SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4509

UPRAVLJANJE UGRADBENIM RAČUNALOM ZA BLOKOVSKU OBRADU MJERNIH PODATAKA U STVARNOM VREMENU

Luka Borić

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 16. ožujka 2016.

ZAVRŠNI ZADATAK br. 4509

Pristupnik: **Luka Borić (0036474031)**
Studij: Računarstvo
Modul: Računalno inženjerstvo

Zadatak: **Upravljanje ugradbenim računalom za blokovsku obradbu mjernih podataka u stvarnom vremenu**

Opis zadatka:

U okviru završnog rada potrebno je istražiti mogućnost upravljanja ugradbenim računalom u stvarnom vremenu sa zadaćom blokovske obradbe podataka iz mjernih senzora. Računalo realizirati pomoću evaluacijskog sustava TMS320VC5505 eZdsp temeljnog na procesoru za digitalnu obradbu signala, a kao generator pobudnog signala i kao izvor mjernih podataka koristiti stereo Codec TLV320AIC3204 firme Texas Instruments u sklopu istog razvojnog sustava. Konfigurirati Codec za rad s frekvencijom otipkavanja od 96000 uzoraka u sekundi za A/D i D/A pretvornik. Konfigurirati DMA sklop za automatski prijenos uzoraka pobudnog signala i uzoraka mjernog signala iz memorijskih spremnika prema Codec-u. Dvostruke memorijske spremnike treba konfigurirati za vremenski preklapajući rad, gdje je jedan par spremnika u modu akvizicije, a drugi par spremnika u modu obradbe prethodno prikupljenih podataka i obrnuto. Sustav treba omogućiti i prosljeđivanje obrađenih podataka serijskom sabirnicom preko USB sučelja nadređenom računalu također u stvarnom vremenu, kao i upravljanje s mjernim sustavom korištenjem paralelnog I/O sučelja. Istražiti mogućnost upravljanja sustava s i bez jezgre operacijskog sustava za rad u stvarnom vremenu. Procijeniti i diskutirati potrebne zahtjeve na brzinu izvođenja upravljačkog programa i programa za obradu mjernih podataka kojima se ispunjava uvjet na rad u stvarnom vremenu. Programsku potporu razviti u programskom jeziku C i u assembleru za vremenski kritične odsječke. Za dodatne informacije obratiti se mentoru.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Danko Basch

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Mario Žagar

Zahvala

Zahvaljujem se mentoru prof.dr.sc. Davoru Petrinoviću na pomoći i savjetima tijekom izrade ovog rada, te na strpljivosti i trudu tijekom njegovog mentorstva.

Sadržaj

Popis skraćenica

DSP	digital signal processor
ADC	analog-to-digital converter
DAC	digital-to-analog converter
CCS4	Code composer studio v4
USB	Universal serial bus
SPI	Serial Peripheral Interface
EEPROM	Electrically Erasable Programmable Read-Only Memory
LED	Led emmiting diode
JTAG	Joint test action group
DMA	Direct memory access
ALU	Arithmetic logic unit
MAC	Multiply-Accumulate unit
I_2S	Inter-IC Sound
I_2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver/Transmitter
PLL	Phase-locked loop
LDO	Low-drop regulator
IDE	Integrated development environment
CPU	Control processing unit
RTC	Real time clock
BIOS	basic input/output system
FTDI	Future Technology Devices International

1. Uvod

U sklopu većeg projekta implementacije sustava za mjerenje smjera i brzine vjetra u završnom radu koji slijedi bit će prikazan i objašnjen postupak upravljanja ugradbenim računalom za blokovsku obradu mjernih podataka u stvarnom vremenu.

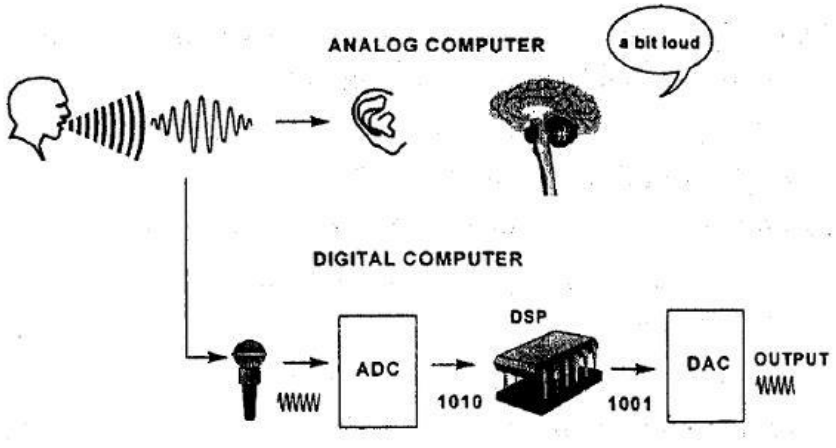
Postupak upravljanja sustavom realiziran je na evaluacijskom razvojnom sustavu TMDX5505EZDSP, proizvođača Texas Instruments, temeljenom na procesoru za digitalnu obradu signala. Glavne komponente razvojnog sustava koje koristimo su TMS320VC5505 procesor za digitalnu obradu signala, skraćenog naziva C5505, i TLV320AIC3204 stereo audio codec, skraćenog naziva AIC3204.

Realizaciju projekta započeo sam proučavanjem sustava za rad u stvarnom vremenu i upoznavanjem s radom i načinom korištenja razvojnog sustava TMDX5505EZDSP. Nakon navedenog krenuo sam s implementacijom alogritama za obradu podataka i kofiguracijom kodeka. Ujedno je trebalo proučiti moguće načine modifikacije razvojne pločice kako bi omogućili serijski prijenos podataka USB portom. Navedeni postupci odredili su i strukturu završnog rada koji slijedi.

2. Sustavi za rad u stvarnom vremenu

Sustavi za rad u stvarnom vremenu (eng. *real-time systems*) su računalni sustavi čiji je zadatak mjerenje, kontrola i reagiranje na utjecaje iz okoline. Sustav je najčešće povezan s okolinom putem senzora, no postoje i drugi različiti ulazni-izlazni uređaji koji predstavljaju sučelje između okoline i sustava. Mjerni podaci se najčešće prenose u blokovima radi lakše i brže obrade. Za sustav kažemo da radi „u stvarnom vremenu“ jer mora zadovoljiti razna vremenska i ostala ograničenja koja mu se nameću zbog povezanosti s vanjskom okolinom. Često ih nazivamo i „reakcijskim sustavima“ jer im je primarna zadaća reagiranje tj. obrađivanje podataka dobivenih iz okoline. Sustav za rad u stvarnom vremenu može biti komponenta većeg sustava u koji je ugrađen te ga zbog toga nazivamo ugradbenim.

Poznato je da su svi signali u prirodi su analogni. Ukoliko želimo vršiti digitalnu obradu signala, prije toga taj signal moramo dobiti u diskretnom obliku. U okviru sustava za digitalnu obradu signala, komponenta koja vrši diskretizaciju signala naziva se analogno-digitalni pretvornik (ADC). Nasuprot procesa diskretizacije signala postoji i proces za pretvaranje digitalnog signala u analogni, za koji je zadužen digitalno-analogni pretvornik (DAC). Obrada signala u stvarnom vremenu podrazumijeva učitavanje signala iz prirode, nekakvu obradu nad učitanim signalom i njegovu reprodukciju.



Slika 1. Sustav za digitalnu obradu signala u stvarnom vremenu

(preuzeto iz http://www.rt-rk.uns.ac.rs/studentski-portal)

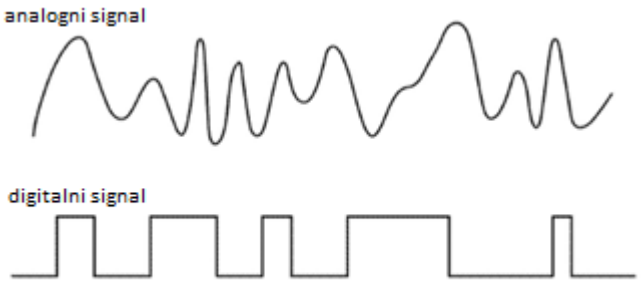
2.1. Procesor za digitalnu obradu signala (DSP)

Digitalni procesor signala (eng. *digital signal processor*), skraćenog naziva DSP, je specijalizirani mikroprocesor s arhitekturom optimiziranom za numeričku manipulaciju i operacije sa signalima. Obično se koristi za mjerenje, filtriranje te reproduciranje kompresiranih kontinuiranih analognih signala iz stvarnog svijeta. Većina procesora opće namijene može obavljati digitalno obrađivanje signala, no DSP-ovi obavljaju te zadatke s većom efikasnošću i preciznosti. Zbog tog razloga su pogodniji u prenosivim uređajima kao što su mobiteli. DSP-ovi obično koriste specijalne arhitekture memorije koje su u stanju dohvatiti veći broj podataka i instrukcija u isto vrijeme.

Algoritmi digitalnog obrađivanja signala tipično sadrže velik broj matematičkih operacija koje moraju biti obavljene brzo i neprestano nad skupom podataka. Signali (najčešće s video ili audio senzora) se konstantno pretvaraju iz analognih u digitalne, obrađuju u digitalnom obliku te pretvaraju nazad u analogne. Velik broj DSP aplikacija ima problema s latentnošću tj. kako bi sustav ispravno radio, DSP mora obaviti sve operacije u fiksnom vremenu.

