäjän haluamalla näytteenotto taajuudella ja laittaa saadun analogisen signaalin ulostuloon käyttäjän kuultavaksi.

Käyttäjä voi säätää DA-muunoksessa käytettyä näytteenottotaajuutta painonapeilla 48kHz ja 50 Hz välillä ja samalla kuunnella kuulokkeista kuinka äänen laatu muuttuu.

6.1.1 Codec-piirin ohjelmointi

Piiri ohjelmoidaan muuttelemalla sen 8-bittisten rekistereiden arvoja, kirjoittamiseen voidaan käyttää joko SPI- tai I2C-väylää. Laiteen ohjelmointi pitää suorittaa oikeassa järjestyksessa oikean toiminnan takaamiseksi (kuvio 9). Kaikkia ohjelmoinnin vaiheita ei tosin ole pakko suorittaa mikäli resetissä asetetut oletusarvot ovat sovellukseen sopivia.



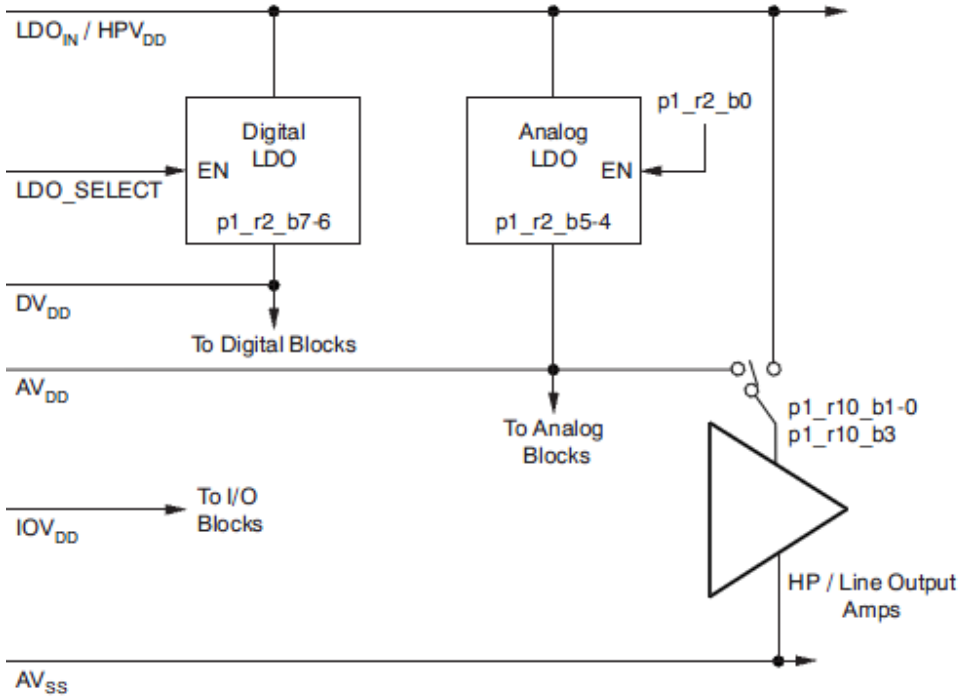
Kuvio 9: Codecin ohjelmointijärjestys (Design and Configuration Guide for the TLV320AIC3204 2010, 16)

```
/* Codec-piirin asetukset */

    AIC3204_rset( 0, 0 );      // valitaan sivu 0
    AIC3204_rset( 1, 1 );      // resetoidaan codec ohjelmallisesti

    AIC3204_rset( 0, 1 );      // valitaan sivu 1
    AIC3204_rset( 1, 8 );      // poistetaan AVDD ja DVDD yhteys
    AIC3204_rset( 2, 1 );      // Analogilohko käyttöön AVDD päälle
                               // jännite 1,72V
    AIC3204_rset( 0, 0 );      // valitaan sivu 0
```

Ennen kuin voidaan aloittaa varsinainen ohjelmointi, piiri pitää alustaa jotta varmistutaan siitä, että piiri on oletustilassa. Alustus tapahtuu resetoimalla piiri, tämä voidaan tehdä joko ohjelmallisesti kirjoittamalla 1-bitti rekisteriin 1, tai vetämällä piirin

reset pinni alas vähintään 10 ns ajaksi. Piirin rekistereihin ei saa kirjoittaa 1 ms resetin jälkeen.

Käyttöjännite syötetään piirille LDOin nastasta, otetaan käyttöön analog LDO ja käytetään sitä piirin käyttöjännitteen muodostamiseen. DVdd ja AVdd on oletuksena yhdistetty, tämä yhteys pitää katkaista ohjelmallisesti mikäli analog LDO on käytössä. Jännitelähteen lohkokaavio näkyy kuviossa 10.



Kuvio 10 Jännitelähteen lohkokaavio (Design and Configuration Guide for the TLV320AIC3204 2010, 9)

```
/* PLL ja kellojen asetus/käynnistys  */

    AIC3204_rset( 27, 0x1d );  // BCLK and WCLK ulostuloksi
    AIC3204_rset( 30, 0x88 );  //  BCLK=DAC_CLK/N =(12288000/8)=
                               //1.536MHz = 32*fs
    AIC3204_rset( 28, 0x00 );  // Data offset = 0
```

I2S-väylän tarvitsemat asetukset. BCLK ja WCLK pinni asetetaan ulostuloiksi jotta niitä voidaan käyttää I2S-väylän kelloina. WCLK määräytyy halutun näytteenottotaajuuden mukaan. BCLK pitää olla 32*näytteenottotaajuus, sopivan taajuuden säätämiseen voidaan käyttää N-jakajaa. Data offset asetetaan nollaan.

```
AIC3204_rset( 4, 3 );      //PLL lähteeksi MCLK,PLL codec kelloksi
AIC3204_rset( 6, 8 );      // PLL kertoja J=8
AIC3204_rset( 7, 15 );     // PLL kertoja D HI_BYTE(D)
AIC3204_rset( 8, 0xdc );   // PLL kertoja D LO_BYTE(D)
AIC3204_rset( 5, 0x91 );   //PLL käyntii P ja R kertoja = 0
AIC3204_rset( 13, 0 );     // Hi_Byte(DOSR) DOSR = 128
AIC3204_rset( 14, 0x80 );  // Lo_Byte(DOSR) DOSR = 128
AIC3204_rset( 20, 0x80 );  // AOSR = 128
```

Codecin kellolähde voidaan tuota piirille monella eri tavalla, joko MCLK, BCLK, GPIO tai din nastan kautta. Mikäli sopivaan ulkoista kelloa ei ole tarjolla voidaan käyttää PLL:ä ja esijakajia sopivan taajuuden muodostamiseen. Kehitysalustalla on codec-piirille oma ulkoinen 12MHz kide joka on kytketty MCLK pinniin, ohjelmassa käytetään MCLK pinniä PLL lähteenä. 12MHz kello kerrotaan 8.406:lla ja näin saadaan kelloksi 100MHz. PLL:n P ja R kertojia ei tarvita joten ne asetetaan nollaan. Sekä AD-, että DA-kanavan ylinäytteistys arvoksi asetetaan 128.

```
AIC3204_rset( 11, jakaja1 );  // NDAC päälle, käyttäjä säätää
AIC3204_rset( 12, jakaja2 );  // MDAC päälle, käyttäjä säätää
AIC3204_rset( 18, 0x88 );     // NADC päälle = 8
AIC3204_rset( 19, 0x82 );     // MADC päälle = 2
```

Kun ollaan päätetty mitä ylinäytteistys arvoa käytetään(AOSR ja DOSR), voidaan AD- ja DA-muunnoksen näytteenottotaajuus asetetaan sopivaksi jakajilla, joita käyttäjä pystyy muuttamaan painonapeilla.

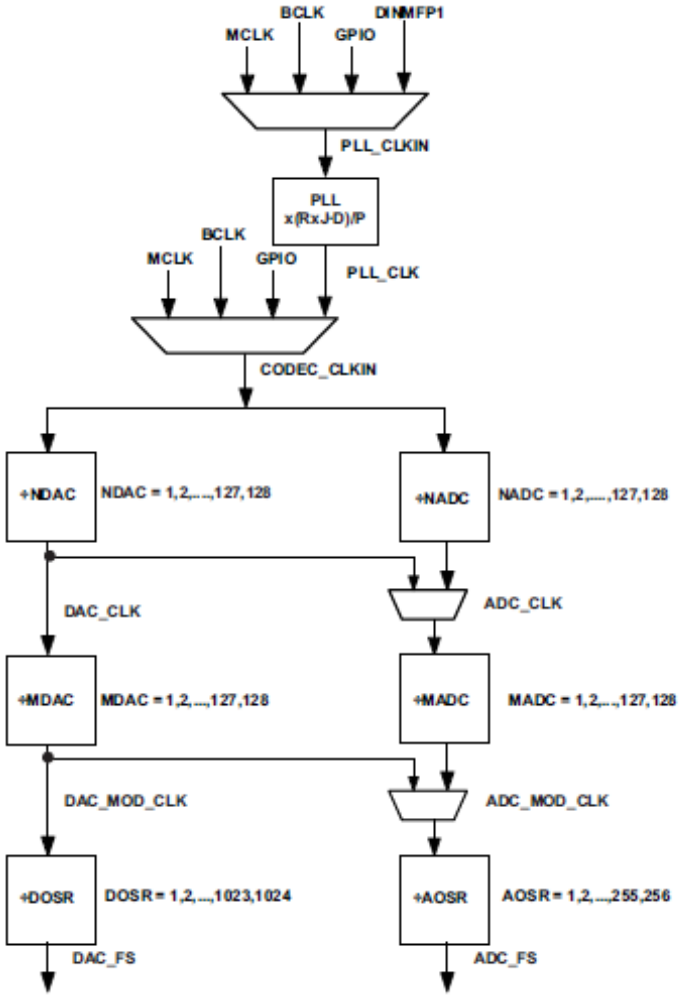
AD-muuntimen näytteenottotaajuus (Fs) lasketaan yhtälön (1) mukaisesti.

$$Fs = \frac{ADC\_clk}{AOSR*MADC*NADC} \quad (1)$$

DA-muuntimen näytteenottotaajuus (Fs) lasketaan yhtälön (2) mukaisesti.

$$Fs = \frac{DAC\_clk}{DOSR*MDAC*NDAC} \quad (2)$$

Codec-piirin kellosisääntulojen ja jakajien väliset yhteydet näkyvät kuviossa 11.



Kuvio 11 Kellonmuodostuspuu (Ultra Low Power Stereo Audio Codec 2008, 80)

```
/* DAC reititys ja käynnistys */
    AIC3204_rset( 0, 1 );       // valitaan sivu 1
    AIC3204_rset( 0x0c, 8 );    // LDAC=>vasen kuuloke
    AIC3204_rset( 0x0d, 8 );    // RDAC=>oikea kuuloke
    AIC3204_rset( 0, 0 );       // valitaan sivu 0
    AIC3204_rset( 64, 2 );      // vas volume=oik volume
    AIC3204_rset( 65, 0 );      // vasen DAC vahvistus 0dB
    AIC3204_rset( 63, 0xd4 );   // käynnistetään oik ja vas DAC
    AIC3204_rset( 0, 1 );       // Valitaan sivu 1
    AIC3204_rset( 0x10, 10 );   // äänet päälle vas kuuloke, +10dB
    AIC3204_rset( 0x11, 10 );   // äänet päälle oik kuuloke, +10dB
    AIC3204_rset( 9, 0x30 );    // virrat päälle oik ja vas kuuloke
    AIC3204_rset( 0, 0 );       // valitaan sivu 0
    USBSTK5505_wait( 100 );     // viive
```

DA-kanava reititetään siten, että ulostuloksi valitaan nastat johon stereojakki on kytketty. Asetetaan vahvistukset sopivaksi ja laitetaan kuulokkeisiin virrat päälle. Viive varmistaa sen, että DA-kanavaa ei käytetä ennenkuin kaikki asetukset ovat tulleet voimaan.

```
/* ADC reititys ja käynnistys */
AIC3204_rset( 0, 1 );      // Valitaan sivu 0
AIC3204_rset( 0x34, 0x30 );// STEREO 1 Jack vas sisääntulo
                           //impedanssi  40kohm micpga +nasta

AIC3204_rset( 0x37, 0x30 );// STEREO 1 Jack oik sisääntulo
                           //impedanssi  40kohm  micpga +nasta

AIC3204_rset( 0x36, 3 );   //vas kanavan micpga -nasta 40 kohm
                           //läpi maihin


AIC3204_rset( 0x39, 0xc0 );// oik kanavat micpga -nasta 40 kohm
                           //läpi maihin


AIC3204_rset( 0x3b, 0 );   // PGA_L äänet päälle
AIC3204_rset( 0x3c, 0 );   // PGA_R äänet päälle
AIC3204_rset( 0, 0 );      // valitaan sivu 0
AIC3204_rset( 0x51, 0xc0 );// virrat päälle vas ja oik ADC
AIC3204_rset( 0x52, 0 );   // äänet päälle vas ja oik ADC

AIC3204_rset( 0, 0 );
```

Lopuksi valitaan AD-kanavan sisääntuloksi stereo in nastat, asetetaan sisääntulo impedanssiksi 40kohm ja käynnistetään AD-kanava. Tämän jälkeen codec-piiri on käyttövalmis.

6.1.2 Prosessorin asetukset ja pääohjelma

```
SYS_EXBUSSEL |= 0x0100;  // Sarjaväylä 0 I2S0 protokollaksi
/* I2S settings */
I2S0_SRGR = 0x0;
I2S0_CR = 0x8010; //I2S sananpituus 16-bit, orja mode, väylä
käyttöön
```

```
I2S0_ICMR = 0x3f;    // sallitaan keskeytykset
```

Jotta audiodatan siirto codecin ja prosessorin välillä onnistuu, valitaan sarjaliikenne protokollaksi I2S. Seuraavilla riveillä asetetaan sanan pituus 16-bittiin, asetetaan I2S orjatilaan, jolloin prosessorin I2Sn_CLK ja I2Sn_FS asettuvat sisääntuloiksi ja väylän kellot tulevat codecilta. Lopuksi käynnistetään väylä ja sallitaan keskeytykset.

```
/* luetaan codecin ADCltä tulevaa dataa */

// odotetaan että vastaanoton keskeytyslippu laskee
while((RcvR & I2S0_IR) == 0);
// vasemman kanavan 16-bit data muuttujaan
data3 = I2S0_W0_MSW_R;
// oikean kanavan 16-bit data muuttujaan
data4 = I2S0_W1_MSW_R;

/* lähetetään data takaisin codecille */

// odotetaan että lähetys keskeytyslippu laskee
while((XmitR & I2S0_IR) == 0);
// vasemman kanavan 16-bit data muuttujasta lähetysrekisteriin
I2S0_W0_MSW_W = data3;
// oikean kanavan 16-bit data muuttujasta lähetysrekisteriin
I2S0_W1_MSW_W = data4;
```

Pääohjelman puolella luetaan I2S-väylältä tulevaa dataa muuttujaan I2S-väylän vastaanotto rekisteristä ja samantien lähetetään data I2S-väylää pitkin codec-piirille sijoittamalla muuttujan arvo I2S-väylän lähetys rekisteriin. I2S-väylän kirjoitus ja lukutoimenpiteen valmistumista tarkkaillaan keskeytyslipun avulla, kun lippu nousee pystyyn, on edellinen toimenpide suoritettu ja voidaan aloittaa uusi.