Slika 2. Postupak digitalne obrade signala

(preuzeto iz https://en.wikipedia.org/wiki/Digital_signal_processor)

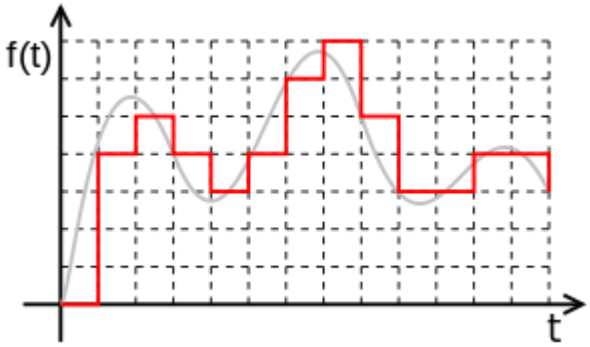


Slika 3. Razlika između analognog i digitalnog signala

2.2. A/D i D/A konverteri

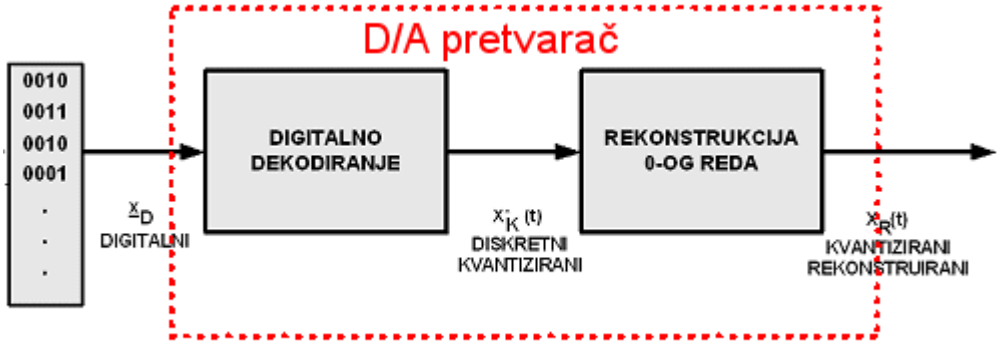
Analogno-digitalni konverter (ADC) je uređaj koji pretvara kontinuiranu fizičku veličinu u njezinu digitalnu reprezentaciju. Pretvaranje uključuje kvantizaciju ulaza što neizbježno rezultira uglavnom malenim pogreškama. ADC pretvorbu ne vrši kontinuirano već periodično, uzorkujući ulaz. Rezultat je sekvenca digitalnih vrijednosti nastalih iz vremenski kontinuiranog analogog signala pretvorenog u diskretni vremenski digitalni signal.

Slika 4. Shematski prikaz A/D konvertera
(preuzeto iz http://laris.fesb.hr/digitalno_vodjenje/text_2-12.htm)



Slika 5. Prikaz pretvorbe
(preuzeto iz http://homes.esat.kuleuven.be/)

Digitalno-analogni konverter (DAC) pretvara digitalne podatke u analogni signal (najčešće struju, napon). DAC provodi inverzni postupak od ADC. Za razliku od analognih signala, digitalni podaci mogu se prenositi, pohranjivati bez gubitaka te zahtjevaju kompleksniju opremu. DAC je potreban da pretvori digitalne podatke u analogni signal koji dovodimo na pojačalo slušalica ili zvučnika kako bi proizveli zvuk.



Slika 6. Shematski prikaz A/D konvertera
(preuzeto iz http://laris.fesb.hr/digitalno_vodjenje/text_2-12.htm)

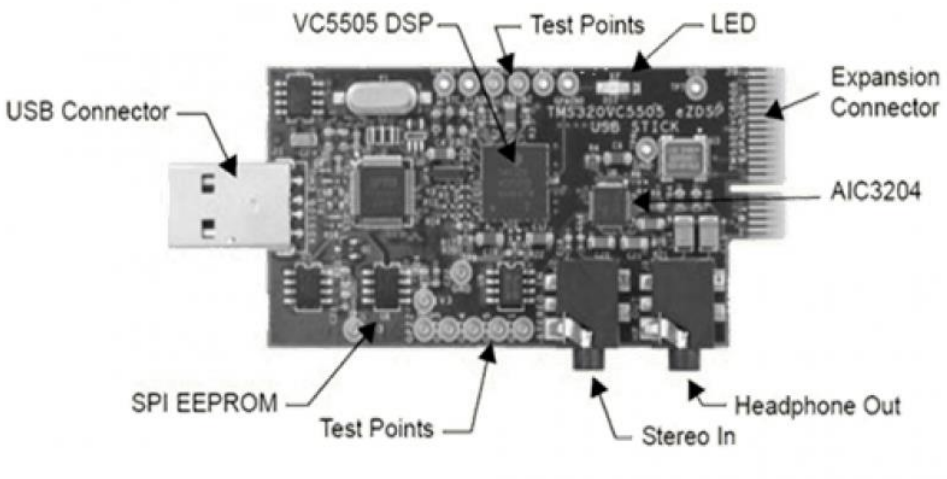
3. Opis razvojnog sustava TMDX5505EZDSP

U ovom poglavlju daje se osvrt na razvojnu okolinu i pripadnu programsku podršku korištenu u implementaciji rješenja.

TMDX5505EZDSP je razvojni sustav građen oko procesora TMS320VC5505. Kako bismo u potpunosti razumjeli rad sustava, potrebno je poznavati njegovu osnovnu građu.

Osnovni dijelovi sustava su :

- TMS320VC5505 procesor za digitalnu obradu signala (VC5505 dsp)
- TLV320AIC3204 stereo audio codec (AIC3204)
- SPI EEPROM memorije kapaciteta 512 Kb
- Korisnički upravljive LED diode
- USB XDS100 JTAG emulatora
- Sučelje za proširenje i testnih priključaka
- USB sučelja za povezivanje s računalom



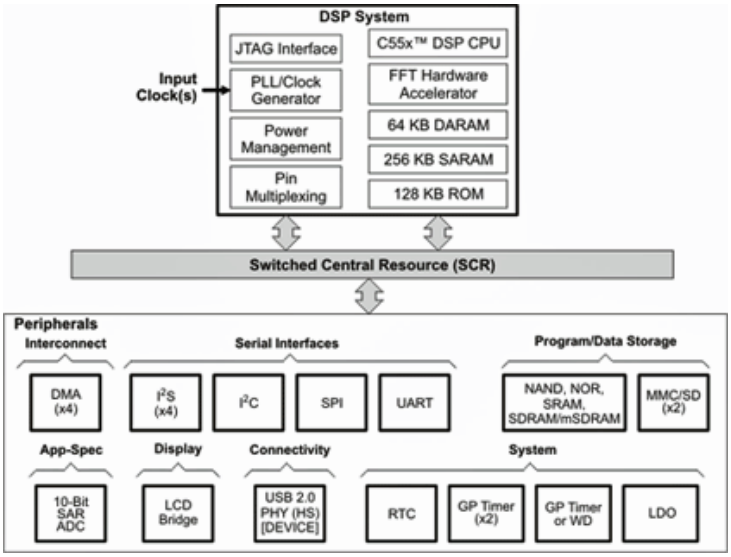
Slika 7. Razvojni sustav TMDX320VC5505
(preuzeto iz http://www.digikey.com/en/product-highlight/t/texas-instruments/tms320vc5505)

3.1. Procesor TMS320VC5505

Kao što je već naznačeno, korišteni razvojni sustav TMDX5505EZDSP izgrađen je oko procesora TMS320VC5505. Procesor TMS320VC5505 pripada porodici C5000 procesora za digitalnu obradu signala (DSP) tvrtke *Texas Instruments*. To je 16-bitni procesor ultra-niske potrošnje, koji za rad s podacima koristi cjelobrojnu aritmetiku. Namijenjen je aplikacijama s malom potrošnjom energije. Temelji se na C55X arhitekturi, koja postiže visoke performanse uz vrlo malu potrošnju energije.

Najvažnije karakteriste TMS320C5535 procesora:

- takt do 100Mhz
- 320Kb on-chip memorije
- interna sabirnička struktura koja omogućava obavaljanje i do četiri 16-bitnih čitanja i dva 16-bitna pisanja u jednom ciklusu
- četiri DMA kontrolera, svaki s četiri kanala
- dvije 17x17 pomnoži-i-zbroji (engl. Multiply-Accumulate units (MAC)) jedinice
- središnja 40-bitna ALU jedinica potpomognuta dodatnom 16-bitnom ALU
- standardne periferije: I_2C, I_2S, UART...
- velike mogućnosti paralelizacije instrukcija



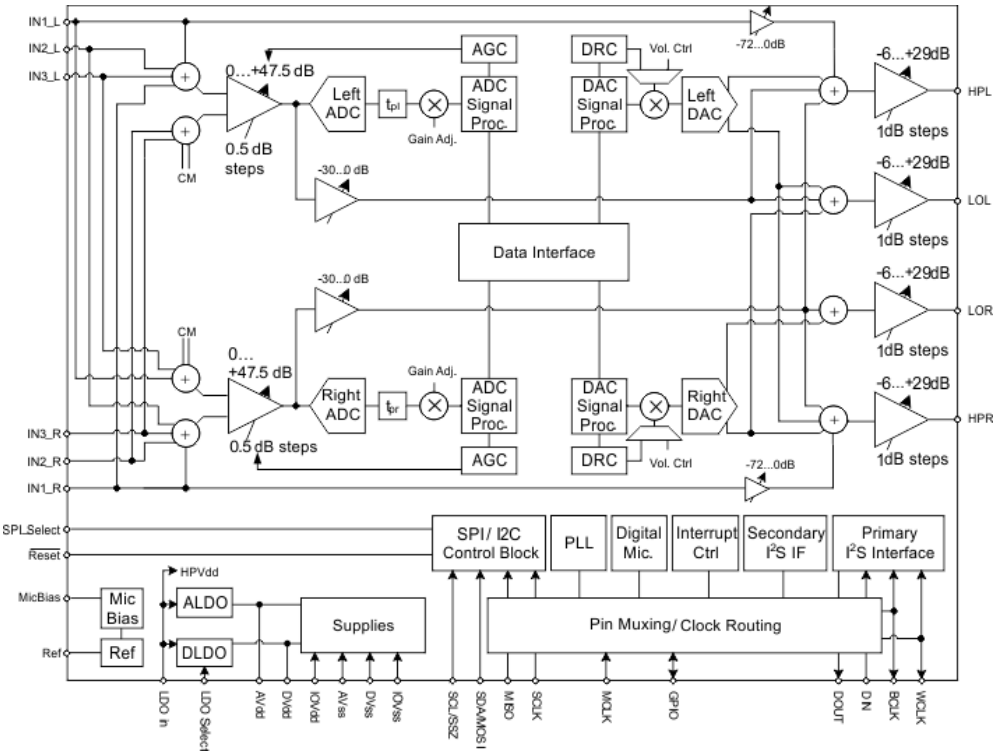
Slika 8. Block diagram DSP-a (preuzeto iz [3])

3.2. TLV320AIC3204 stereo audio kodek

TLV320AIC3204, skraćenog naziva AIC3204, fleksibilan je, niskopotrošni stereo audio kodek s programljivim ulazima i izlazima, mogućnošću kontrole pojačanja i konfiguracije signala otipkavanja. Kodek pruža i izrazito korisne mogućnosti powertuninga koje olakšavaju pronalaženje balansa između štednje energije i performansi. Sadrži fiksne predefinirane procesne blokove, integrirani PLL, integrirani LDO i fleksiblina digitalna sučelja.

Najvažniji dijelovi kodeka su njegovi analogno–digitalni konverteri (ADC) i digitalno–analogni konverteri (DAC) koji vrše pretvorbu signala koji prolazi kroz sustav. Signali u sustav ulaze i izlaze kroz stereo priključke koji se nalaze na površini pločice.

Zbog svoje fleksibilnosti kodek je pogodan za široki spektar primjena i lako prilagodljiv aplikaciji kojoj je namijenjen. Idealan je za baterijski pogonjene prijenosne audio (npr. mp3 player) i telefonske aplikacije. Područje rada mu je od 8kHz mono do 192kHz stereo reprodukcije.

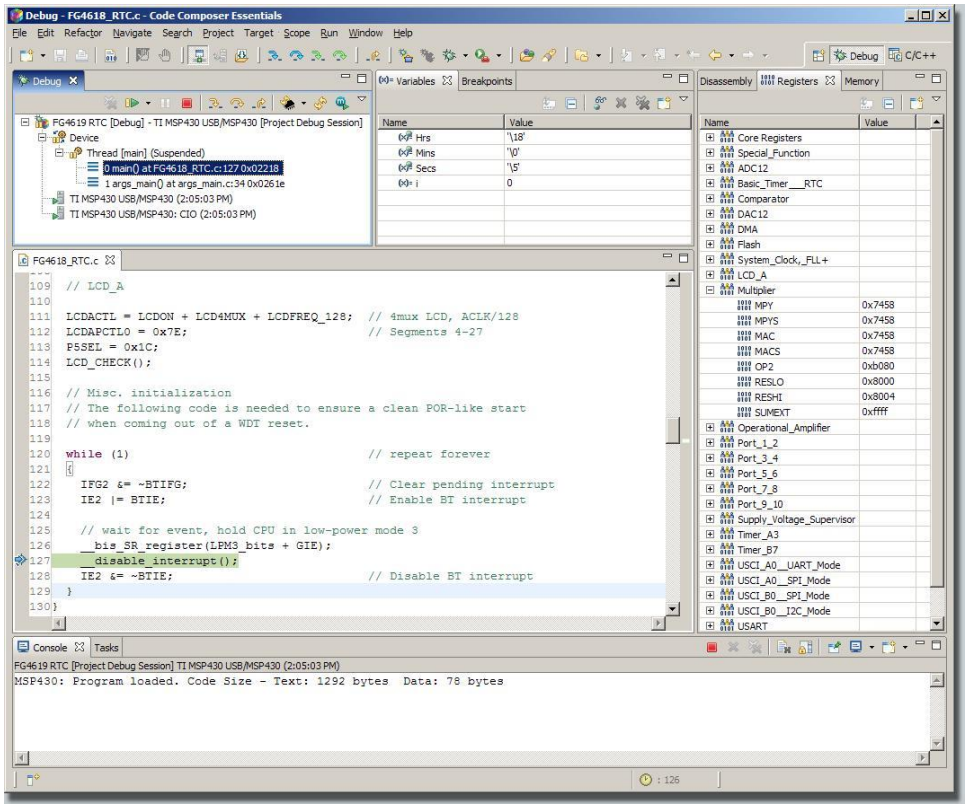


Slika 9. Block diagram AIC320 (preuzeto iz [5])

3.3. Code Composer Studio v4 IDE

Programska podrška za sustav pisana je u Code Composer Studio v4 IDE razvojnom okruženju. Code Composer Studio v4, skraćenog naziva CCS4, predstavlja integrirano razvojno okruženje (IDE) za Texas Instruments mikrokontrolere i namijenske procesore. CCS4 nudi set alata koji omogućavaju razvoj i analizu namijenske programske podrške. Spomenuti set alata uključuje C/C++ programski prevoditelj, asembler, povezivač, prozore za uređivanje izvornog koda, alat za kontrolirano izvršavanje koda (eng. *debugger*), alat za profiliranje koda (eng. *profiler*) i mnoge druge.

CCS4 razvojno okruženje zasnovano je na *Eclipse* platformi. *Eclipse* je široko korišten kao osnova mnogih razvojnih okruženja. Otvorenog je koda (eng. *open source*) i nudi mnoštvo korisnih alata za razvoj programske podrške. CCS4 proširenja u *Eclipse*-u odnose se na omogućivanje upotrebe posebnih odlika namjenskih uređaja koje je proizveo *Texas Instruments*.



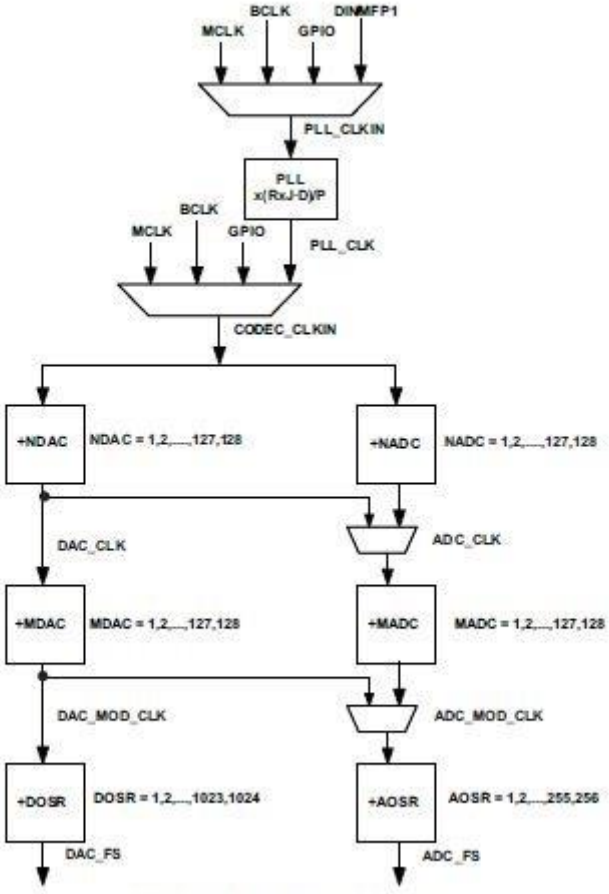
Slika 10. Primjer prikaza raznih prozora CCS4

4. Konfiguracija kodeka

4.1. Opis postupka

Kako bi prilagodili rad kodeka našoj primjeni, potrebno je kodek ispravno inicijalizirati i konfigurirati. Dio koda koji se odnosi na konfiguraciju i inicijalizaciju kodeka pisan je u programskom jeziku Asembler.

A/D i D/A pretvornike kodeka potrebno je podesiti za rad s frekvencijom otipkavanja od 96000 uzoraka u sekundi. Slika 11. prikazuje funkcijski blok dijagram sklopovlja za prilagodbu signala takta koji se dovodi na A/D i D/A pretvornike.



Slika 11. Sklopovlje za prilagodbu otipkavanja *(preuzeto iz [4])*

Prilikom podešavanja ADC i DAC potrebno je prvo na njih dovesti željeni takt otipkavanja (eng. *clock*). Referentni signal takta moguće je na uređaj dovesti kroz mnoštvo raznih pinova, kao što su BCLK, MCL ili GPI pinovi. U slučaju da željeni takt ne može biti generiran iz referentnog signala takta (BCLK, MCL, GPI), TLV320AIC3204 pruža mogućnost upotrijebe on-chip PLL-a koji podržava širok raspon frakcijskih množitelja za generiranje željenog signala otipkavanja. U rješenju zadatka na on-chip PLL propušten je signal sa pina MCLK. PLL je sklop koji na svom izlazu generira signal koji je u istoj fazi kao i signal kojeg prima na ulazu.