6.1.3 Testaus

Testaus toteutettiin kuuntelemalla musiikkia, joka kulki kehitysalustan läpi. AD- ja DA-kanavan näytteenottotaajuutta muunneltiin ajon aikana nappia painamalla ja samalla tehtiin havaintoja äänenlaadusta. Äänenlaadun muutoksien havainnoinnissa luotettiin kuuloon. Näytteenottotaajuus varmistettiin mittaamalla codec-piirin jalasta I2S-väylän käyttämää word clockia, joka asettuu automaattisesti samaksi kuin näytteenottotaajuus.

Sovellus toimi hyvin 48kHz – 5kHz näytteenottotaajuuksien välillä ja muutokset äänen laadussa oli selvästi havaittavissa. Kun näytteenottotaajuus laskettiin alle 5kHz,

ilmestyi ääneen ylimääräistä kohinaa, jonka alkuperää ei saatu selville. Valmistajan ilmoittama alin tuettu näytteenottotaajuus on 8kHz, joten on mahdollista että ylimääräinen kohina generoitui ääneen codec-piirin AD- ja DA-muuntajista.

6.2 Aktiiviset kuulosuojaimet

Työn tarkoitus on rakentaa prototyyppi aktiivisista kuulosuojaimista. Aktiiviset kuulosuojaimet toimivat siten, että ne vahvistavat heikkoja ääniä ja sulkevat pois vaarallisen kovat äänet. Näin saadaan kuulo suojattua ilman että havainnointi kyky heikkenee. Työ toteutetaan siten, että kuulosuojaimien ulkopuolelle kytketään elektreettimikrofoni joka välittää äänen kehitysalustalle. Kehitysalustalla vahvistetaan mikrofonilta tuleva signaali ja samalla tarkkaillaan sisääntulevan äänen tasoa ja säädetään sitä AGC-algoritmilla. Kovat impulssiäänet joita AGC ei pysty vaimentamaan poistetaan manuaalisesti koodissa.

Elektreetti on mahdollista kytkeä suoraan kiinni codec-piirin ohjelmoitavaan biasointinastaan, mutta koska tätä mahdollisuutta ei olla otettu huomioon kehitysalustaa suuniteltaessa jouduttiin biasointijännite tuomaan mikrofonille erikseen.

6.2.1 AGC-algoritmin asetukset

```
/*  AGC parametrien asetus*/

//Vasen Target Level päälle, -24dBFS. Hystereesi 1.5db
AIC3204_rset(86, 0xF3  );
//Oikea Target Level päälle, -24dBFS. Hystereesi 1.5db
AIC3204_rset(94, 0xF3  );
```

Target levelillä määritellään ulostulosignaalin taso, johon AGC pyrkii. DBFS tarkoittaa, että annettua signaalin voimakkuutta verrataa maksimi ulostulosignaalin voimakkuuteen. Hystereesi taas luo halutun tason ympärille ikkunan joka antaa algoritmille pelivaraa, näin vältytään jatkuvalta signaalin säätämiseltä, joka saattaa aiheuttaa signaalin säröytymistä.

```
//Vasen noise treshold -30dB gain hystereesi 4dB
AIC3204_rset(87, 0x82 );
//Oikea noise treshold -30dB gain hystereesi 4dB
AIC3204_rset(95, 0x82  );
```

Noise tresholdin avulla voidaan ehkäistä tehokkaasti kohinan pääsemistä kaiuttimiin. Kaikki signaalit joiden voimakkuus jää alle noise treshold tason, tulkitaan kohinaksi eikä niitä vahvisteta. Noise treshold arvoa verrataan asetettuun target leveliin, myös noise tresholdiin on mahdollisuus asettaa hystereesiä.

```
//Vasen AGC Maximum Gain Setting 40db
AIC3204_rset(88, 80  );
//Oikea AGC Maximum Gain Setting 40db
AIC3204_rset(96, 80  );
```

AGC pystyy vahvistamaan signaalia 0-63.5dB mutta käyttäjä voi halutessaan rajoittaa vahvistusta.

```
//Vasen AGC Attack Time = 11*32 word clock
//Attack Time Scale Factor = 1
AIC3204_rset(89, 0x28  );
//Oikea AGC Attack Time = 11*32 word clock
//Attack Time Scale Factor = 1
AIC3204_rset(97, 0x28  );

//Vasen AGC Decay Time = 11*512 ADC Word Clocks
//Decay Time Scale Factor = 1
AIC3204_rset(90, 0x28  );
//Oikea AGC Decay Time = 11*512 ADC Word Clocks
//Decay Time Scale Factor = 1
AIC3204_rset(98, 0x28  );
```

Attack ja decay time määrittelee kuinka nopeasti signaalin voimakkuutta nostetaan/lasketaan, kun ei saavuteta haluttua tasoa.

```
//vasen noise debounce 16 adc word clock
AIC3204_rset(91, 4  );
//oik noise debounce 16 adc word clock
AIC3204_rset(99, 4  );

//vasen signal debounce 16 adc word clock
AIC3204_rset(92, 4 );
//oik signal debounce 16 adc word clock
AIC3204_rset(100, 4  );
```

Debounce parametreillä määritellään ajanjakso jona signaalin tasoa tarkkailleen, jos aika on pitkä, se hidastaa AGC:n reagointia signaalin muutoksiin, mutta liian lyhyt ajanjakso taas saattaa aiheuttaa äänen säröytymistä.

6.2.2 Pääohjelma

```
/* luetaan codecin ADCltä tulevaa dataa */
// odotettaan että vastaanoton keskeytyslippu laskee
while((RcvR & I2S0_IR) == 0);
data3 = I2S0_W0_MSW_R;  // vasemman kanavan 16-bit data muuttujaan
data4 = I2S0_W1_MSW_R;  // oikean kanavan 16-bit data muuttujaan
//jos näytteen arvo on liian korkea tulkitaan kuulolle vaaralliseksi
//ja nostetaan lippu
if(data3 > 2000 ||data4 > 2000 )
{
  ohi = 1;
  ohi_laskuri = 100000; //laskee 100000*4us=4sec
}
else if(ohi_laskuri == 0 )
{
  //jos ääni ollut alle asetetun arvon tarpeeksi kauan
  //lasketaan lippu
  ohi = 0;
}
//jos lippu ylhäällä, ei lähetetä ääntä takaisin codecille
if(ohi == 0)
{
   /* Write Digital audio input */
   // odotettaan että lähetyksen keskeytyslippu laskee
   while((XmitR & I2S0_IR) == 0);
   I2S0_W0_MSW_W = data3;//vasemman kanavan data lähetysrekisteriin
   I2S0_W1_MSW_W = data4;// oikean kanavan data lähetysrekisteriin
}
//vähennetään ja pidetään huoli ettei laskuri mene ympäri
if(ohi_laskuri > 0)
{
   ohi_laskuri--;
}
```

Pääohjelman tärkein tehtävä on huolehtia, että ne voimakkaat äänet (yli 95db) joita AGC ei pysty vaimentamaan riittävästi eivät päädy kuulokkeisiin. Näin

äänenvoimakkuus kuulokkeiden sisällä pysyy jatkuvasti alle 81db, jolloin kuulovaurion vaaraa ei ole.

6.2.3 Testaus

Kuulosuojaimien testiympäristöön kuului koekytkentäalustalla oleva elektreettimikrofoni, jonka signaali vietiin kehitysalustalle. Kehitysalustalta ääni siirtyi nappikuulokkeiden avulla Peltor-kuulosuojaimien sisälle. Äänentasoa mitattiin desibelimittarin avulla.

Testissä kiinnitettiin huomiota äänenlaatuun, AGC-algoritmin kykyyn vaimentaa/vahvistaa signaalia tarvittaessa ja ohjelmallisesti toteutettuun impulssiäänien poistoon.

Ensimmäisenä huomio kiinnittyi äänenlaatuun, jossa ongelmana oli häiritsevä kohina. Suurin osa kohinasta tulee codec-piirin vahvistimesta. Testeissä havaittiin, että jos elektreettimikrofonin signaali esivahvistettiin erillisellä operaatiovahvistin kytkennällä ennen kehitysalustaa, päästiin kohinasta eroon. Erillisestä esivahvistimesta päätettiin kuitenkin luopua, koska tarkoituksena oli toteuttaa koko sovellus vain kehitysalustan ominaisuuksia käyttäen. Mikäli sovellusta lähdettäisiin jatkokehittämään, tulisi kohina poistaa esimerkiksi codec-piirin suotimilla tai ohjelmallisesti toteutetulla digitaalisella suotimella.

Seuraavana vuorossa oli impulssiäänen poisto-ominaisuus, jota testattiin generoimalla kovia ääniä, jotka kuulosuojaimen tulisi osata poistaa. Tämä sovelluksen osa toimi todella hyvin. Sovellus osasi poistaa kaikki halutun äänentason ylittävät äänet ja äänentasoa oli helppo säätää sopivaksi. Testauksessa käytettiin useita eri äänenvoimakkuuksia (50-95db), mutta todella kovia ääniä ei voitu testiolosuhteiden takia käyttää.

Viimeisenä kokeiltiin AGC-algoritmin toimintaa. Normaalit äänet kuuluivat kuulokkeisiin hyvin, mutta kohinan takia todella heikkoja ääniä ei voitu vahvistaa. Algoritmi osasi vaimentaa asetetun tason ylittäviä ääniä maksimissaan 15db. Tämä tarkoittaa sitä, että kun äänen voimakkuus kuulokkeiden ulkopuolella oli 95db niin, kuulokkeiden sisälle se oli 80db. Taso jolla algoritmi pyrki äänenvoimakkuutta pitämään, riippui ulkopuolella olevasta äänestä jolloin oli mahdollista erottaa äänenvoimakkuuden vaihtelut.

7 Yhteenveto

Markkinoilla on tällä hetkellä kohtuullisen vähän kehitysalustoja, jotka ovat tarkoitettu digitaaliseen signaalinkäsittelyyn ja useasti ne ovat liian kalliita harrastajan budjettiin. Texas Instruments’n tarkoituksena on ollut tuoda markkinoille laite, jolla kuka tahansa voi aloittaa sovelluksien suunnittelun.

Laitteen käyttöönotto on kohtuullisen helppoa ja siihen on olemassa hyvät ohjeet. Valmiin esimerkin päälle on helppo kehitellä omia sovelluksia. Prosessorille on luotu myös valmiita kirjastoja, joiden tarkoituksena on helpottaa ja nopeuttaa sovelluksien kehittämistä. Verrattuna moniin muihin kehitysalustoihin, valmiita esimerkkejä on kuitenkin tarjolla todella vähän, jolloin datalehtien lukutaidon tarpeellisuus korostuu.

Codec-piiri on kehitysalustalla mielenkiintoinen lisä ja yllätti monipuolisuudellaan. Codec-piirin käyttö on kuitenkin rajoittunutta koska sen ominaisuuksia ei ole otettu huomioon kehitysalustan suunnittelussa.

Esimerkkiprojektien tarkoituksena on esitellä kehitysalustan ominaisuuksia ja mahdollisia käyttökohteita. Projekteissa pyrittiin siirtämään mahdollisimman suuri työmäärä codec-piirille jolloin prosessorin laskenta-aikaa säästyy muihin tehtäviin tai vaihtoehtoisesti prosessorin käyttöjännitettä voidaan laskea jolloin virrankulutus pienenee.

Lähteet

Design and Configuration Guide for the TLV320AIC3204.[pdf][viitattu 20.7.2010].
http://focus.ti.com/lit/an/slaa404c/slaa404c.pdf

Texas Instruments. Code Composer Studio Ide. [online][viitattu 20.7.2010].
http://focus.ti.com/dsp/docs/dspsupportatn.tsp?sectionId=3&tabId=415&familyId=44&toolTypeId=30

TMS320VC5505 DSP Inter-Integrated Circuit.[pdf][viitattu 20.7.2010].
http://focus.ti.com/lit/ug/sprufo1a/sprufo1a.pdf

TMS320VC5505 DSP System user's guide.[pdf][viitattu 20.7.2010].
http://focus.ti.com/docs/prod/folders/print/tms320vc5505.html

TMS320VC5505 eZdsp Technical Reference.[pdf][viitattu 20.7.2010].
http://support.spectrumdigital.com/boards/usbstk5505/revb/files/usbstk5505_TechRef_revb.pdf

TMS320VC5505 eZdsp USB Stick Hands-on Training.[video][viitattu 20.7.2010]
http://seminar2.techonline.com/~texasinstruments/TI_ESCBoston2009/TI_noncoupon.html

TMS320VC5505 Fixed-Point Digital Signal Processor.[pdf][viitattu 20.7.2010].
http://focus.ti.com/docs/prod/folders/print/tms320vc5505.html

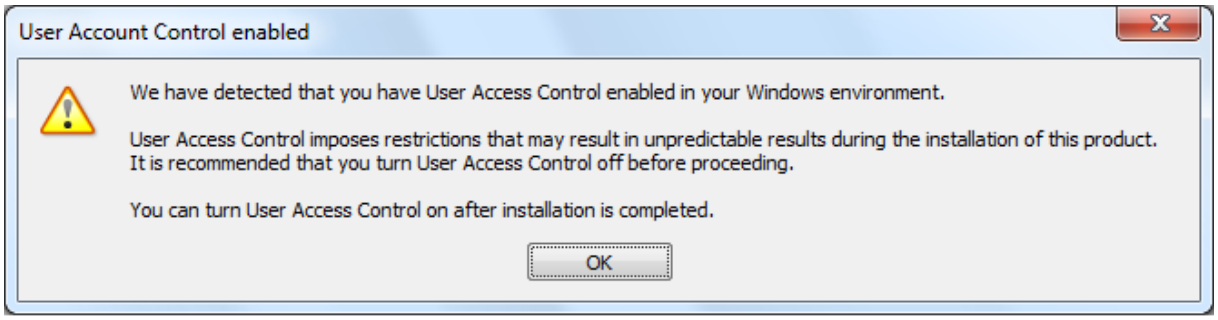
Ultra Low Power Stereo Audio Codec.[pdf][viitattu 20.7.2010].
http://focus.ti.com/lit/ds/symlink/tlv320aic3204.pdf

Liitteet

Liite1: Kehitysalustan käyttöönotto-opas

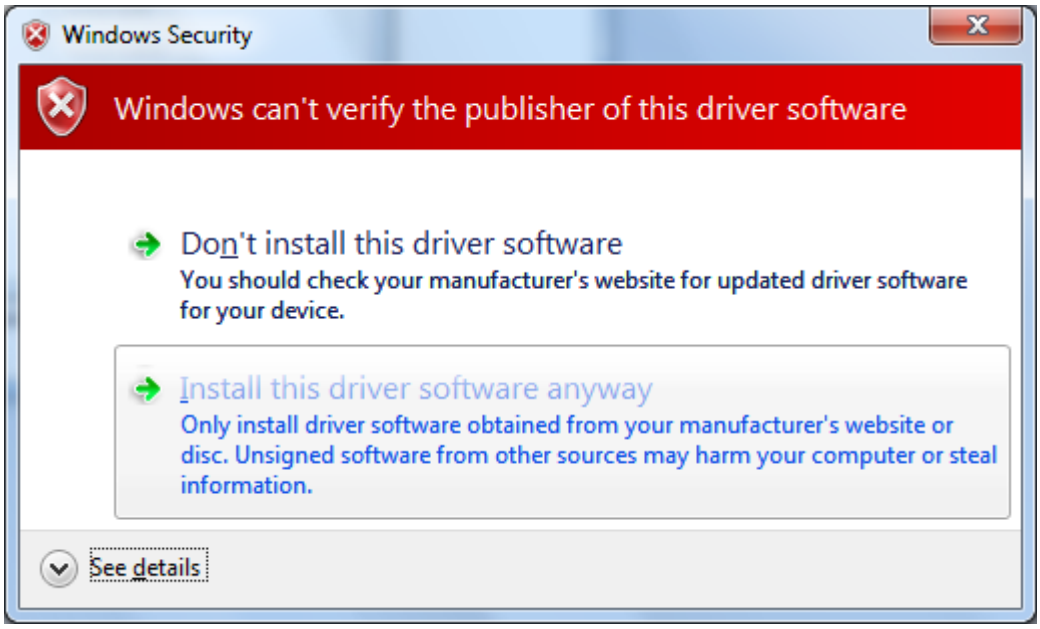
Aluksi asennetaan CCS4 koneelle:

Huom! Ennen CCS4 asennusta tulee varmistaa että käyttäjällä on järjestelmänvalvojan oikeudet, virustorjunta ohjelmat ja käyttäjätilien valvonta ovat pois päältä. Mikäli käyttäjätilien valvonta on päällä, tulee asenuksen alussa varoitus (Kuvio12).