Frekvencija fazno vezane petlje određena je ulaznom frekvencijom PLL_CLKIN i parametrima P, R, J i D, što prikazuje izraz (1.):

$$CODEC\_CLKIN = PLL\_CLK = \frac{PLL\_CLKIN * R * J.D}{P} \quad (1.)$$

Signal CODEC_CLKIN, koji sada predstavlja signal takta, multipleksiranjem se usmjerava kroz niz visoko fleksibilnih djelitelja (eng. *clock dividers*) koji se koriste za generiranje potrebnih frekvencija otipkavanja za željeni rad ADC-a i DAC-a. Djelitelji NDAC, NADC, MDAC, MADC, DOSR i AOSR služe kako bi podijelili izvorni signal takta željeni broj puta i tako namjestili frekvenciju otipkavanja ulaznog (ADC) i izlaznog (DAC) takta otipkavanja.

Ulazni takt otipkavanja računa se izrazom (2.):

$$ADC\_FS = \frac{CODEC\_CLKIN}{NADC * MADC * AOSR} \quad (2.)$$

gdje je ADC_FS ulazni takt otipkavanja, CODEC_CLKIN frekvencija dobivena iz fazno vezane petlje, a NADC, MADC i AOSR korišteni djelitelji.

Izlazni takt otipkavanja računa se izrazom (3.):

$$DAC\_FS = \frac{CODEC\_CLKIN}{NDAC * MDAC * DOSR} \quad (3.)$$

gdje je DAC_FS izlazni takt otipkavanja, CODEC_CLKIN frekvencija dobivena iz fazno vezane petlje, a NDAC, MDAC i DOSR korišteni djelitelji.

4.2. Postupak konfiguriranja

U nastavku su detaljno prikazani dijelovi koda koji se odnose na inicijalizaciju kodeka za rad s frekvencijom otipkavanja 96 khz za ADC i DAC. Važno je napomenuti da je AIC3204 registarski programljiv pa se konfiguriranje kodeka svodi na popunjavanje određenih registara odgovarajućim vrijednostima.

U prikazu konfiguracije korišten je skraćeni zapis programskog koda u Asembleru.
- oznacava liniju komentara
w - oznacava I_2C nardebu za pisanje iza koje slijedi adresa uređaja, adresa I_2C registra i vrijednost koju zapisujemo

U prvom koraku izabiremo rad sa registrima stranice 0, te programski resetiramo uređaj kako bi inicijalizirali sve registre na njihove početne vrijednosti. Početne vrijednosti svih registara i njihove namjene mogu se vidjeti u TLV320AIC3204 Application Reference Guide [4]

```
################################################
# Software Reset
################################################
#
# Select Page 0
w 30 00 00
#
# Initialize the device through software reset
w 30 01 01
#
################################################
```

Sljedeći korak je odabiranje procesnih blokova potrebnih za ispravan rad konvertera. Odabiremo procesne blokove filtra B koji je najpogodniji za uzorkovanje frekvencijom 96 khz

```
################################################
# Configure Processing Blocks
################################################
#
# Select Page 0
w 30 00 00
#
# PRB_P17 and PRB_R13 selected
w 30 3C 11 0d
#
################################################
```

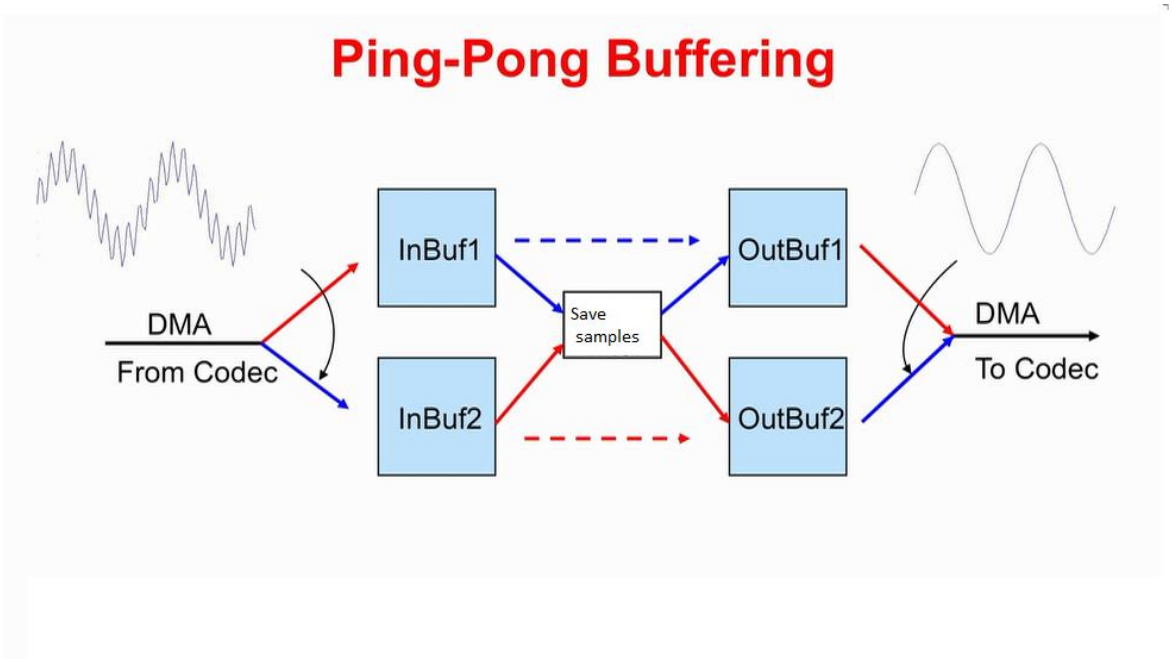
Kao ulaz referentnog signala takta izabire se pin MCLK te se on propušta na PLL. Postavljamo vrijednosti parametara P, R, J, D. Izabiremo *master* način rada kodeka te namještamo djelitelje NDAC, MDAC, NADC, MADC, AOSR, DOSR.

```
################################################
# Clock and Interface Settings
# ------------------------------------------
# The codec receives: MCLK = 12.288 MHz,
################################################
#
# PLL_clkin = MCLK, codec_clkin = PLL_CLK,
# PLL on, P=1, R=1, J=8, D=1920
w 30 04 03 91 08 07 80
#
# NDAC = 2, MDAC = 8, dividers powered on
w 30 0b 82 88
#
# DOSR = 64
w 30 0D 00 40
#
# NADC = 2, MADC = 8, dividers powered on
w 30 12 82 88
#
# AOSR = 64
w 30 14 40
#
################################################
```

5. Programiranje procesora

5.1. Opis programa na DSP-u

Program implementira algoritam za blokovsku obradu podataka dobivenih s mjernih senzora u stvarnom vremenu. Program ulazne podatke dobiva kao uzorke izmjerene na senzorima koji su na kodek spojeni preko stereo inputa. Podaci se ADC-om uzorkuju frekvencijom 96 kHz na svom ulasku u kodek te prosljeđuju DSP-u na daljnju obradu. Uzorci prolaze kroz DSP te se vraćaju nazad u kodek gdje pri izlasku iz kodeka prolaze kroz DAC koji također radi na 96 kHz. Uređaj koristi jedan od četiri DMA kontrolera za slanje uzoraka između kodeka i DSP-a preko I_2S serijske sabirnice. Svaki DMA kontroler ima četiri kanala. U našem slučaju dva kanala koriste se za primanje desnih i lijevih uzoraka s kodeka, a preostala dva za slanje lijevih i desnih uzoraka nazad kodeku. Ovi uzorci putuju I_2S sabirnicom u full-duplex modu bez intervencije CPU-a. Full-duplex označava prijenos podataka u oba smjera istovremeno. Kada se nalaze u DSP-u, uzorci prolaze kroz ping-pong spremnike (eng. *ping-pong buffers*).



Slika 12. Ping-pong buffering (preuzeto iz [1])

Ping-pong buffering sustav sastoji se od dva ulazna i dva izlazna buffera koji svaki DMA ciklus mijenjaju uloge. U prvi buffer InBuf1 kontinuirano se prikupljaju podaci iz kodeka, dok se u InBuf2 čuvaju prethodno prikupljeni podaci. Ti prethodno prikupljeni podaci istovremeno se šalju u izlazni buffer OutBuf2. Kada DMA završi s punjenjem InBuf1, bufferi mijenjaju uloge i nastavljaju s radom. S druge strane, dok se OutBuf2 puni prikupljenim podacima, OutBuf1 šalje prethodno prikupljene podatke nazad kodeku. Nakon što OutBuf1 pošalje sve podatke kodeku, bufferi također mijenjaju uloge i nastavljaju s radom. Podaci se najčešće u međukoraku između slanja s ulaznog na izlazni buffer spremaju u memoriju kako bi se u budućnosti mogli koristiti za druge upotrebe.

Dio koji slijedi detaljno prikazuje dijelove koda koji se odnose na glavnu funkcionalnost DSP-a. Kako je program DSP-a glavni program iz kojeg se pozivaju i inicijaliziraju svi ostali dijelovi sustava, glavni program mora pozvati funkcije koje ih inicijalizaciju.