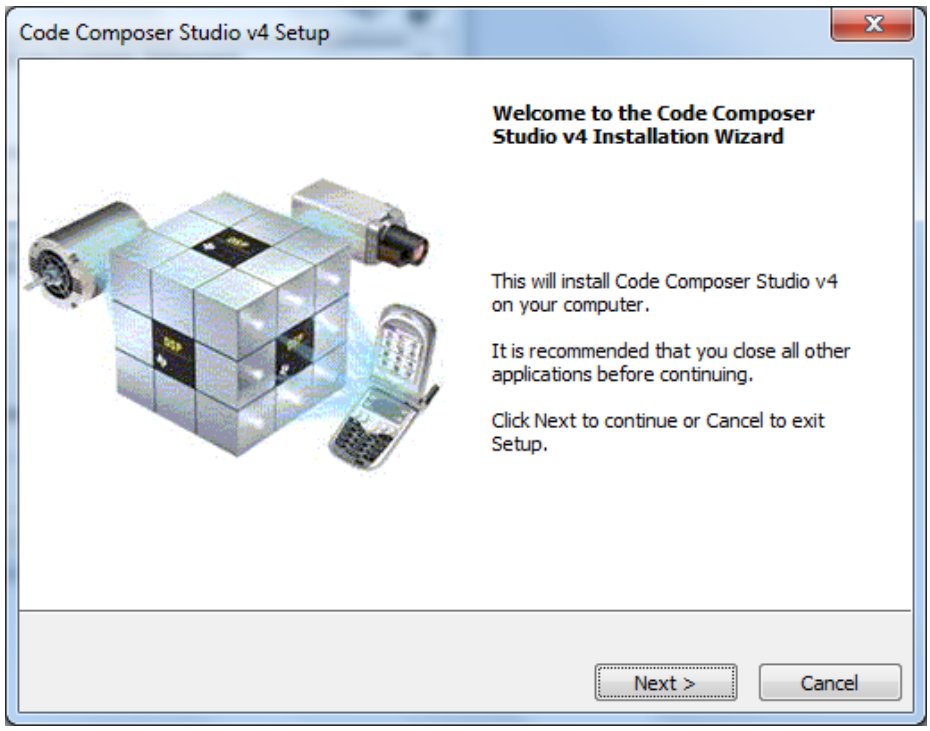
Kuvio 12

Jos käyttäjätilien valvontaa ei poisteta käytösta voi windows antaa ylimääräisiä varoituksia (Kuvio 13).



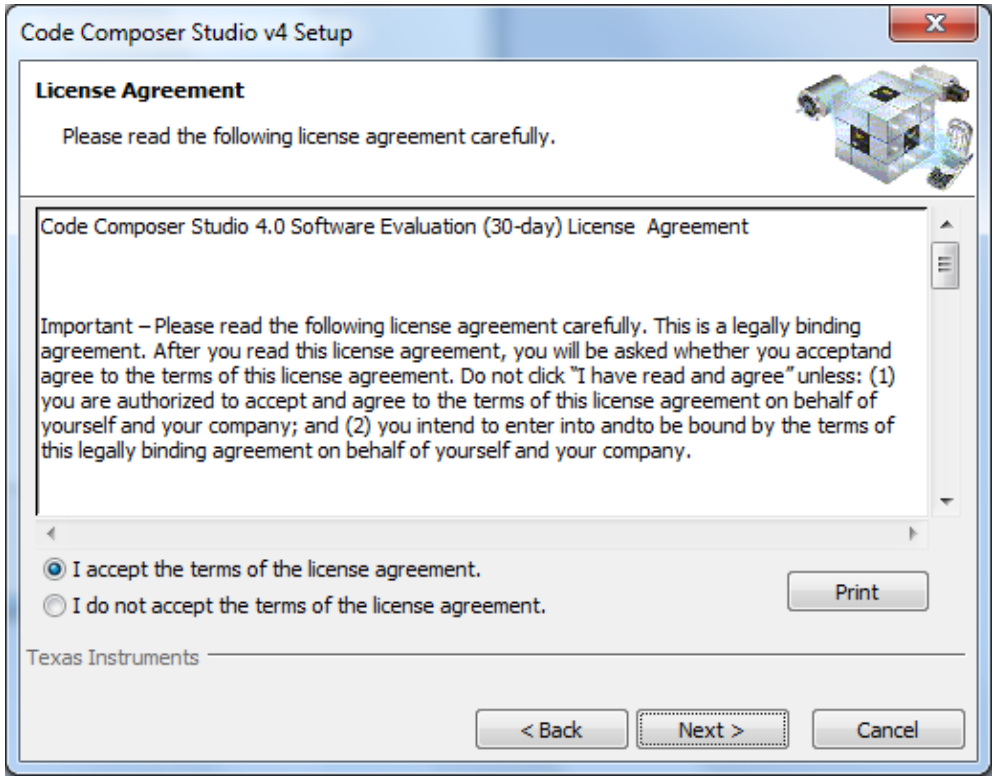
Kuvio 13

1. Aseta CCS4 DVD-levy tietokoneen DVD-asemaan ja käynnistä asennus. Aluksi ruutuun tulee asennuksen aloitusnäkymä (Kuvio 14).



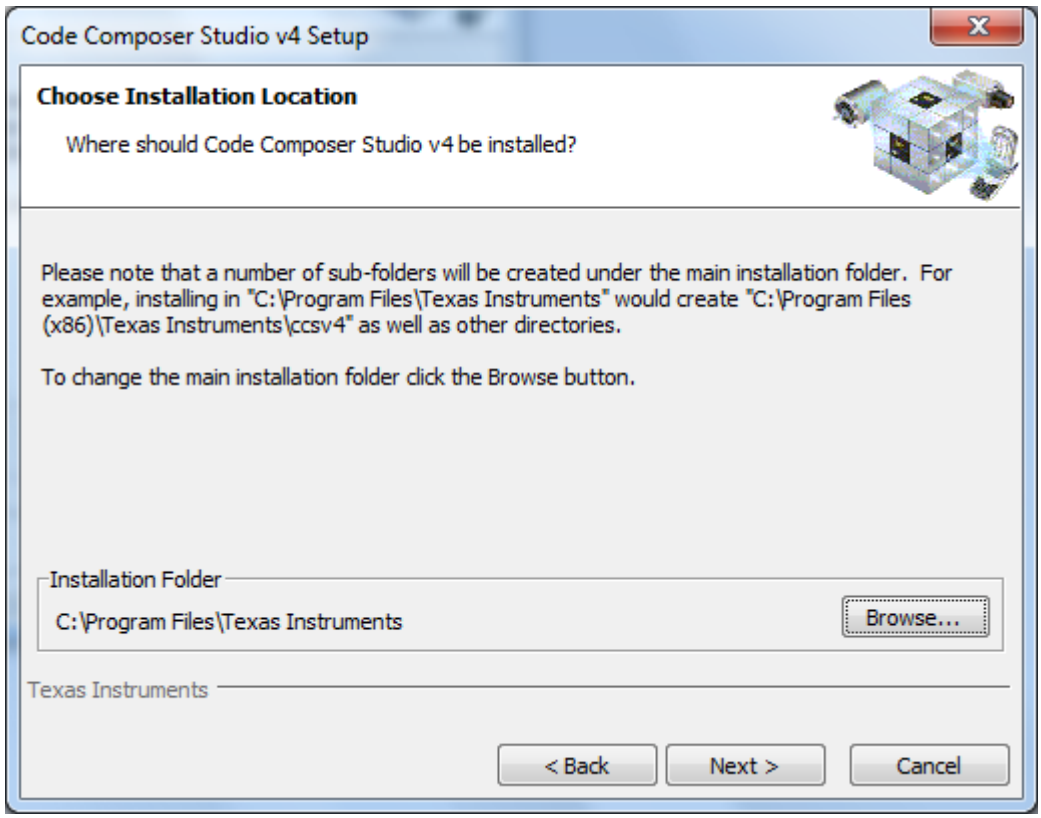
Kuvio 14

2. Hyväksy lisenssiehdot (Kuvio 15)



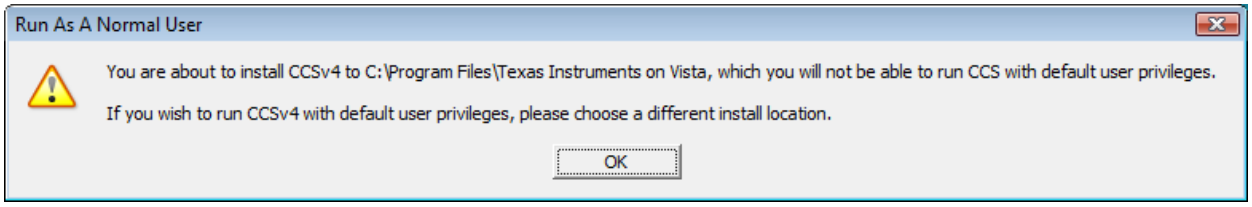
Kuvio 15

3. Valitse asennuskansio (Kuvio 16).



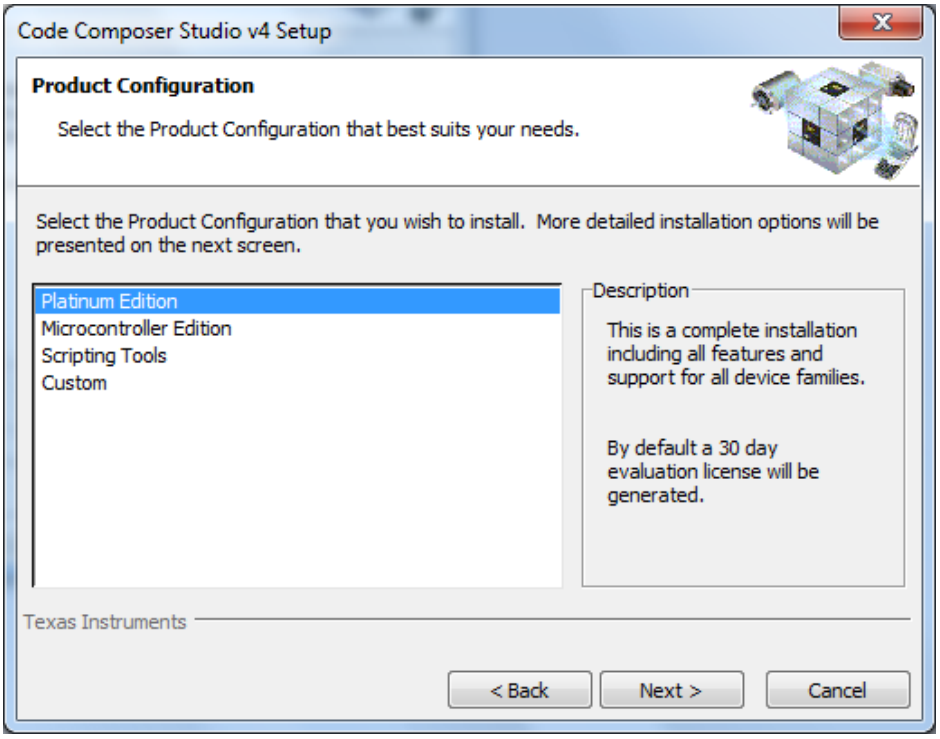
Kuvio 16

Mikäli käyttöjärjestelmänä on Windows Vista, tulee ruutuun varoitus jonka voi ohittaa kun käytössä on järjestelmänvalvojan oikeudet (Kuvio17).