Prije početka rada procesora potrebno je inicijalizirati PLL. PLL inicijaliziramo za rad s frekvencijom od približno 98 Mhz.

```
PLL_98Mhz();
```

Početak rada procesora započinje učitavanjem mjernog signala. Signal se s 3.5 mm ulaza preko A/D pretvornika učitava u kodek. Kako bi mogli čitati s kodeka potrebno ga je inicijalizirati.

```
AIC3204_Init ();
```

Kako bi omogućili prijenos podataka između kodeka i DSP-a potrebno je inicijalizirati i DMA i I_2S sabirnicu kojom će se podaci prenositi. I_2S sabirnica za procesor se inicijalizira u slave modu što znači da takt za rad dolazi od kodeka. Također je potrebno u interrupt enable registrima (IER0, IER1) omogućiti prekide koji dolaze od DMA i Real Time Clock-a (RTC).

```
IER0 = 0x0110;
IER1 = 0x0004;
```

```
setDMA_adress();
enable_dma_Int();

set_i2s0_slave();
enable_i2s0();
```

Ujedno je potrebno pokrenuti sva četiri kanala DMA kontrolera potrebna za prijenos ulaznih i izlaznih podataka između procesora i kodeka.

```
set_dma0_ch0_is20_Lout();
set_dma0_ch1_is20_Rout();
set_dma0_ch2_is20_Lin();
set_dma0_ch3_is20_Rin();
```

Na kraju potrebno je inicijalizirati i Real Time Clock (RTC).

```
enable_rtc_second_int();
```

Kako bi vizualno mogli lakše pratiti rad sustava, ovisno koja sekvenca koda se izvršava na DSP-u palit će se i gasiti LED dioda koja se nalazi na pločici. Funkcije koje se brinu za paljenje i gašenje LED diode su sljedeće:

Program za paljenje diode:

```
Void turnOnLED(void)
{
      Uint16 temp;
      Temp = ST1_55;

      If((temp&0x2000) == 0 )
            {
                  temp |= 0x2000;
                  ST1_55 = temp;
            }
}
```

Program za gašenje diode:

```
Void turnOffLED(void)
{
      Uint16 temp;
      Temp = ST1_55;
```

```
            If((temp&0x2000) != 0 )
                  {
                        temp |= 0xDFFF;
                        ST1_55 = temp;
                  }
      }
```

Prolazak mjernih uzoraka kroz DSP, odnosno manipuliranje dinamikom podataka koji putuju ping-pong bufferima odvija se u velikoj *while* petlji. Ova petlja izvršava se beskonačno mnogo puta kako bi naš sustav neprekidno izvršavao željenu funkcionalnost. U ovom radu lijeve uzorke poslat ćemo na izlaz u sekvenci (N,N,0), a desne ćemo samo proslijediti direktno na izlaz. Na ovaj način, ako kao ulazni signal u sustav pustimo glazbeni signal jasno će se čuti razlika u obrađivanju uzoraka na lijevoj i desnoj slušalici.

Lijeve ulazne uzorke na izlaz šaljemo u sekvenci N,N,0 gdje N označava veličinu buffera (N = XMIT_BUFF_SIZE = 100). Prije nego što ih pošaljemo na izlaz, mjerne uzorke spremamo u pomoćno polje na korištenje ostalim algoritmima kojima su potrebni za izračun brzine i smjera vjetra. Varijabla „br“ predstavlja brojač pomoću kojeg znamo u kojem smo stanju sekvence N,N,0. Sljedeći kod odnosi se samo na prijenos podataka kada je aktivan DMA kanal 2. Kod za drugi ulazni DMA kanal 1 je ekvivalentan.

```
while(1) {
      br = 0;
      for(i=0; i<100; i++) {
            niznula[i] = 0;
      }
      if(RunForL == 1)
      {
            RunForL = 0;
            If(currentRcvL_DMAchannel == 2)
            {
                  buffcopy(&RcvL1[0], &spremi[0], XMIT_BUFF_SIZE);
                  if (br<2) {
                        turnOnLED();
```

```
                        buffcopy(&spremi[0],$OutL1[0], XMIT_BUFF_SIZE);
                        br++;
                  else {
                        turnOffLED();
                        buffcopy(&niznula[0],$OutL1[0],XMIT_BUFF_SIZE);
                        br = 0;
                        }
            }
```

Desne ulazne podatke direktno prosljeđujemo na izlaz.

```
if(RunForR == 1)
      {
            RunForR = 0;
            If(currentRcvL_DMAchannel == 2)
            {
                  buffcopy(&RcvR1[0], &OutR1[0], XMIT_BUFF_SIZE);
            }
            else
            {
                  buffcopy(&RcvR2[0], &OutR2[0], XMIT_BUFF_SIZE);
            }
}
```

Funkcija „*buffcopy“* jedna je od najvažnijih funkcija i njezin je zadatak premiještanje uzoraka s ulazne strane na izlaznu. Utemeljena je na radu s pokazivačima koji pokazuju na adrese ulaznih i izlaznih buffera. Kod funkcije „*buffcopy“* izgleda ovako:

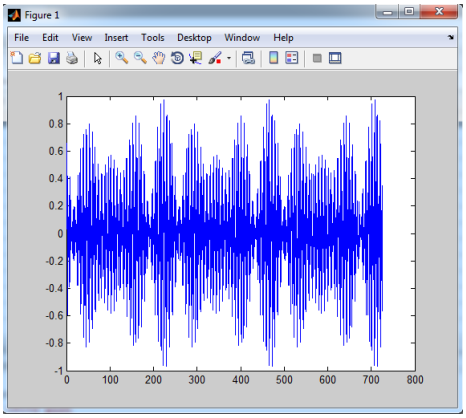
```
void buffcopy(Int16 *input, Int16 *output, int16 size) {
      Int16 i;
      for (i=0; i<size; i++)
      {
            *(output + i ) = *(input + i);
      }
}
```

Kako je zadaća našeg sustava obrađivati signale koji se koriste za izračun brzine i smjera vjetra, radi testiranja sustava u Matlab-u smo generirali jedan takav signal. Signal je u Matlab-u kreiran na sljedeći način :

```
si1=(amp1(1)*cos(w1(1)*ni+fi1_init(1))+amp1(2)*cos(w1(2)*ni+fi1_init(2
))+amp1(3)*cos(w1(3)*ni+fi1_init(3)));
```

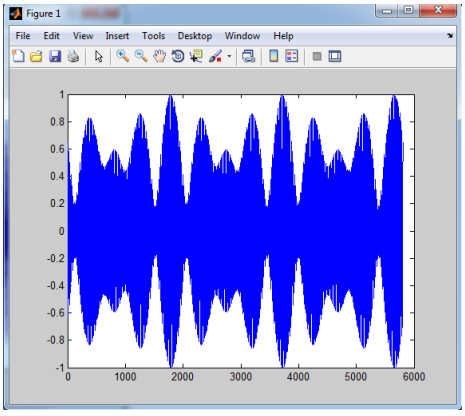
prvih 30 vrijednosti signala s1 su :

0,656914584344374	-0,594791147497120	0,415221161321561	-0,170910628153746
-0,0773077361636385	0,276007666965624	-0,390973541663505	0,412594485209475
-0,354163109739607	0,244416780212768	-0,117136770550811	0,00116679165947864
0,0862690782427313	-0,141369700295254	0,170782567004243	-0,185481324638791
0,194037155909772	-0,197864417502063	0,190203745063835	-0,159216925750926
0,0940648864357669	0,00828759476360313	-0,138179065569241	0,272408225044914
-0,378332826093482	0,422188139103372	-0,379417669061202	0,243928785556523
-0,0332156160491251	-0,212729821109985		



Slika 13. Signal s1 u Matlab-u

Na izlazu iz sustava signal izgleda ovako :



Slika 14. Signal s1 u Matlab-u

Prije nego što procesor počne obrađivati podatke potrebno je članove ulaznog niza s1 pretvoriti u q15 format množenjem sa 2^{15}. Tada će vrijednosti svih brojeva biti umjesto iz intervala (-1,1) u intervalu od $(-2^{15}, 2^{15})$

5.2. TMDX5505EZDSP kao samostalni uređaj

Kako bi razvojni sustav mogao raditi kao samostalni uređaj, potrebno je programski kod, koji sadrži konfiguracije i opis ponašanja sklopa, zapisati u SPI EEPROM programsku memoriju. Prvi korak je u mapi projekta pronaći .out file i konvertirati ga u .bin oblik (*bootimage*).

Skriptu „*hex55“* potrebnu za provedbu konverzije moguće je skiniti sa sljedeće poveznice :

https://wwwa.ti.com/downloads/sds_support/TICodegenerationTools/download.htm#C5500

Konverzija se u cmd konzoli provodi upisivanjem sljedećeg:

hex55 -i [**input file name**] -o [**ouput file name**] -boot -v5505 -b -serial8

Konzola bi nakon uspješnog stvaranja bootimage-a trebala dati sljedeći ispis :

```
"ezdsp.out" ==> .cinit <BOOT LOAD>

"ezdsp.out" ==> vectors <BOOT LOAD>

"ezdsp.out" ==>.text  <BOOT LOAD>

"ezdsp.out" ==> .const.1  <BOOT LOAD>

"ezdsp.out" ==> .const.2  <BOOT LOAD>

"ezdsp.out" ==> .const.3  <BOOT LOAD>
```

Kada stvorimo bootimage, možemo ga programirati u SPI EEPROM koristeći pomoćni flashing program „programmer_USBKey.out“. Pomoćni program dostupan je na sljedećoj poveznici:

https://code.google.com/archive/p/c5505-ezdsp/

Potrebno je pokrenuti Code Composer Studio i slijediti sljedeće korake :

Target -> Launch TI Debug

Target -> Connect Target

Target -> Load Program...

Učitati „programmer_USBKey.out“ file.

Pokrenuti program i učitati cijeli put (eng. *full path*) do direktorija u kojem se nalazi bootimage. Primjer takvog puta je:

C:\temp\ezdsp.bin

Program će započeti zapisivanje koda u memoriju. U konzoli Code Composer Studia ispisuje se sljedeće:

SPI EEPROM

Writing data to device...

Opening c:\temp\led.bin

Input file opened

Programming Complete

Nakon ovog koraka uspješno je zapisan program u memoriju. Pločicu izvadimo iz laptopa te ju nakon 10 sekundi vratimo. Sada je pločici za rad potrebno samo napajanje iz laptopa ili drugih uređaja, a program se na njoj odvija samostalno i neovisno o nadređenom uređaju.

5.3. Upravljanje sustavom pomoću jezgre operacijskog sustava (DSP/BIOS)

DSP/BIOS je skalabilna jezgra za rad sustava u stvarnom vremenu. Dizajnirana je za primjene u aplikacjima koje zahtijevaju sinkronizaciju i rad sa zakazanim događajima, host-to-target komunikaciju ili instumentaciju za rad u stvarnom vremenu. DSP/BIOS pruža mogućnosti preventivnog multi-threadinga, apstrakcije hardwarea, analize u stvarnom vremenu i razne konfiguracijske alate. DSP/BIOS je dizajnirana u svrhu minimizacije CPU zahtjeva na ciljani uređaj. Taj cilj postignut je na sljedeće načine:

- Svi DSP/BIOS objekti mogu se konfigurirati statički i ograničiti da rade kao „*executable program image*“. Ovo smanjuje veličinu koda i optimizira internu strukturu podataka.

- Instumentacijski podaci (kao *log* i *trace*) formatiraju se na uređaju domaćinu

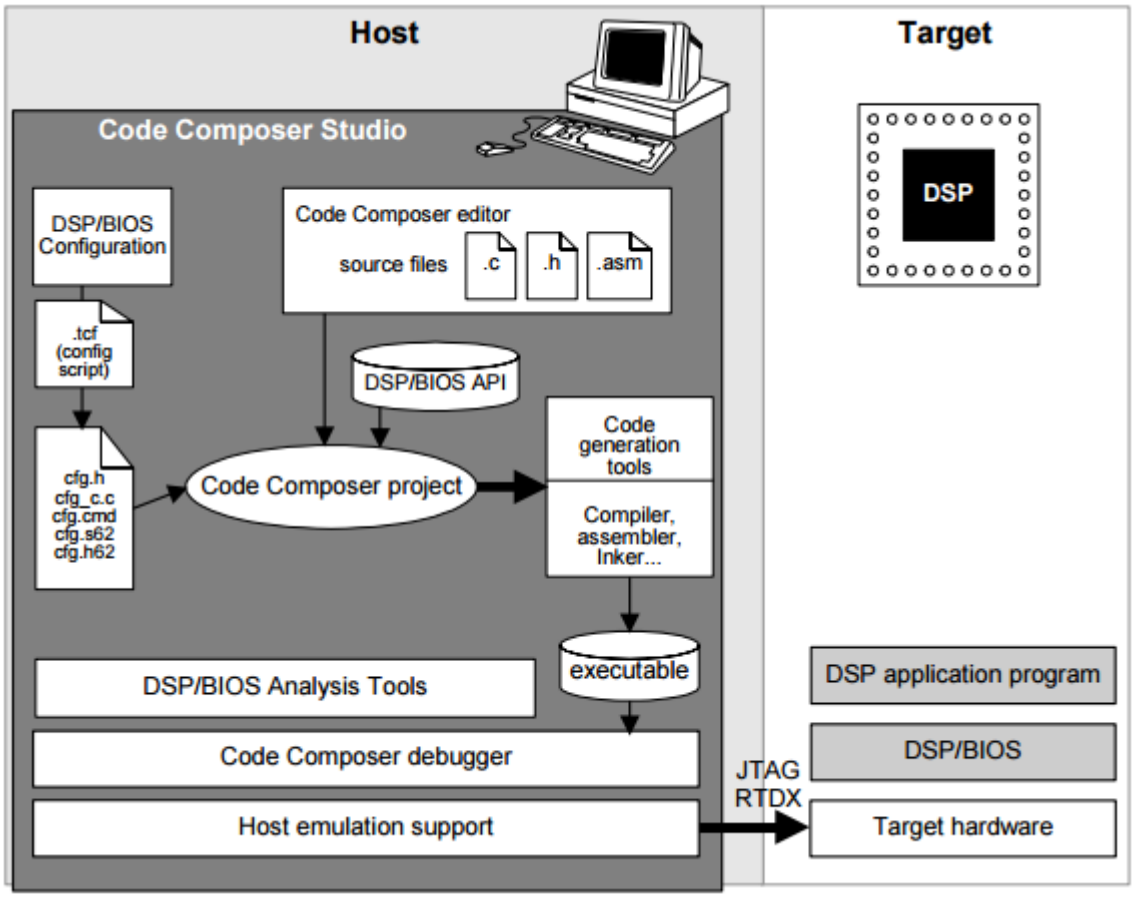
- API su modulirani tako da samo se samo oni koje koristi program suzbijaju u izvršni program

- Knjižnice (eng. *library*) su optimizirane tako da zahtjevaju najmanji mogući broj instrukcijskih ciklusa, sa većim djelom implementiranim u zbirnom jeziku (eng. *assembly language*)

- Komunikacija između ciljnog uređaja i DSP/BIOS alata za analizu izvodi se kroz pozadinsku petlju koja je u pasivnom načinu rada (eng. *idle mode*). Ovo osigurava da DSP/BIOS alati za analizu ne zadiru u rad programa. Ako je CPU ciljnog uređaja prezaposlen da obavlja pozadinske zadaće, DSP/BIOS prestaje primati podatke dok se CPU ne oslobodi

- Provjeravanje grešaka, koje zahtjeva rad CPU-a, drži se na minimumu. Umjesto toga, API dokumentacija navodi ograničenja pri pozivu API funkcija te je odgovornost programera da ih se drži

Slika 13. prikazuje komponente DSP/BIOS-a tijekom generiranja programa i pronalaženja pogrešaka.



Slika 15. DSP/BIOS (preuzeto iz [8])

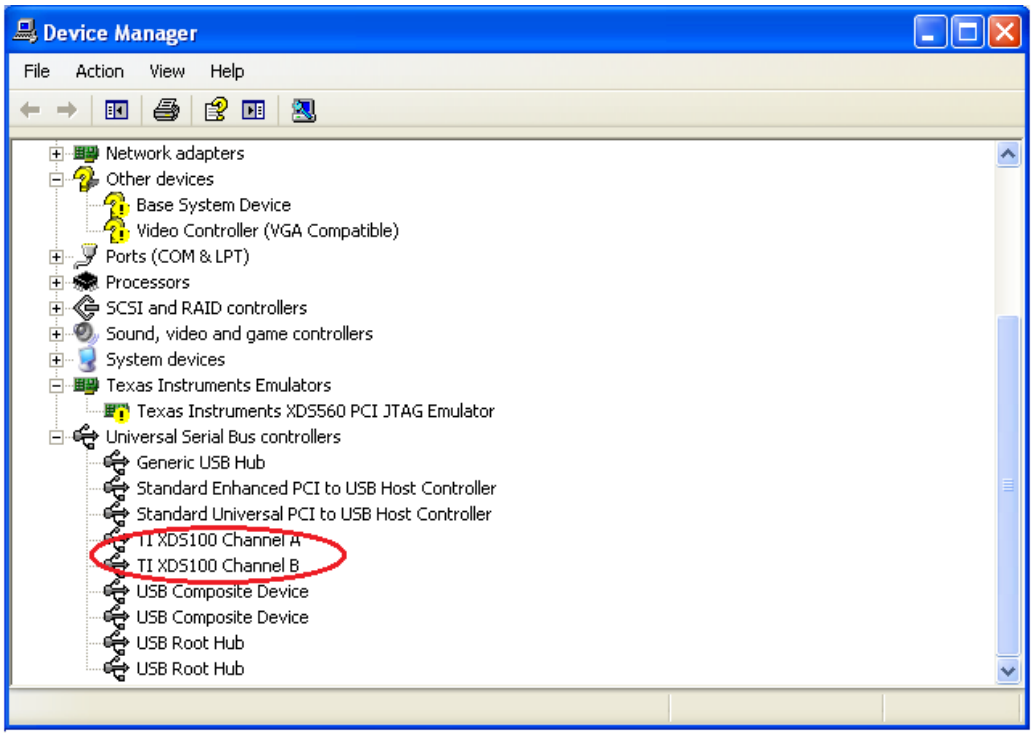
Osnovne komponente DSP/BIOS-a su:

- **DSP/BIOS API –** na domaćinskom PC-u potrebno je napisati program (C, C++, Asembler) koji poziva funkcije DSP/BIOS API-a
- **DSP/BIOS konfiguracijski alat** – služi za stvaranje konfiguracije koja definira statičke objekte koje će korisnički program koristiti
- **DSP/BIOS alat za analizu** – ovi alati pružaju mogućnost testiranja korisničkog programa na ciljnom uređaju uz praćenje rada CPU-a, praćenje vremena, praćenje pojedinih dretvi i dr.

6. Modifikacija za slanje podataka USB portom

Zahtijevniji dio zadatka bio je ostvariti slanje obrađenih podataka nadređenom uređaju. Proučavanjem sheme sklopa i čitanjem službenih dokumenata, moguće je uočiti da je USB-UART konverter neiskorišten. UART pinovi DSP-a nisu povezani s Rx/Tx (transmit/recieve) UART pinovima FTDI čipa. Povezivanjem ovih pinova u teoriji bi uspješno omogućili podršku za USB-serial port. U nastavku je detaljno opisan postupak povezivanja Rx/Tx UART pinova slobodnog kanala B FTDI-a na UART pinove DSP na konektoru ekspanzije.

Program za pronalaženje pogrešaka (eng. *debugger*) izgrađen je oko FTDI uređaja (FT2232D). FTDI je kompozitni uređaj na čipu i sadrži dva sučelja. Jedno od sučelja koristi se kao USB serijski port, dok se drugo koristi kao USB–JTAG adapter. Kanal A koristi se kao JTAG, dok je kanal B korišten kao USB-serial port emulator.