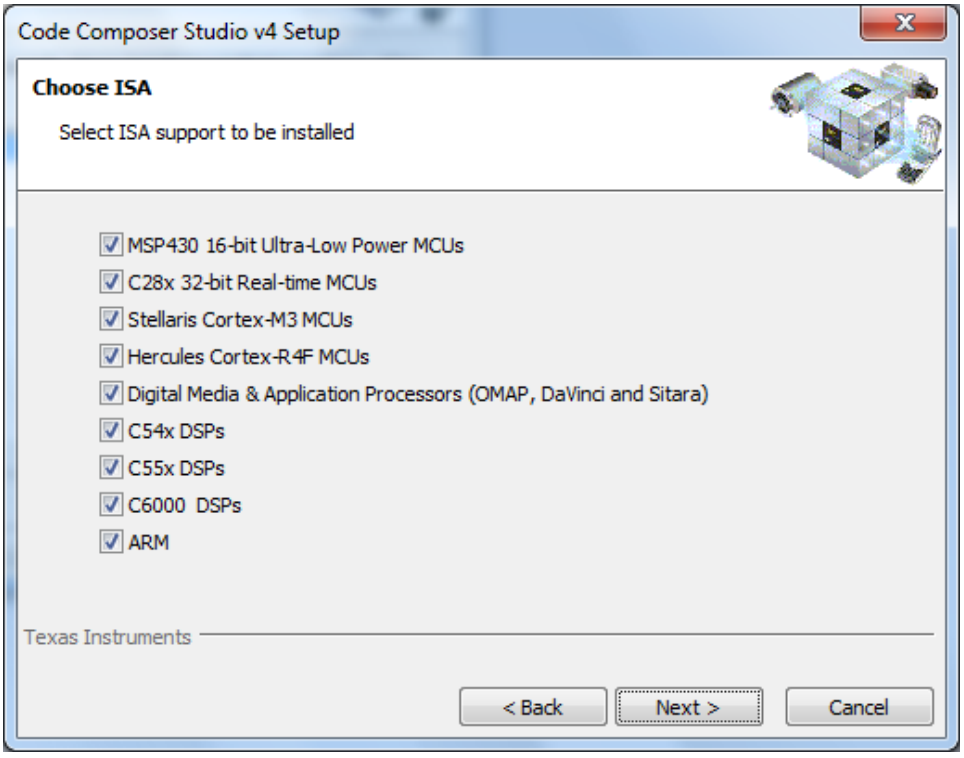
Kuvio 17

4. Valitse asennuspaketiksi platinum edition (Kuvio 18).

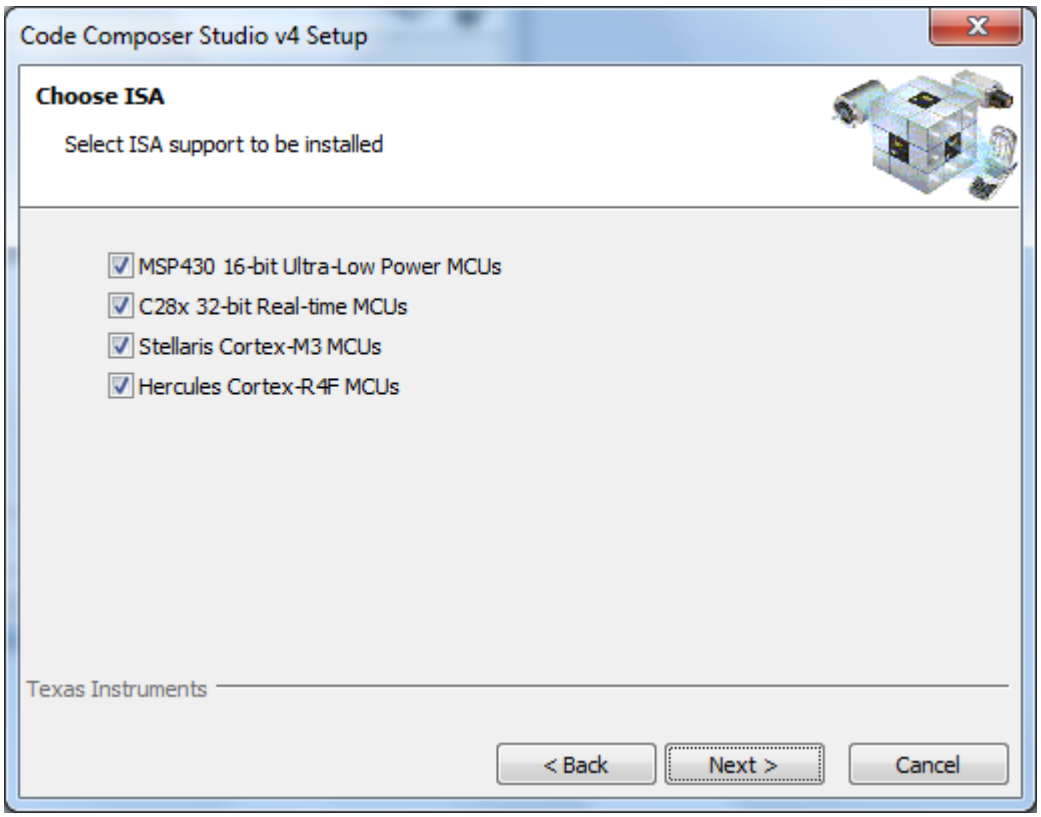


Kuvio 18

5. Valitse mihin piiriperheisiin haluat tuen, mikäli käytät ainostaan TMS320VC5505 Ezdsp-kehitysalustaa valitse ainoastaan C55x DSPs (Kuvio 19 ja 20).

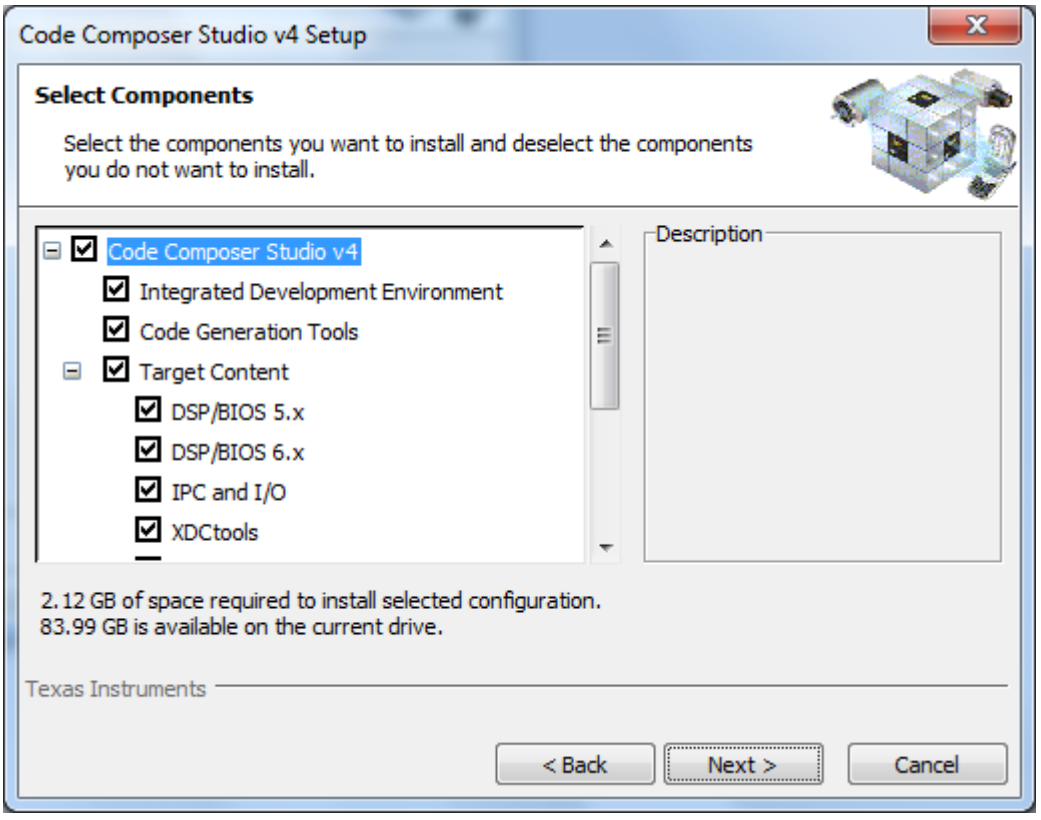


Kuvio 19



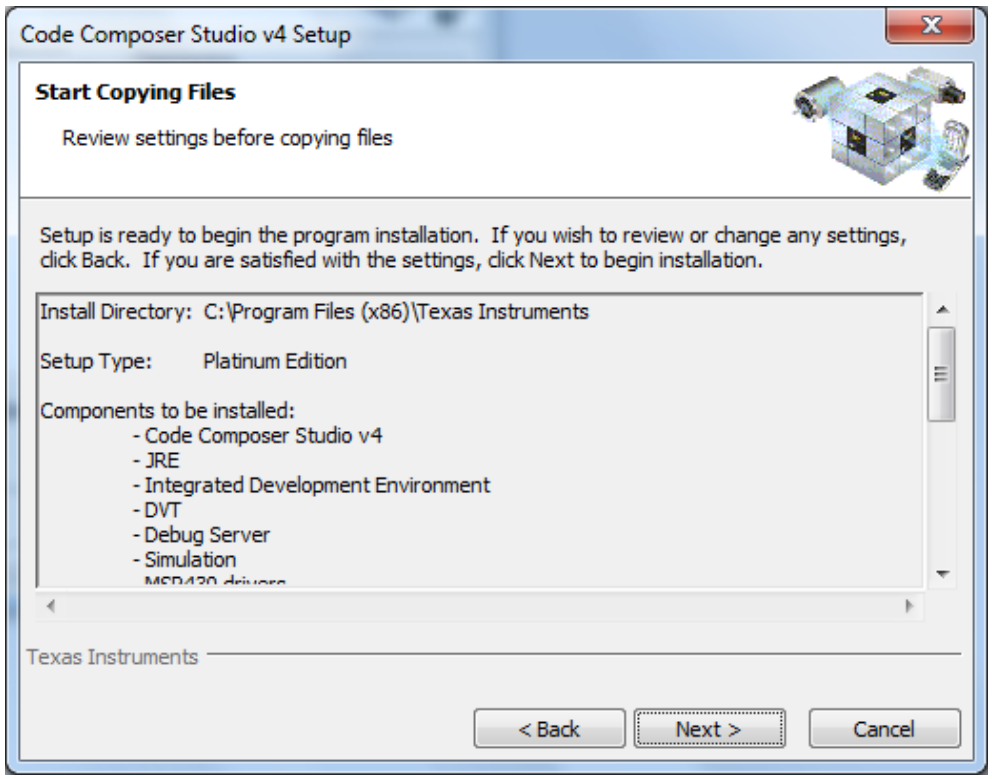
Kuvio 20

6. Valitse asennettavat komponentit, kaikki tarpeellinen on valittuna oletuksena (Kuvio 21).



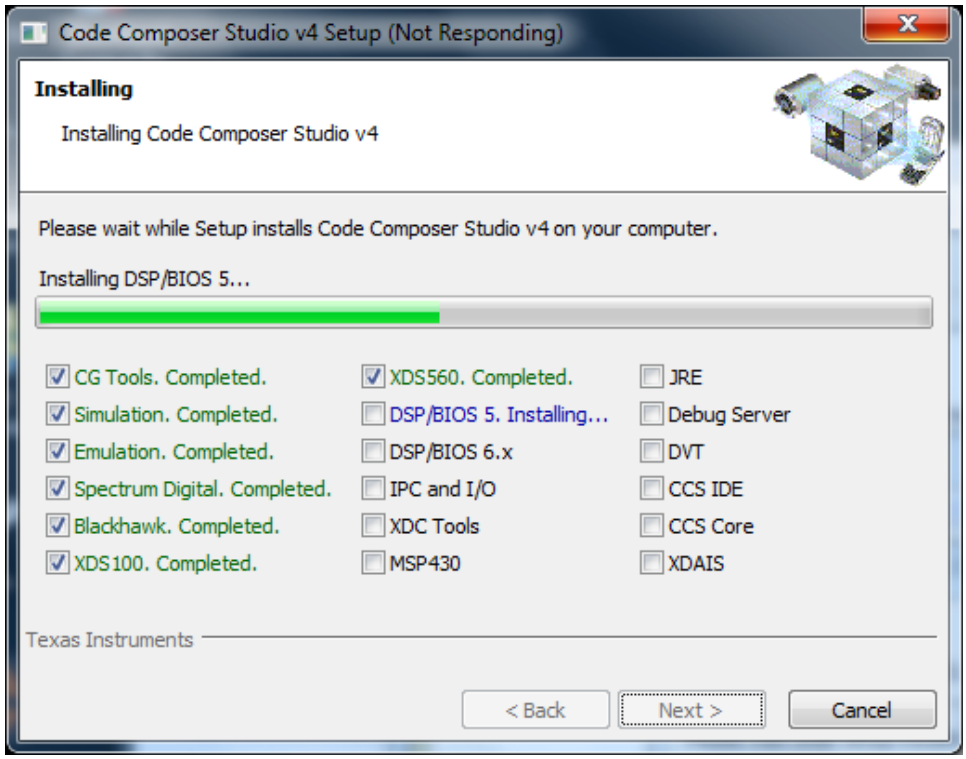
Kuvio 21

7. Tarkista asennus-yhteenvedosta, että kaikki valinnat ovat tehty oikein (Kuvio22).



Kuvio 22

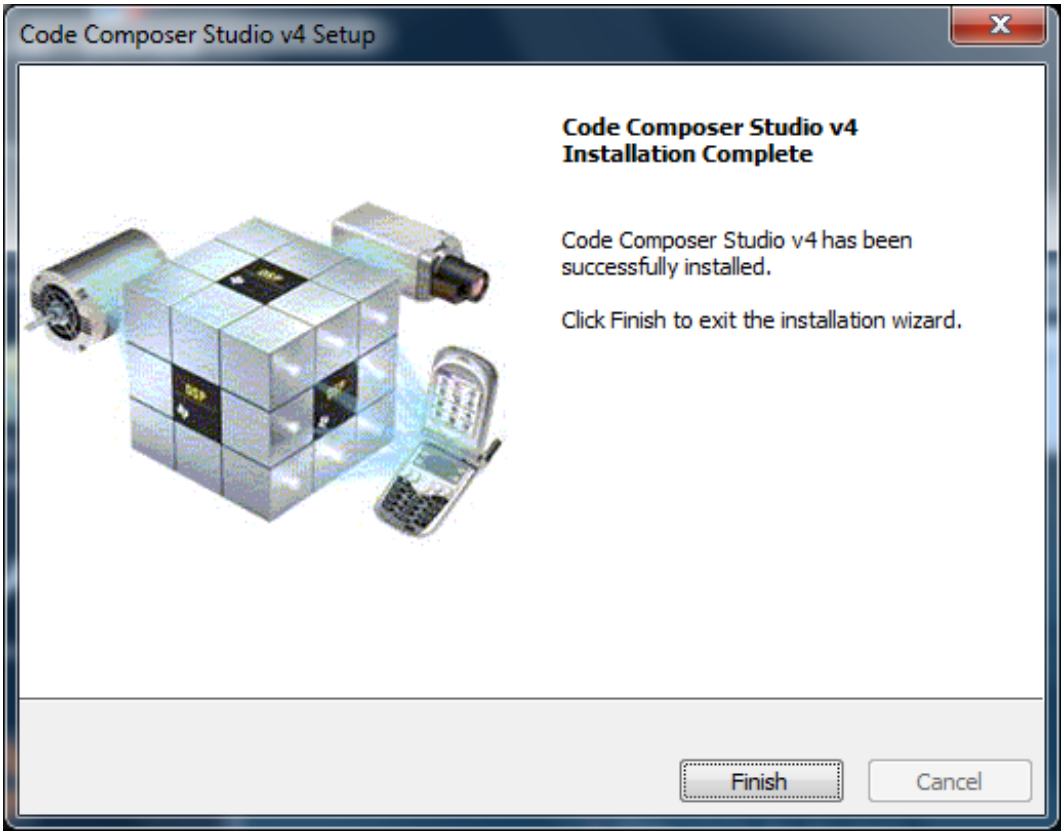
8. Asennusohjelma asentaa CCS4-ohjelman koneelle (Kuvio 23).



Kuvio 23

Mikäli asennuksen aikana käyttöjärjestelmä pyytää varmennuksia joidenkin laitteiden asennukseen, hyväksy kaikki asennukset.

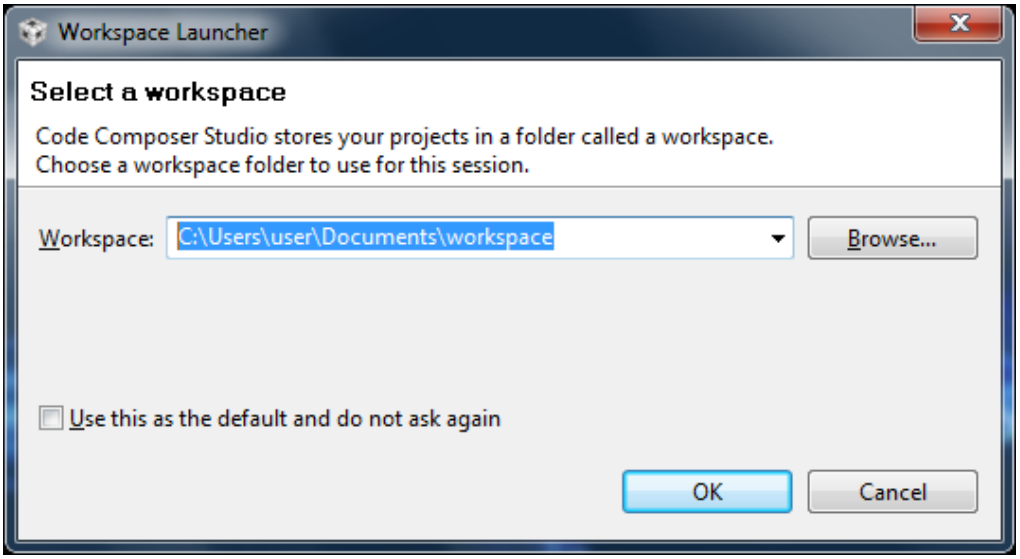
9. Lopuksi tulee ilmoitus onnistuneesta asennuksesta (Kuvio 24).



Kuvio 24

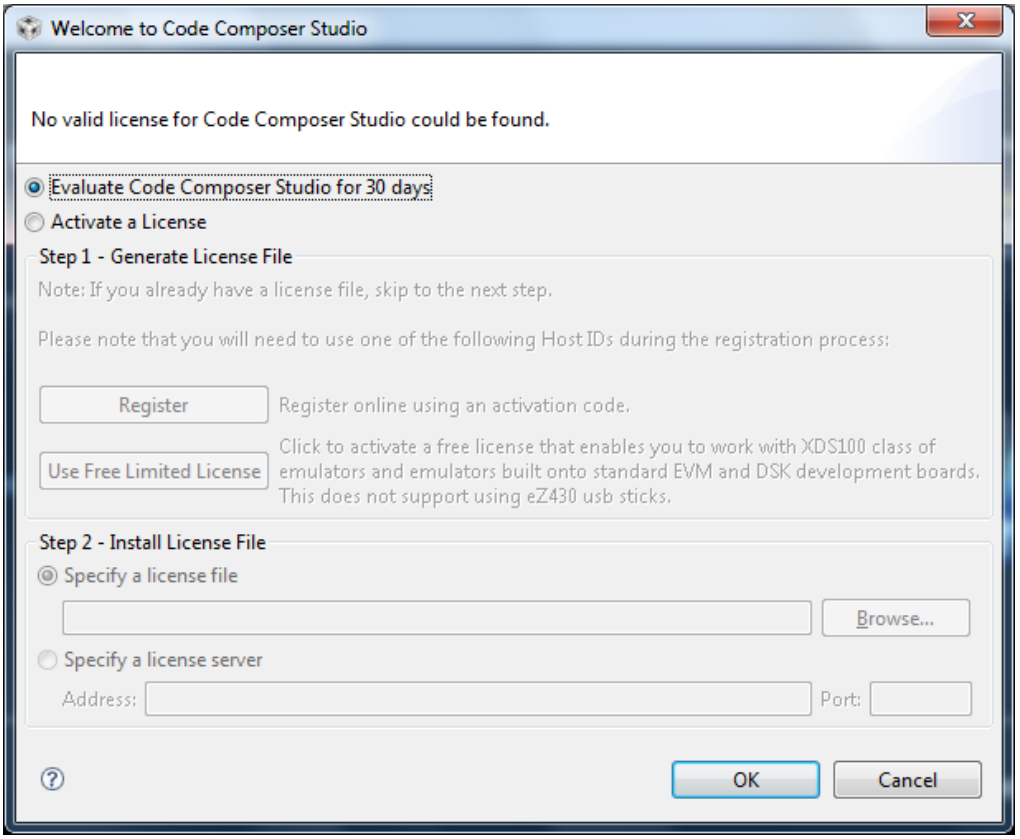
10. Laita VC5505 eZDS- kortti USB-porttiin, windows tunnistaa ja asentaa kortin automaattisesti
11. Käynnistä CCS4 työpöydän pikakuvakkeesta

12. Valitse työtilan sijainti (tänne tallennetaan kaikki projektin tiedostot, jos et halua tehdä tätä valintaa aina ohjelman käynnistyessä valitse "Use this as the default and do not ask again" (Kuvio 25).



Kuvio 25

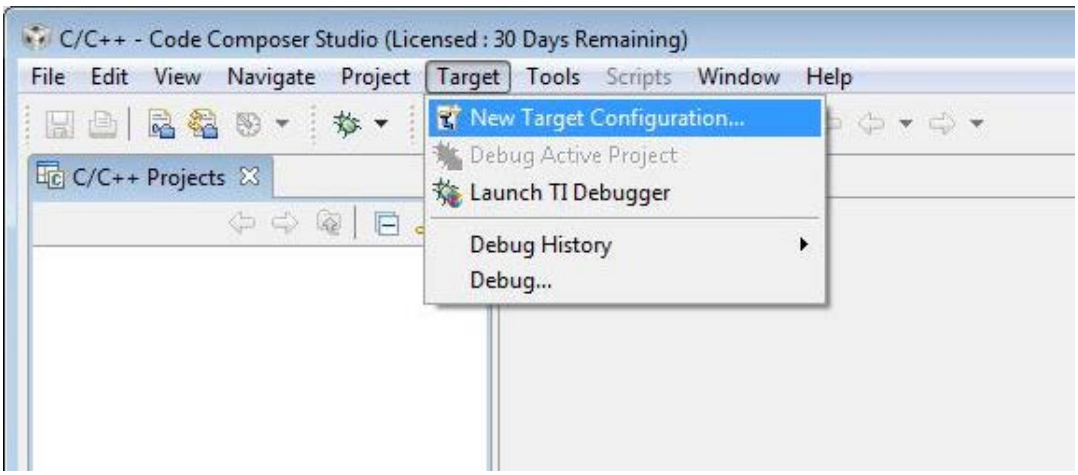
13. Valitse lisenssiksi 30 päivän kokeiluversio (Kuvio 26), myöhemmin voit aktivoida lisenssin ilmaiseksi Texas Instruments’n sivuilla.



Kuvio 26

Nyt Code Composer Studio on käyttövalmis, seuraavaksi otetaan kehitysalusta käyttöön:

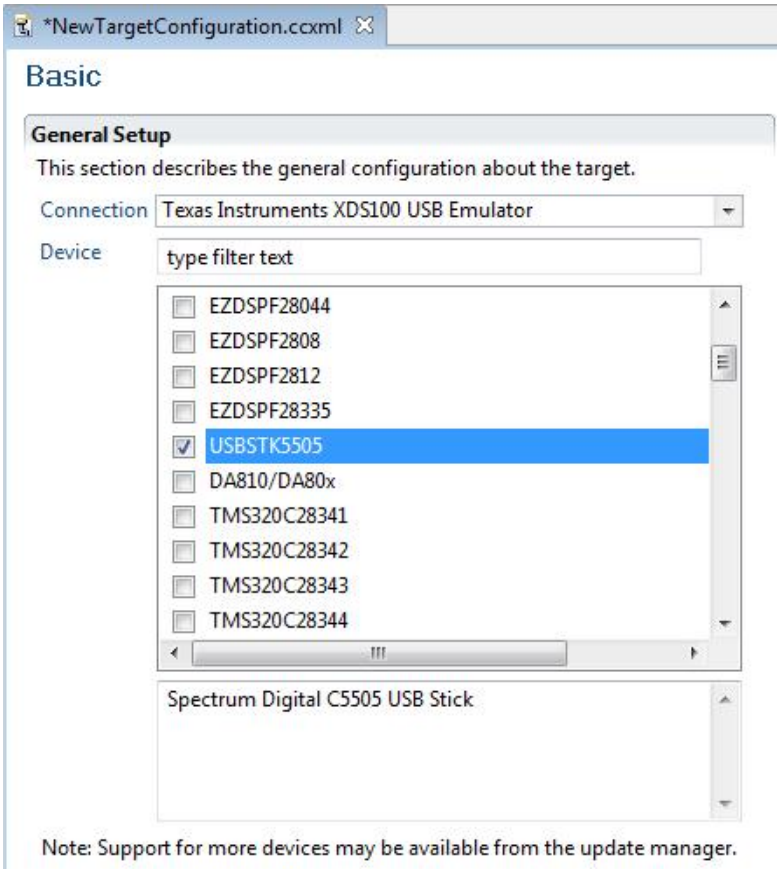
1. Käynnistä CCS4
2. Valitse Target valikosta "New Target Configuration File" (Kuvio 27).



Kuvio 27

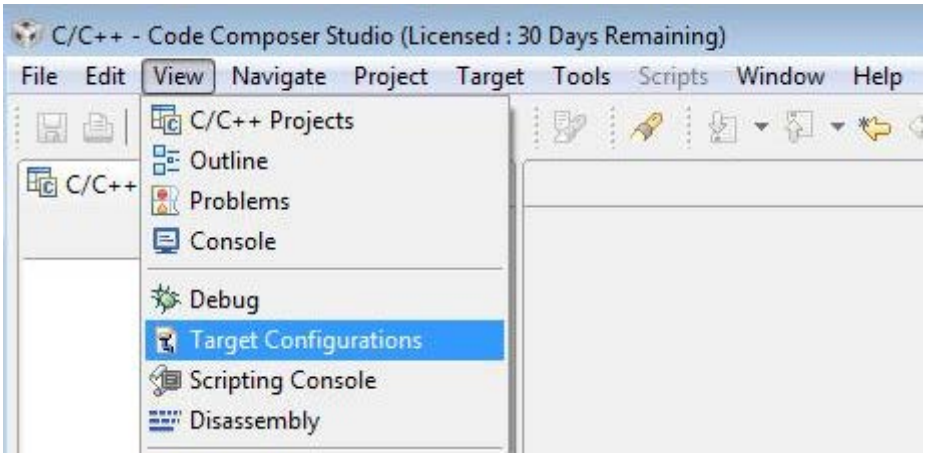
3. Nimeä configuration file, valitse Finnish

4. Valitse "connection menu" valikosta "Texas Instruments XDS100 USB Emulator ja "Board or Device" valikosta "USBSTK5505" (Kuvio28).



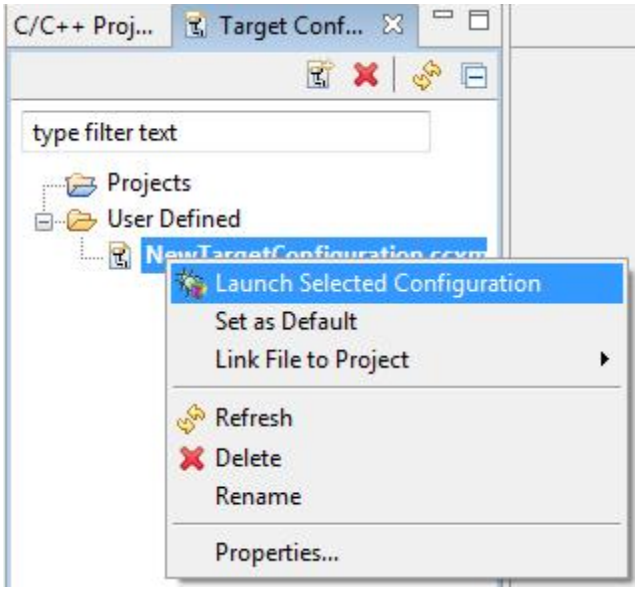
Kuvio 28

5. Valitse tallenna ja sulje valikko.
6. Valitse "View" valikosta "target configurations" (Kuvio 29).



Kuvio 29

7. Avaa "User Defined Folder" ja klikkaa hiiren oikealla napilla äsken tehtyä configuration tiedostoa, valitse "Launch Selected Configuration" (Kuvio 30).