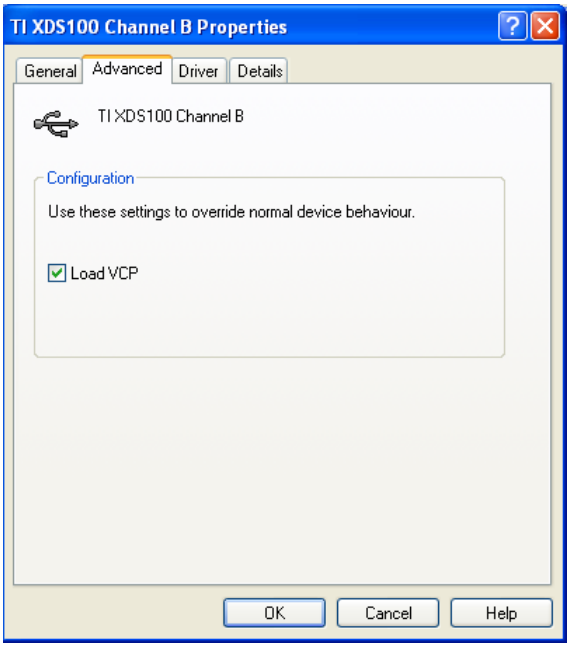
Slika 16. Prikaz kanala A i B

Napomenuto je da pločici nedostaje UART veza DSP-a s FT2232D čipom koji se koristi za debuggiranje, što bi omogućilo korištenje USB porta. Pogledom na shemu FT2232D-a, moguće je vidjeti da se pinovi kanala B korišteni kao Tx/Rx za UART nalaze na pinovima 40 i 39.

Pin#	Generic Pin name	Pin Definitions by Chip Mode **Note 2						
		232 UART Mode	245 FIFO	Enhanced Asynchronous and Synchronous Serial	MPSSE **Note 4	MCU Host Bus Emulation Mode **Note 5	Fast Opto-Isolated Serial Mode	CPU FIFO Interface Mode
40	BDBUS0	TXD	D0	D0	A8	FSDI	D0	D0
39	BDBUS1	RXD	D1	D1	A9	FSCLK	D1	D1
38	BDBUS2	RTS#	D2	D2	A10	FSDO	D2	D2
37	BDBUS3	CTS#	D3	D3	A11	FSCTS	D3	D3
36	BDBUS4	DTR#	D4	D4	A12	**Note 3	D4	D4
35	BDBUS5	DSR#	D5	D5	A13	D5		D5
33	BDBUS6	DCD#	D6	D6	A14	D6		D6
32	BDBUS7	RI#	D7	D7	A15	D7		D7
30	BCBUS0	TXDEN	RXF#	WR# **Note 9	CS#	CS#		CS#
29	BCBUS1	SLEEP#	TXE#	RD# **Note 9	ALE	A0		A0
28	BCBUS2	RXLED#	RD#	WR# **Note 7	RD#	RD#		RD#
27	BCBUS3	TXLED#	WR	RD# **Note 7	WR#	WR#		WR#
26	SI/WUB	SI/WUB	SI/WUB	SI/WUB	**Note 8	**Note 8	SI/WUB	**Note 8
40	BDBUS0	TXD	D0	D0	A8	FSDI	D0	

Slika 17. Prikaz pinova Rx i Tx kanala B

Signali kanala A koriste za debugiranje dok se signali kanala B uopće ne koriste. Sljedeći korak je u *device manageru* dodijeliti kanalu B UART funkcionalnost.

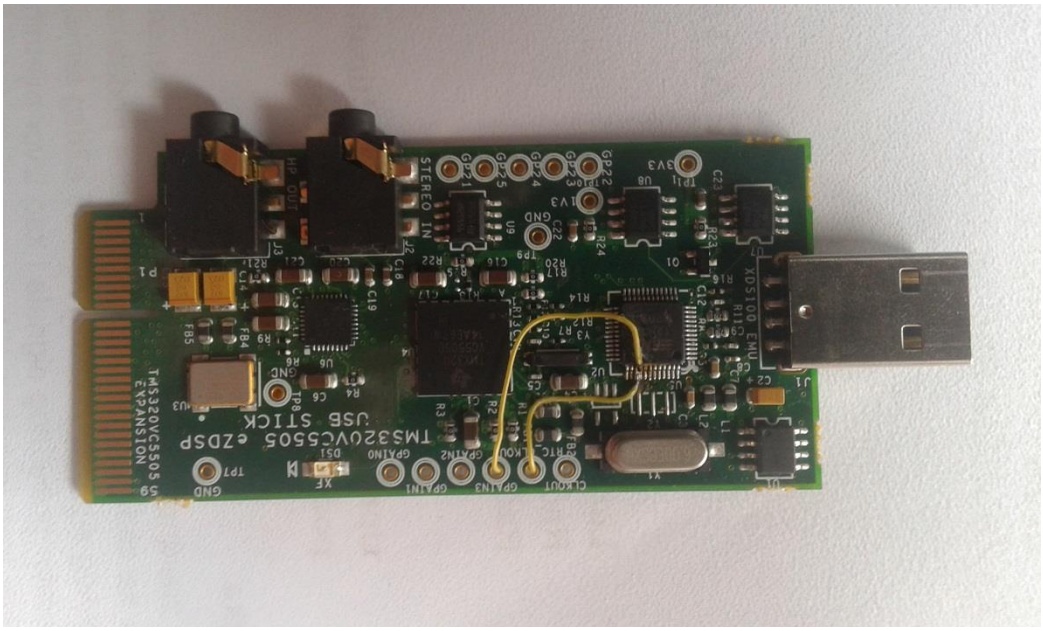


Slika 18. Dodjeljivanje funkcije kanalu B

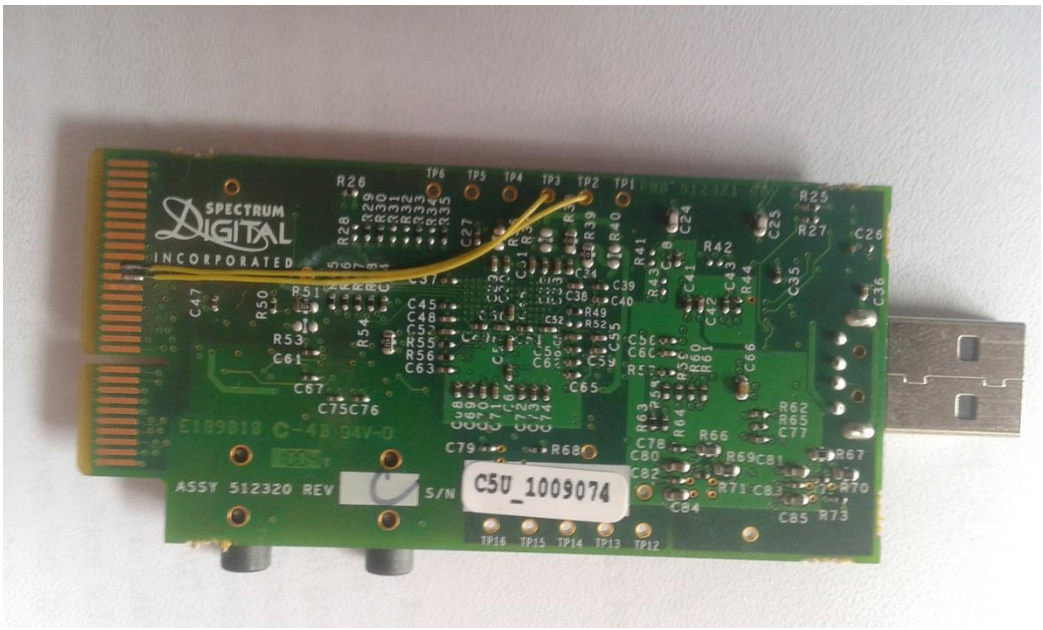
Provjeravanjem sheme DSP-a vidimo da se UART signali DSP-a nalaze na priključcima za proširenje (eng. *expansion connector*) pločice. UART Tx nalazi se na pinu broj 40, dok se UART Rx nalazi na pinu broj 38. Dakle, sljedeći korak je povezati pin 40 FTDI-a sa pinom 38 konektora ekspanzije DSP-a, te pin 39 FTDI-a sa pinom 40 konektora ekspanzije DSP-a.

(DSP) pin 40 (Tx) < --- > (FTDI) pin 39 (Rx)
(DSP) pin 38 (Rx) < --- > (FTDI) pin 40 (Tx)

Vezu je ostvarena kratkim spajanjem navedenih pinova žicom i tako uspostvaili UART konekciju koja do sada nije postojala. Na slikama broj 17. i broj 18. prikazana je modifikacija napravljena nad razvojnim sustavom u svrhu omogućavanja USB serijskog prijenosa.