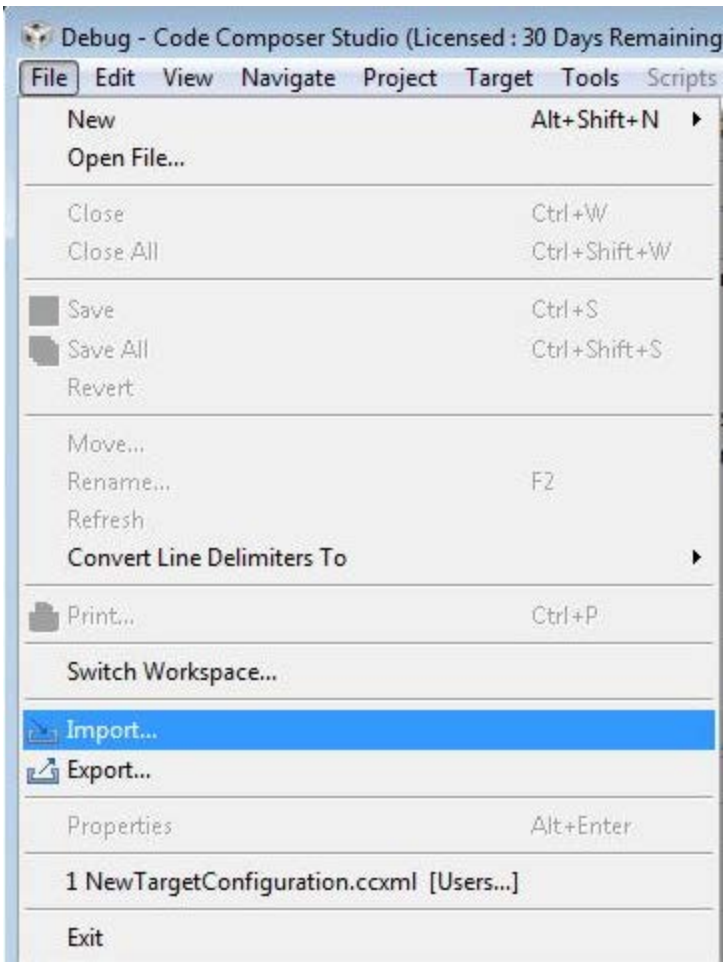
Kuvio 30

8. Valitse "Target" valikosta "Connect Target" (Kuvio 31). Nyt CCS4 ottaa yhteyden kehitysalustaan ja ajaa luodut asetukset. Kun yhteys on luotu konsoliin ilmestyy viesti "Target Connection Complete"

Kuvio 31

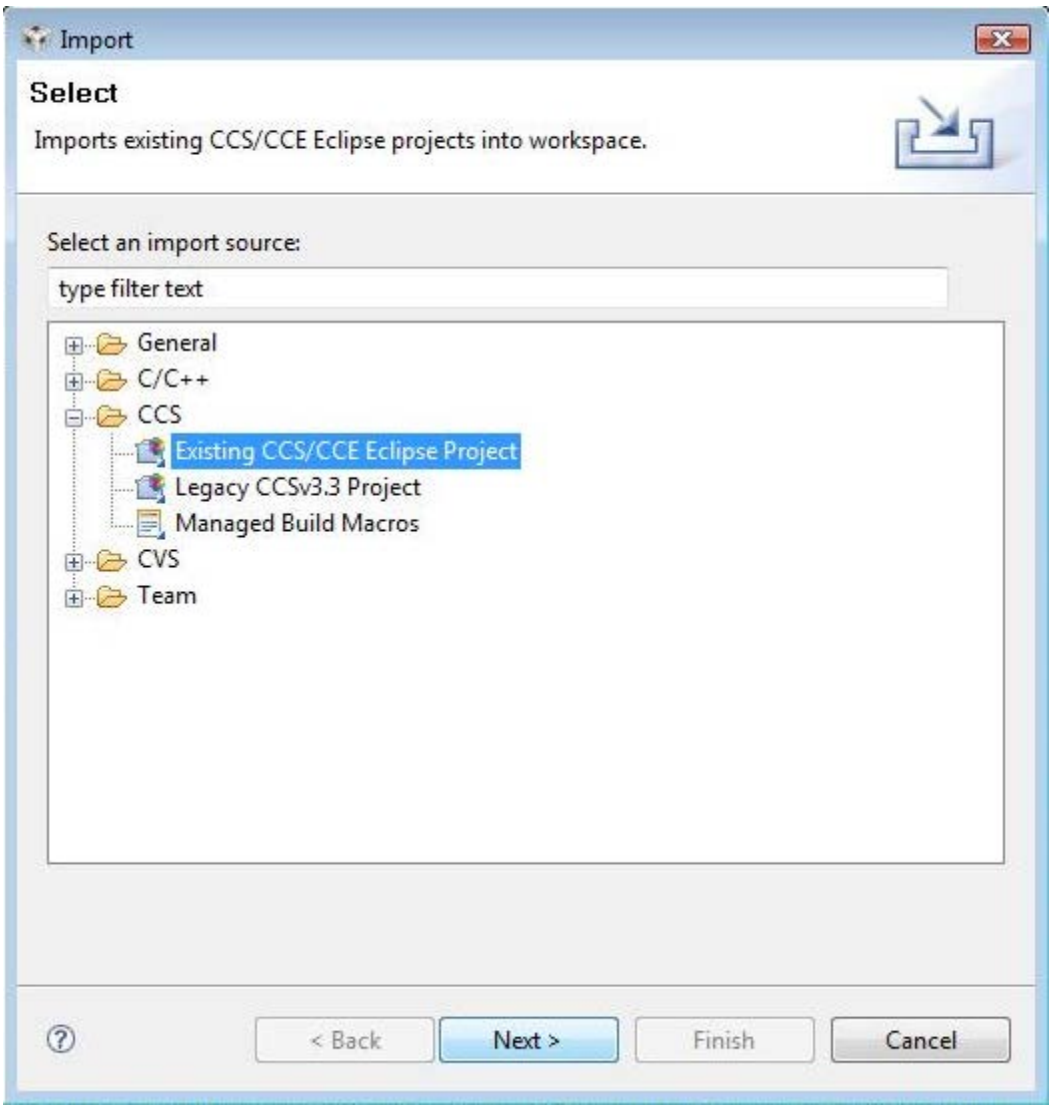
Nyt on kehitysalusta käyttökunnossa, seuraavaksi ajetaan levylle esimerkkiohjelma

1. Valitse "File->Import" (Kuvio 32).



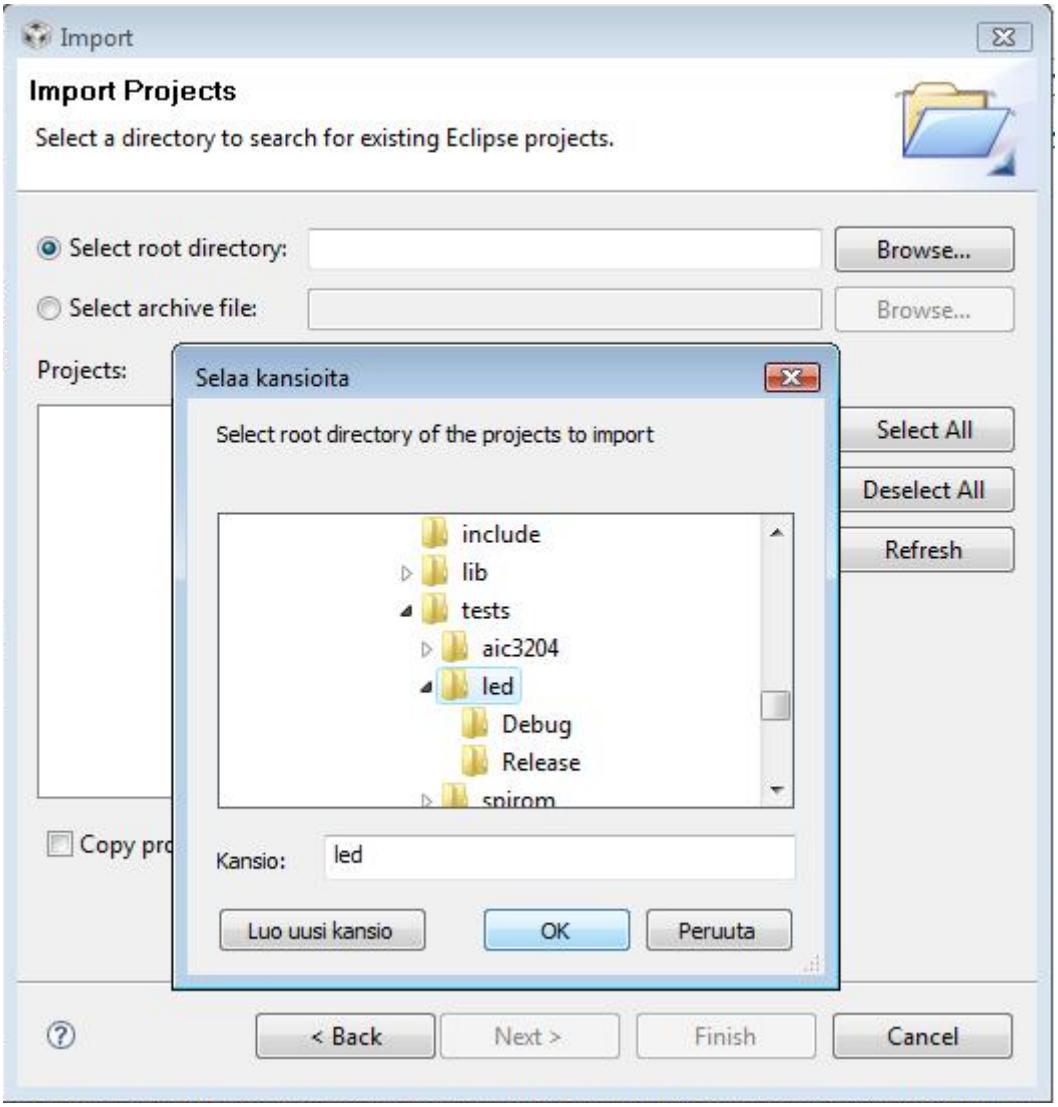
Kuvio 32

2. Avaa "CCS"-valikko ja valitse "Existing CCS/CCE Eclipse Project", valiste seuraava (Kuvio 33).



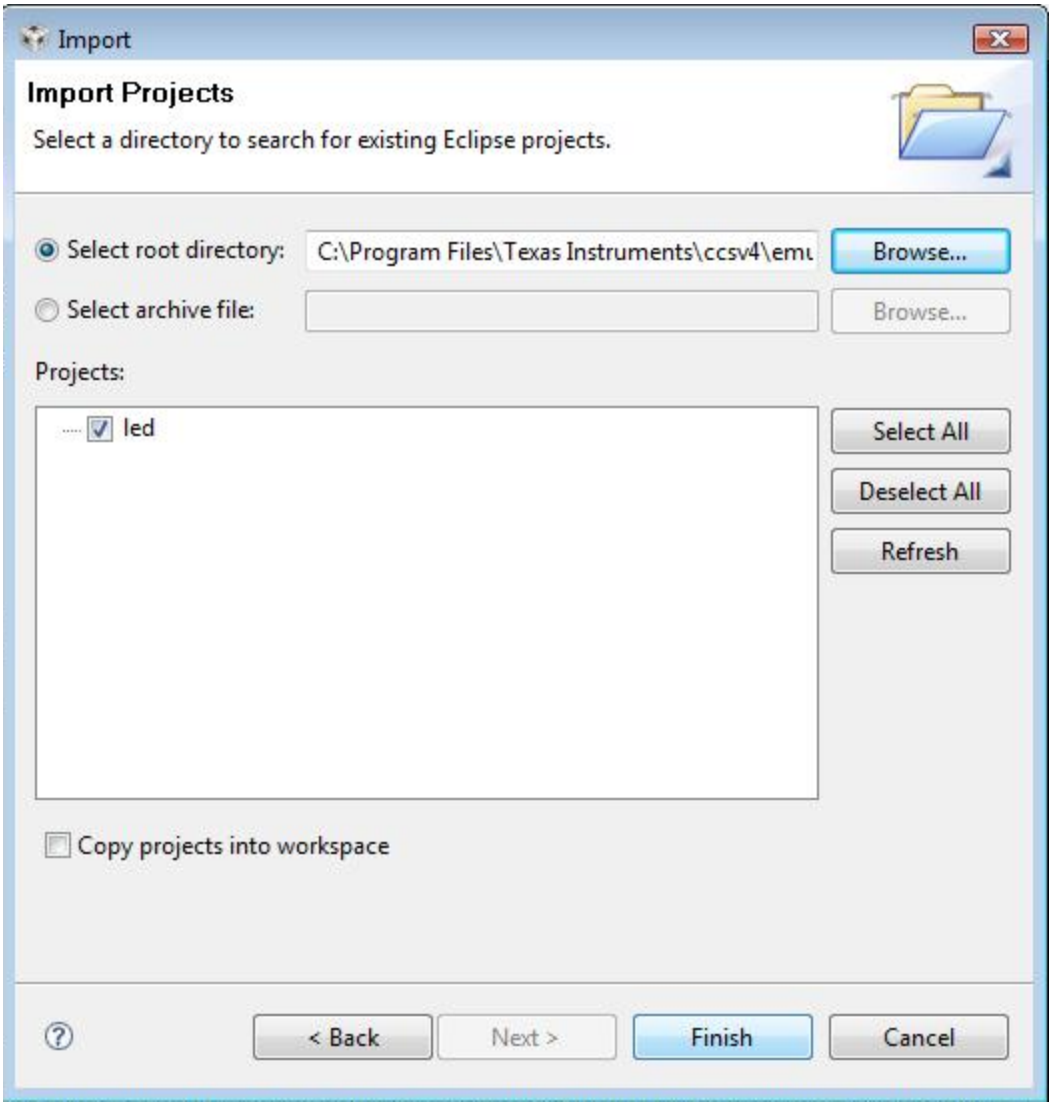
Kuvio 33

3. Valitse hakemistopolku "<asennuskansio>\ccsv4\emulation\boards\usbstk5505\tests\led" ja paina ok, oletus asennuskansio on "C:\programfiles\Texas Instruments" (Kuvio 34).



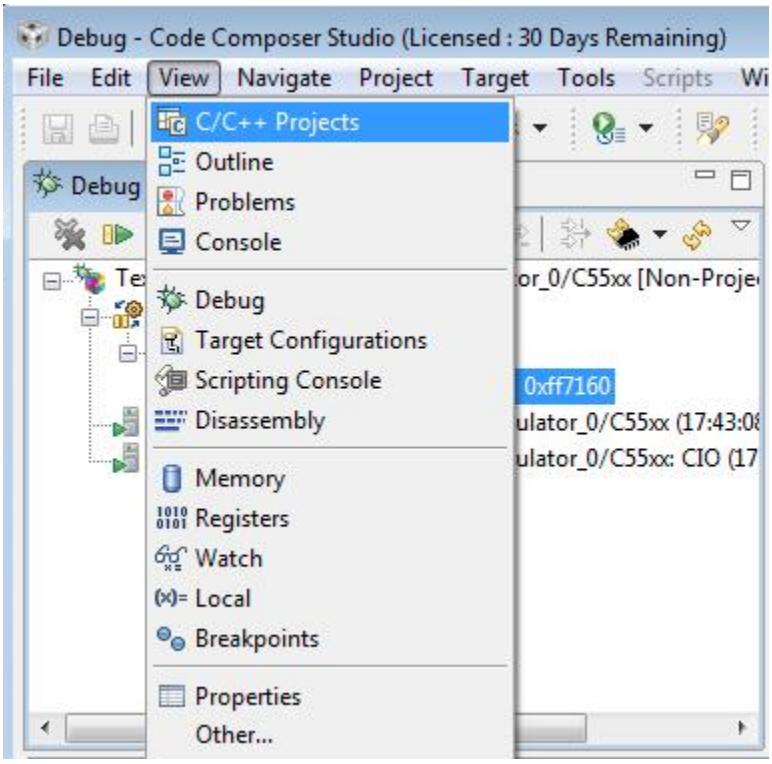
Kuvio 34

4. Varmista että led-projekti on valittuna ja paina Finnish (Kuvio35).



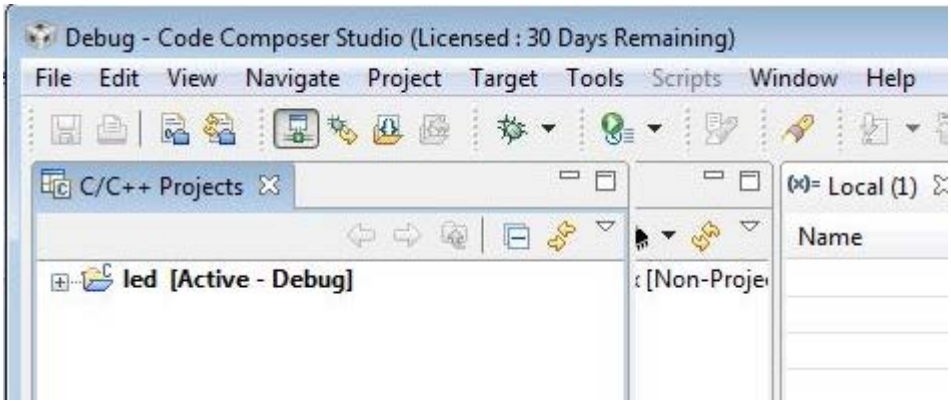
Kuvio 35

5. Valitse "View->C/C++Projects" (Kuvio 36).



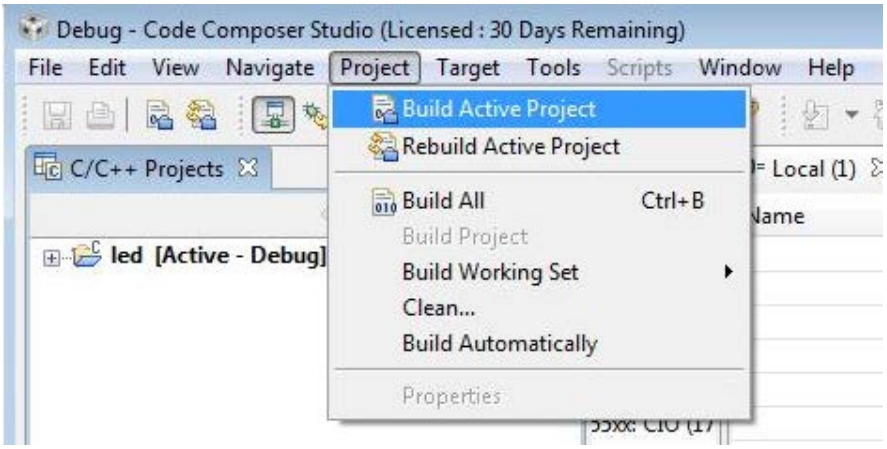
Kuvio 36

Vasemmalle aukeaa kansio jossa näkyy Led project (Kuvio 37).



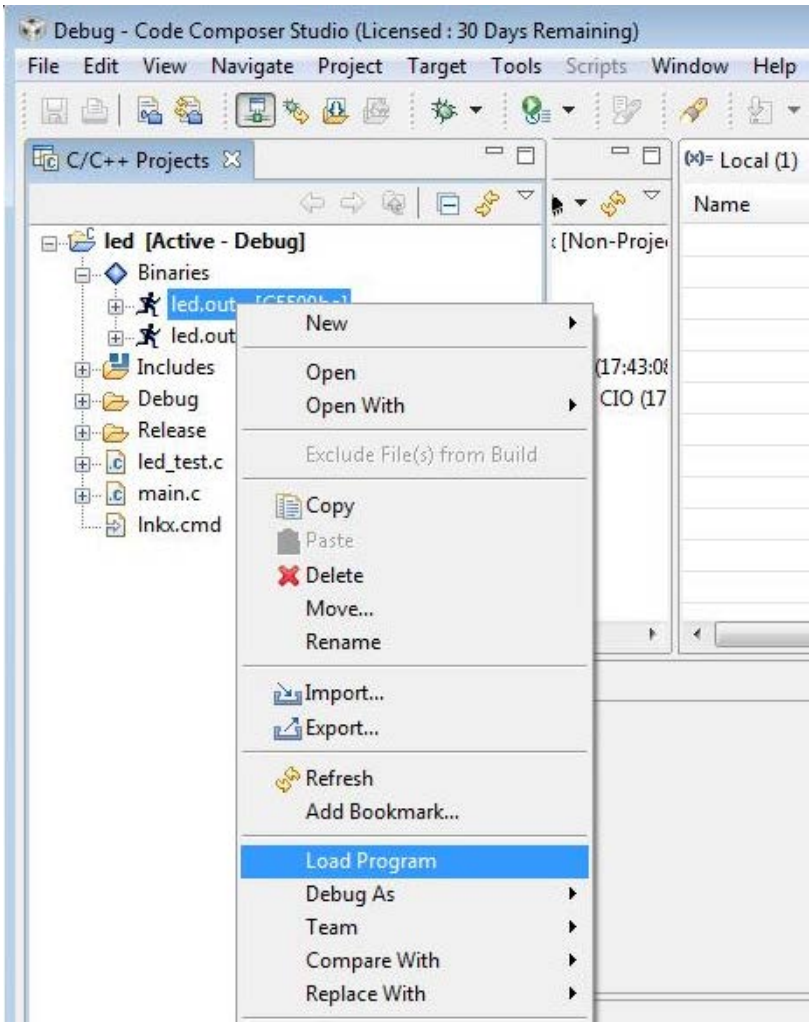
Kuvio 37

6. Valitse "Project->Build Active Project" (Kuvio 38).



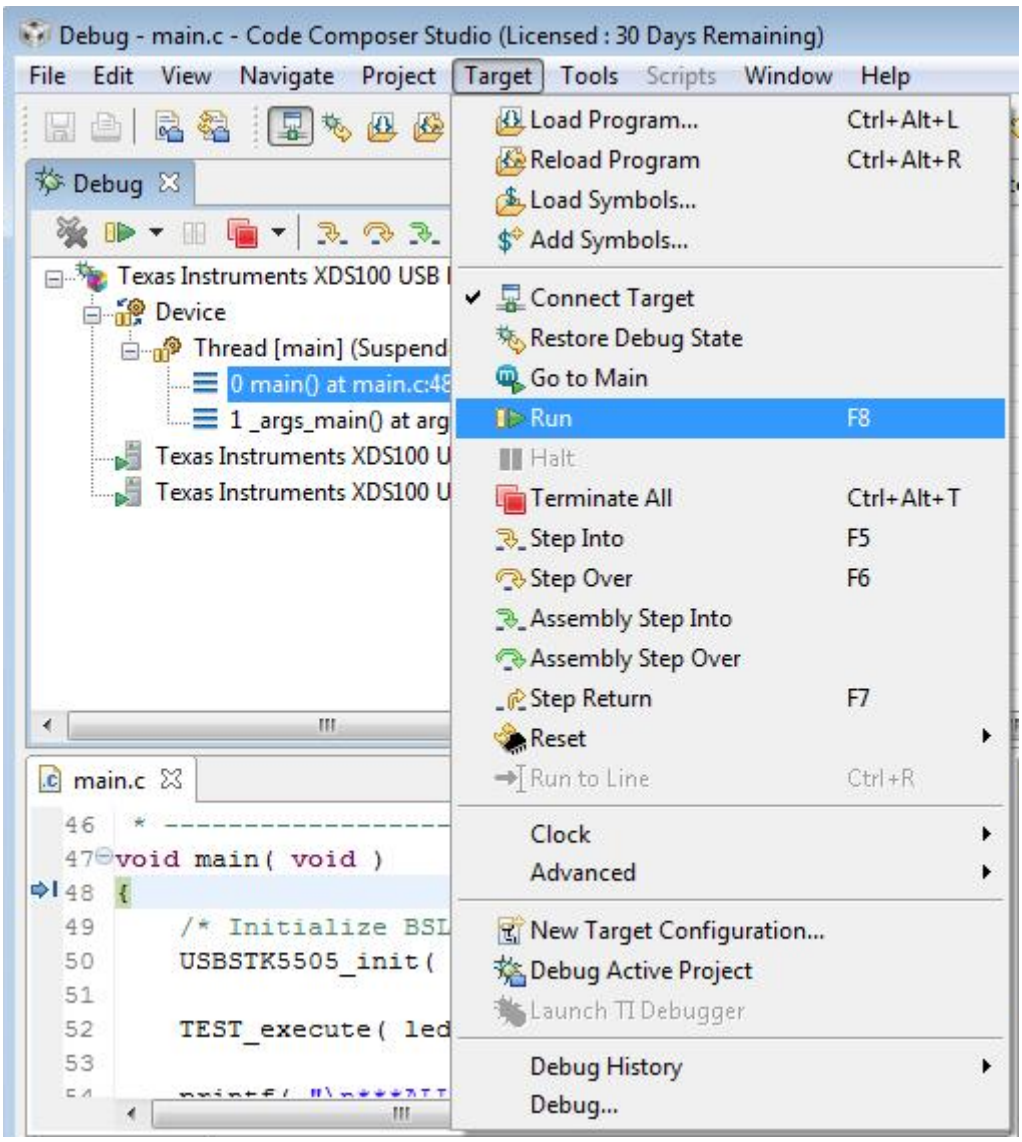
Kuvio 38

7. Avaa C\C++Project välilehden "binaries" kansio.
8. Paina oikealla hiiren napilla "led.out" tiedoston päällä ja valitse "load program" (Kuvio 39).



Kuvio 39

9. Valitse "Target->Run". VC5505 ezDSP USB kortin ledin pitäisi alkaa vilkkumaan (Kuvio 40).



Kuvio 40

Nyt olet ajanut ensimmäisen toimivan sovelluksen kehitysalustalla. Target valikosta voit myös valita erilaisia debug toimintoja, sekä keskeyttää ohjelman ajamisen.

Liite2: Esimerkkiprojektien lähdekoodit

```
/*****************************************************************
* Mika Heikkinen
* Esimerkkiprojekti 1: Näytteenotto taajuuden muokkaus main.c
* 16.8.2010
*****************************************************************/

#include "stdio.h"
#include "usbstk5505.h"
#include "usbstk5505_gpio.h"
#include "usbstk5505_i2c.h"
#include "aic_settings.h"
#define XmitL 0x10
#define XmitR 0x20
#define RcvR 0x08
#define RcvL 0x04



void main(void)
{

    Int16 sample, data1, data2 ,data3, data4, kytkin,
    kytkin2,jakaja1=132, jakaja2=132,
    kytkin3, kytkin4,testi2;

   /* alustuksia */
   USBSTK5505_init( );
   // asetetaan A20 pinni => GPIO26(codecin resetti)
   SYS_EXBUSSEL |= 0x0020;
   USBSTK5505_GPIO_init();
   USBSTK5505_GPIO_setDirection(GPIO26, GPIO_OUT);
   USBSTK5505_GPIO_setDirection(GPIO22, GPIO_IN);
   USBSTK5505_GPIO_setDirection(GPIO23, GPIO_IN);
   USBSTK5505_GPIO_setDirection(GPIO24, GPIO_IN);
   USBSTK5505_GPIO_setDirection(GPIO25, GPIO_IN);
   USBSTK5505_GPIO_setOutput( GPIO26, 1 );  // codec pois resetistä
   USBSTK5505_I2C_init( );                  // I2C väylän aslustus

            /* sarjaväylän asetukset */

   // sarjaväylä 0 käyttämään I2S0     protokollaa
   SYS_EXBUSSEL |= 0x0100;
   //codecille sovelluksen oletus asetukset
   AIC3204_settings_48kHz(jakaja1, jakaja2);
   /* I2S settings */
   I2S0_SRGR = 0x0;
   // I2S sananpituus 16-bit, orja mode, väylä käyttöön
   I2S0_CR = 0x8010;
   I2S0_ICMR = 0x3f;    // Sallitaan keskeytykset

    while(1)
    {
           /*io porttien luku*/
           kytkin = USBSTK5505_GPIO_getInput(GPIO22);
           kytkin2 = USBSTK5505_GPIO_getInput(GPIO23);
           kytkin3 = USBSTK5505_GPIO_getInput(GPIO24);
           kytkin4 = USBSTK5505_GPIO_getInput(GPIO25);
```

```
/* kytkinten painallus kasvattaa/pienentää lähetetyn
parametrien arvoa jotka  määrää jakajan*/
if(kytkin == 1)
{
          jakaja1++;
          if(jakaja1 == 251)
          jakaja1 = 132;

          USBSTK5505_wait( 1000 );
          AIC3204_settings_48kHz(jakaja1, jakaja2);

}
if(kytkin2 == 1)
{
          jakaja2++;
          if(jakaja2 == 251)
          jakaja2 = 132;

          USBSTK5505_wait( 1000 );
          AIC3204_settings_48kHz(jakaja1,jakaja2);

}
if(kytkin3 == 1)
{
          jakaja1--;
          if(jakaja1 == 131)
          jakaja1 = 250;

          USBSTK5505_wait( 1000 );
          AIC3204_settings_48kHz(jakaja1, jakaja2);

}
if(kytkin4 == 1)
{
          jakaja2--;
          if(jakaja2 == 131)
          jakaja2 = 250;

          USBSTK5505_wait( 1000 );
          AIC3204_settings_48kHz(jakaja1, jakaja2);

}

/* luetaan codecin ADCltä tulevaa dataa */


// odotetaan että vastaanoton keskeytyslippu laskee
while((RcvR & I2S0_IR) == 0);
// vasemman kanavan 16-bit data muuttujaan
data3 = I2S0_W0_MSW_R;
// oikean kanavan 16-bit data muuttujaan
data4 = I2S0_W1_MSW_R;


/* lähetetään data takaisin codecille */


// odotetaan että lähetys keskeytyslippu laskee
while((XmitR & I2S0_IR) == 0);
```

```
            // vasemman kanavan 16-bit data muuttujasta
            //lähetysrekisteriin
            I2S0_W0_MSW_W = data3;
            // oikean kanavan 16-bit data muuttujasta
            //lähetysrekisteriin
            I2S0_W1_MSW_W = data4;

   }
}


/*********************************************************************
* Mika Heikkinen
* Esimerkkiprojekti 1: Funktioiden esittely aic_settings.c
* 16.8.2010
*********************************************************************/
#ifndef AIC_SETTINGS_H_

Int16 AIC3204_rset( Uint16 regnum, Uint16 regval );
Int16 AIC3204_rget(  Uint16 regnum, Uint16* regval );
void AIC3204_settings_48kHz(Uint16 jakaja1,Uint16 jakaja2);

#define AIC_SETTINGS_H_

#endif /*AIC_SETTINGS_H_*/



/*********************************************************************
* Mika Heikkinen
* Esimerkkiprojekti 1: codecin asetukset aic_settings.c
* 16.8.2010
*********************************************************************/
#include "stdio.h"
#include "usbstk5505.h"
#include "usbstk5505_gpio.h"
#include "usbstk5505_i2c.h"
#include "aic_settings.h"

#define AIC3204_I2C_ADDR 0x18 //codecin osoite


Int16 AIC3204_rset( Uint16 regnum, Uint16 regval )
{
    Uint8 cmd[2];
    cmd[0] = regnum & 0x007F;
    cmd[1] = regval;

    return USBSTK5505_I2C_write( AIC3204_I2C_ADDR, cmd, 2 );
}
void AIC3204_settings_48kHz(Uint16 jakaja1, Uint16 jakaja2)
{
     int apu = jakaja1-128;
           int apu2 = jakaja2-128;
           long int apu4;
           long int jakaja3 = 0;
           long int apu3 = 100000000/apu;
```

```
        apu3 = apu3/apu2;
        apu3 = apu3/128;

        jakaja3 = (100000000/(apu3*32*apu))+128;

        if(jakaja3 > 254)
        {
                  jakaja3 = 255;
        }
        if(jakaja3 < 129)
        {
                  jakaja3 = 129;
        }

        printf( "bclk = %ld \n", apu4 );
        printf( "jakaja1 = %d, jakaja2 = %d  \n", apu, apu2 );
        printf( "näytteenotto taajuus on %ld  \n", apu3 );
        printf( "N jakaja %ld  \n", jakaja3 );


        /* Codec piirin asetukset */

        AIC3204_rset( 0, 0 ); // Valitaan rekisterisivu 0
        AIC3204_rset( 1, 1 ); // Resetoidaan codec
        AIC3204_rset( 0, 1 ); // Valitaan sivu 1
        AIC3204_rset( 1, 8 ); // poistetaan AVDD ja DVDD yhteys
        AIC3204_rset( 2, 1 ); // Analogilohko käyttöön AVDD päälle
        AIC3204_rset( 0, 0 ); // valitaan sivu 0

       /* PLL ja kellojen asetus/käynnistys  */

        AIC3204_rset( 27, 0x1d );   // BCLK and WCLK ulostuloksi
        AIC3204_rset( 30, jakaja3 );//  BCLK=DAC_CLK/N
        AIC3204_rset( 28, 0x00 );   // Data offset = 0


AIC3204_rset( 4, 3 ); //PLL lähteeksi MCLK,PLL codec kelloksi
AIC3204_rset( 6, 8 );      // PLL kertoja J=8
AIC3204_rset( 7, 15 );     // PLL kertoja: HI_BYTE(D) (4060)
AIC3204_rset( 8, 0xdc );   // PLL kertoja: LO_BYTE(D)

AIC3204_rset( 5, 0x91 );   //PLL käyntii P ja R kertoja = 0
AIC3204_rset( 13, 0 );     // Hi_Byte(DOSR) DOSR = 128
AIC3204_rset( 14, 0x80 );  // Lo_Byte(DOSR) DOSR = 128
AIC3204_rset( 20, 0x80 );  // AOSR = 128


AIC3204_rset( 11, jakaja1 );//NDAC päälle,käyttäjä säätääjakaja1
AIC3204_rset( 12, jakaja2 );//MDAC päälle, käyttäjä säätää jakaja2
AIC3204_rset( 18, 0x88 );   // NADC päälle = 8
AIC3204_rset( 19, 0x82 );   // MADC päälle = 2

/* DAC reititys ja käynnistys */
AIC3204_rset( 0, 1 );      // Valitaan sivu 1
AIC3204_rset( 0x0c, 8 );   // LDAC=> vasen kuuloke
AIC3204_rset( 0x0d, 8 );   // RDAC=>oikea kuuloke
AIC3204_rset( 0, 0 );      // valitaan sivu 0
AIC3204_rset( 64, 2 );     // vas volume=oik volume
AIC3204_rset( 65, 0 );     // vasen DAC vahvistus 0dB
AIC3204_rset( 63, 0xd4 );  // käynnistetään oik ja vas DAC
AIC3204_rset( 0, 1 );      // Valitaan sivu 1
AIC3204_rset( 0x10, 0 );   // äänet päälle vas kuuloke, 0dB
AIC3204_rset( 0x11, 0 );   // äänet päälle oik kuuloke, 0dB
```

```
    AIC3204_rset( 9, 0x30 );   // virrat päälle oik ja vas kuuloke
    AIC3204_rset( 0, 0 );      // valitaan sivu 0
    USBSTK5505_wait( 100 );    // viive
    /* ADC reititys ja käynnistys */
    AIC3204_rset( 0, 1 );      // Valitaan sivu 1
    AIC3204_rset( 0x34, 0x30 );// STEREO 1 Jack vas sisääntulo
                               // impedanssi  40kohm micpga +nasta

    AIC3204_rset( 0x37, 0x30 );// STEREO 1 Jack oik sisääntulo
                               // impedanssi  40kohm micpga +nasta

    AIC3204_rset( 0x36, 3 ); //vas kanavan micpga -nasta 40 kohm läpi
                             //maihin
    AIC3204_rset( 0x39, 0xc0 );//oik kanavat micpga -nasta 40 kohm
                               //läpi maihin

    AIC3204_rset( 0x3b, 0 );   // MIC_PGA_L äänet päälle
    AIC3204_rset( 0x3c, 0 );   // MIC_PGA_R äänet päälle
    AIC3204_rset( 0, 0 );      // valitaan sivu 0
    AIC3204_rset( 0x51, 0xc0 );// virrat päälle vas ja oik ADC
    AIC3204_rset( 0x52, 0 );   // äänet päälle vas ja oik ADC

    AIC3204_rset( 0, 0 );
    USBSTK5505_wait( 100 );    // viive
}



/************************************************************************
* Mika Heikkinen
* Esimerkkiprojekti 2: Aktiivinen kuulosuojain
* 16.8.2010
************************************************************************/

#include "stdio.h"
#include "usbstk5505.h"
#include "usbstk5505_gpio.h"
#include "usbstk5505_i2c.h"
#define AIC3204_I2C_ADDR 0x18
#define XmitL 0x10
#define XmitR 0x20
#define RcvR 0x08
#define RcvL 0x04



Int16 AIC3204_rset( Uint16 regnum, Uint16 regval );
Int16 AIC3204_rget(  Uint16 regnum, Uint16* regval );

void main(void)
{

    Int16 data3, data4, ohi = 0, rget_laskuri = 0;

    Uint32 ohi_laskuri=0;

    Uint16 agc_gain=0, agc_ntf;
```

```
  /* alustuksia */
    USBSTK5505_init( );
    //asetetaan A20 pinni=>GPIO26(codecin resetti)

            SYS_EXBUSSEL |= 0x0020;
            USBSTK5505_GPIO_init();
            USBSTK5505_GPIO_setOutput( GPIO26, 1 ); // codec pois
resetistä
            USBSTK5505_I2C_init( );             // I2C väylän alustus


            /* sarjaväylän asetukset*/
            SYS_EXBUSSEL |= 0x0100;  //sarjaväylä 0 käyttämään I2S
protokollaa

            //Softa resetti

            AIC3204_rset( 0, 0 );      // Valitaan rekisterisivu 0
            AIC3204_rset( 1, 1 );      // Resetoidaan codec

     //Kello asetukset ja interface

            AIC3204_rset( 27, 0x1d );  // BCLK ja WCLK I2S väylän
kelloiksi
            AIC3204_rset( 28, 0x00 );  // Data offset = 0
            AIC3204_rset( 4, 0 );      // MCLK pinnistä codecin kello
            AIC3204_rset( 30, 0x87 );  // BCLK=DAC_CLK/N =(12288000/8)=
//1.536MHz = 32*fs

            AIC3204_rset( 11, 0x81 );  // Ndac = 1
            AIC3204_rset( 12, 0x82 );  // MDAC = 2

            AIC3204_rset( 18, 0x81 );  // NADC = 1
            AIC3204_rset( 19, 0x82 );  // MADC = 2

            AIC3204_rset( 13, 0 );     // msb(DOSR) DOSR = 128
            AIC3204_rset( 14, 0x80 );  // lsb(DOSR) DOSR = 128
            AIC3204_rset( 20, 0x80 );  // AOSR = 128

            AIC3204_rset( 0, 0 );      // Valitaan rekisterisivu 0

            //Prosessing blocks

            AIC3204_rset( 60, 0x8 );  // DAC prosessing block 8
            AIC3204_rset( 61, 0x7 );  // ADC prosessing block 7

            //jännitelähteet ja powertune

            AIC3204_rset( 0, 1 );  // Valitaan rekisterisivu 1
            AIC3204_rset( 1, 8 );  // poistetaan AVDD ja DVDD yhteys
            AIC3204_rset( 2, 1 );  // Analogi lohko päälle LDO käyttöön

            AIC3204_rset( 61, 0xff );  // ADC Powertune mode 1

            AIC3204_rset( 3, 0x8 );    // left DAC Powertune mode 1
            AIC3204_rset( 4, 0x8 );    // right DAC Powertune mode 1

            //AIC3204_rset( 59, 40 );    // pga 20db
            //AIC3204_rset( 60, 40 );    // pga 20db

            AIC3204_rset( 0, 0 );      // Valitaan rekisterisivu 0
```

```
        /*  AGC parametrien asetus */

        //Vasen Target Level päälle, -12dBFS. Hystereesi 1.5db
        AIC3204_rset(86, 0xB3  );
        //Oikea Target Level päälle, -12dBFS. Hystereesi 1.5db
        AIC3204_rset(94, 0xB3  );

        //Vasen noise treshold -30dB gain hystereesi 4dB
        AIC3204_rset(87, 0x82 );
        //Oikea noise treshold -30dB gain hystereesi 4dB
        AIC3204_rset(95, 0x82  );

        //Vasen AGC Maximum Gain Setting 58db
        AIC3204_rset(88, 116  );
        //Oikea AGC Maximum Gain Setting 58db
        AIC3204_rset(96, 116  );


        //Vasen AGC Attack Time = 11*32 word clock
        //Attack Time Scale Factor = 1
        AIC3204_rset(89, 0x28  );
        //Oikea AGC Attack Time = 11*32 word clock
        //Attack Time Scale Factor = 1
        AIC3204_rset(97, 0x28  );

        //Vasen AGC Decay Time = 11*512 ADC Word Clocks
        //Decay Time Scale Factor = 1
        AIC3204_rset(90, 0x28  );
        //Oikea AGC Decay Time = 11*512 ADC Word Clocks
        //Decay Time Scale Factor = 1
        AIC3204_rset(98, 0x28  );

        //vasen noise debounce 16 adc word clock
        AIC3204_rset(91, 4  );
        //oik noise debounce 16 adc word clock
        AIC3204_rset(99, 4  );

        //vasen signal debounce 16 adc word clock
        AIC3204_rset(92, 4 );
        //oik signal debounce 16 adc word clock
        AIC3204_rset(100, 4  );

        //ADC reititys ja käynnistys

        AIC3204_rset( 0, 1 );           // Valitaan sivu 1


AIC3204_rset( 0x34, 0x10 );  // STEREO 1 Jack vas sisääntulo
                             // impedanssi  10kohm micpga +nasta

AIC3204_rset( 0x37, 0x10 );  // STEREO 1 Jack oik sisääntulo
                             // impedanssi  10kohm micpga +nasta

 AIC3204_rset( 0x36, 1 );    //vas kanavan micpga -nasta 10 kohm
 //läpi maihin
AIC3204_rset( 0x39, 0x40 );  //oik kanavat micpga -nasta 10 kohm
//läpi maihin

AIC3204_rset( 0, 0 );        // valitaan sivu 0

AIC3204_rset( 83, 0 );       // Vasen ADC 0db vahvistus
AIC3204_rset( 84, 0 );       // oikea ADC 0db vahvistus
AIC3204_rset( 0x51, 0xc0 );  // virrat päälle vasen ja oikea ADC
```

```
AIC3204_rset( 0x52, 0 );     // äänet päälle vas ja oikea ADC
USBSTK5505_wait( 100 );      // viive

       //DAC reititys ja käynnistys

AIC3204_rset( 0, 1 );        // valitaan sivu 1
AIC3204_rset( 0x0c, 8 );     // LDAC => vasen kuuloke
AIC3204_rset( 0x0d, 8 );     // RDAC => oik kuuloke
AIC3204_rset( 0, 0 );        // valitaan sivu  0
AIC3204_rset( 64, 2 );       // vas volume=oik volume
AIC3204_rset( 65, 0 );       // vasen dac gain = oik dac gain = 0db
AIC3204_rset( 63, 0xd4 );    // käynnistetään oik ja vas DAC
AIC3204_rset( 0, 1 );        // valitaan sivu 1
AIC3204_rset( 0x10, 0x0 );   // äänet päälle vasen kuuloke
AIC3204_rset( 0x11, 0x0 );   // äänet päälle oikea kuuloke
AIC3204_rset( 9, 0x30 );     // virrat päälle vas ja oik kuuloke
AIC3204_rset( 0, 0 );        // valitaan sivu 0
USBSTK5505_wait( 100 );      // viive


/* I2S settings */

I2S0_SRGR = 0x0;
I2S0_CR = 0x8010;    //I2S 16-bit, orja mode, väylä  käyttöön
I2S0_ICMR = 0x3f;    // Sallitaan keskeytykset


while(1)
{
                /* luetaan codecin ADCltä tulevaa dataa */
     // odotettaan että vastaanoton keskeytyslippu laskee
     while((RcvR & I2S0_IR) == 0);
     data3 = I2S0_W0_MSW_R;  // vasemman kanavan 16-bit data
     //muuttujaan
     data4 = I2S0_W1_MSW_R;  // oikean kanavan 16-bit data
     //muuttujaan
     //jos näytteen arvo on liian korkea tulkitaan kuulolle
     //vaaralliseksi
     //ja nostetaan lippu
     if(data3 > 6000 ||data4 > 6000 )
     {
                ohi = 1;
                ohi_laskuri = 100000; //laskee 100000*4us=4sec
     }
     else if(ohi_laskuri == 0 )
     {
       //jos ääni ollut alle asetetun arvon tarpeeksi kauan
                          //lasketaan lippu
                          ohi = 0;
     }
```

```
            //jos lippu ylhäällä, ei lähetetä ääntä takaisin codecille
            if(ohi == 0)
            {
            /* Write Digital audio input */
            //odotettaan että lähetyksen keskeytyslippu laskee
            while((XmitR & I2S0_IR) == 0);
            I2S0_W0_MSW_W = data3;//vasemman kanavan data
            //lähetysrekisteriin
            I2S0_W1_MSW_W = data4;// oikean kanavan data
            //lähetysrekisteriin
            }
            //vähennetään ja pidetään huoli ettei laskuri mene ympäri
            if(ohi_laskuri > 0)
            {
                                    ohi_laskuri--;
            }
            //jos näytteen arvo on liian korkea tulkitaan kuulolle
            //vaaralliseksi
            //ja nostetaan lippu

            //haetaan 5000 kierroksen välein lippu tietoja
                        rget_laskuri++;
                        if(rget_laskuri == 5000)
                        {

                                    AIC3204_rget(93, &agc_gain);
                                    rget_laskuri = 0;
                                    AIC3204_rget(45, &agc_ntf);
                                    rget_laskuri = 0;
                        }
             }
}
Int16 AIC3204_rset( Uint16 regnum, Uint16 regval )
{
            Uint8 cmd[2];
            cmd[0] = regnum & 0x007F;
            cmd[1] = regval;

            return USBSTK5505_I2C_write( AIC3204_I2C_ADDR, cmd, 2 );
}
Int16 AIC3204_rget(  Uint16 regnum, Uint16* regval )
{
    Int16 retcode = 0;
    Uint8 cmd[2];

    cmd[0] = regnum & 0x007F;
    cmd[1] = 0;

    retcode |= USBSTK5505_I2C_write( AIC3204_I2C_ADDR, cmd, 1 );
    retcode |= USBSTK5505_I2C_read( AIC3204_I2C_ADDR, cmd, 1 );

    *regval = cmd[0];
    USBSTK5505_wait( 10 );
    return retcode;
}
```