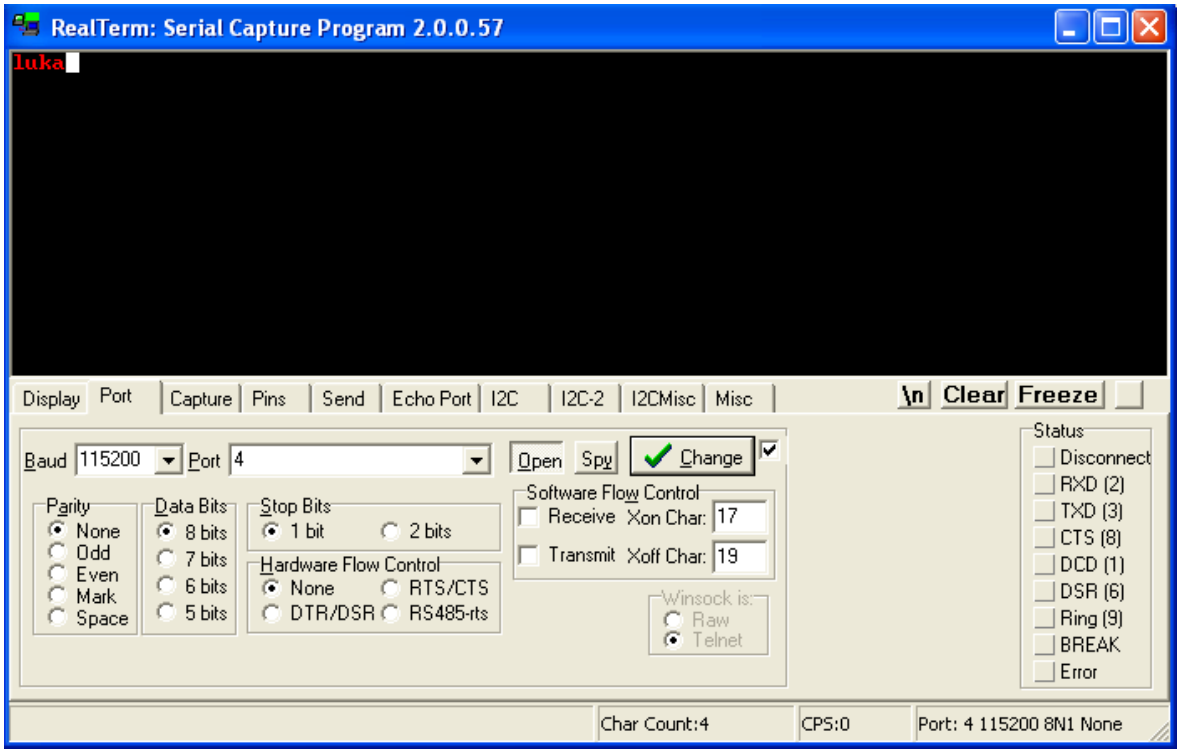


Slika 19. Prikaz modifikacije s gornje strane pločice



Slika 20. Prikaz modifikacije s donje strane pločice

Za provjeru ispravnosti modifikacije pomoću „Realterm“ terminala posalno je par znakova na seriju. Realterm je konfiguriran za slanje na port4 s baudrateom 115200. Ostale opcije su ostavljene su netaknutima.



Slika 21. Slanje znakova terminalom

Slika 22. Program uspješno prima znakove

7. Zaključak

Završni rad bavi se proučavanjem sustava za obradu mjernih podataka u stvarnom vremenu. Rad započinje upoznavanjem s razvojnim okruženjem i pripadnom programskom podrškom. Jedna od glavnih prepreka pri izradi rada bila je zastarjela programska podrška koja je za ispravan rad zahtijevala sada već nepodržani operacijski sustav Windows XP. Problem je riješen tako da je na „VMware workstation 12“ virtualnom stroju osposobljen „Windows XP professional“ pogodan za instaliranje programske podrške. Proučavanjem tehničke dokumentacije uspješno je konfiguriran kodek za rad s frekvencijom otipkavanja od 96 kHz. Sustav uspješno čita ulazne podatke, obrađuje ih, i prosljeđuje na izlaz. Nakon uspješne hardwareske modifikacije sustava osposobljen je USB serijski prijenos podataka u smjeru prema uređaju. U daljnjem radu potrebno je osposobiti uređaj za USB serijski prijenos podataka u oba smjera, te osposobiti mogućnost upravljanja sustavom pomoću jezgre operacijskog sustava.

8. Literatura

[1] Texas Instruments. http://www.ti.com.

[2] TMS320C55x Optimizing C/C++ Compiler v 4.4; User's Guide, Literature Number: SPRU281G, Texas Instruments, prosinac 2011.

http://www.ti.com.cn/cn/lit/ug/spru281g/spru281g.pdf

[3] TMS320C5505 Fixed-Point Digital Signal Processor, Literature Number: SPRS660F, Texas Instruments, rujan,2013.,

http://www.ti.com/lit/ds/symlink/tms320c5505.pdf

[4] TLV320AIC3204 Application Reference Guide, Reference Guide, Literature Number: SLAA557, Texas Instruments studeni, 2012

http://www.ti.com/lit/ml/slaa557/slaa557.pdf

[5] TLV320AIC3204 Ultra Low Power Stereo Audio Codec, Literature Number: SLOS602C, Texas Instruments, rujan 2008., revidirano studeni 2014.

http://www.ti.com/lit/ds/slos602c/slos602c.pdf

[6] B. Jeren. Digitalna obradba signala, slajdovi s predavanja. FER – ZESOI, Sveučilište u Zagrebu, 2007.

http://dos.zesoi.fer.hr/predavanja.html

[7] Petrinović,D., Ekvivalencija vremenski kontinuiranih i diskretnih signala i sustava,pdf., skripta, FER-ZESOI, Sveučilište u Zagrebu, 2008.

https://www.fer.unizg.hr/predmet/dos_b/predavanja

[8] TMS320C55x DSP/BIOS 5.x Application Programming Interface (API), Reference Guide, Literature Number: SPRU404Q, Texas Instruments, kolovoz 2012.

http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=spru404&fileType=pdf

9. Sažetak

U ovom radu opisan je postupak konfiguracije i programiranja sustava za obrađivanje podataka u stvarnom vremenu. Sadrži općeniti pregled sustava za obradu podataka u stvarnom vremenu, zatim slijedi opis korištenih tehnologija i pripadne programske podrške te završava s postupcima konfiguracije kodeka, implementacije algoritma za blokovsku obradu mjernih uzoraka i osposobljavanjem prijenosa podataka serijskim USB portom.

Ključni pojmovi :

1. Procesor za digitalnu obradu signala
2. Analogno digitalni konverter
3. Digitalno analogni konverter
4. Audio kodek

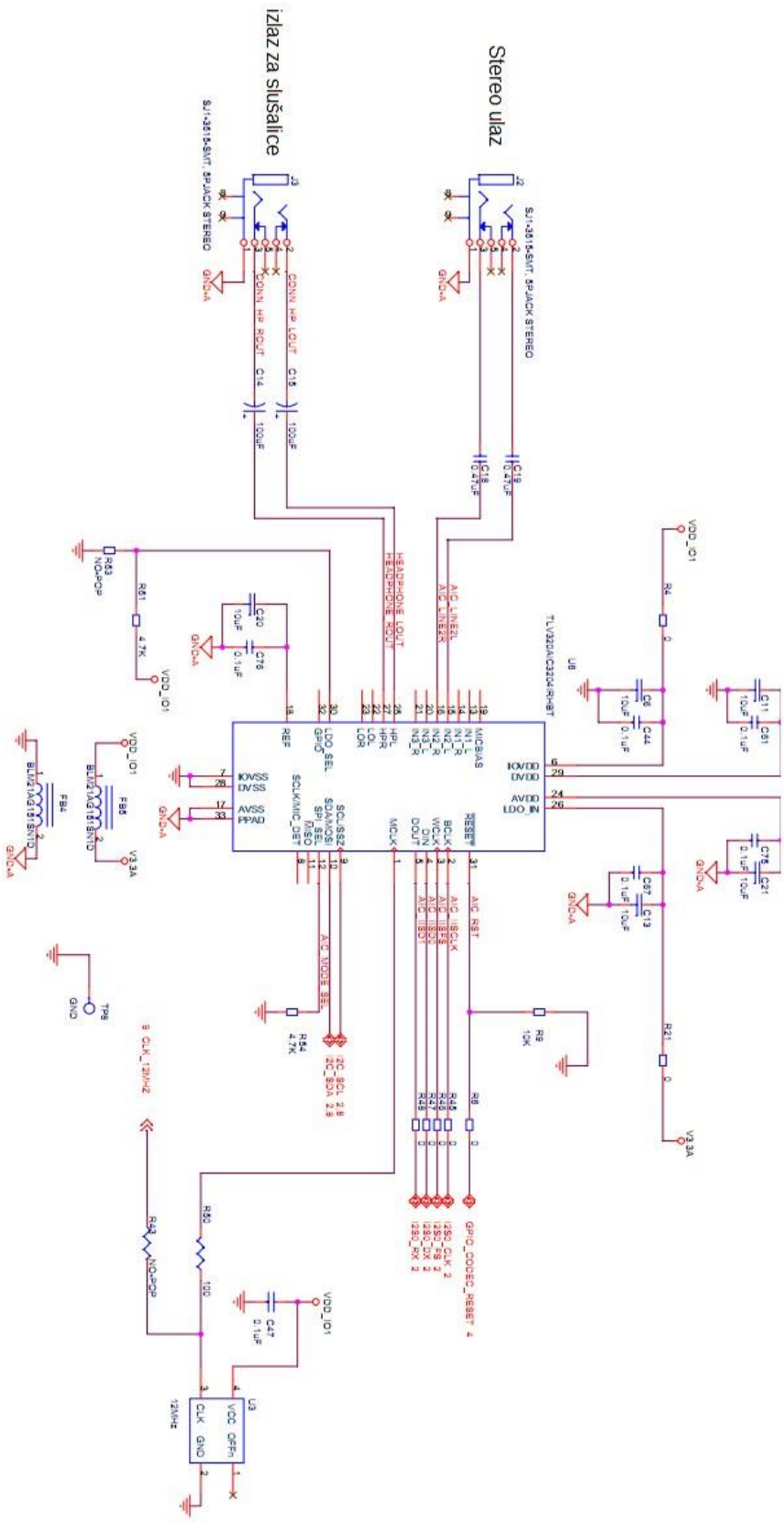
Summary

This paper gives an overview of the solution to the problem of real time control of embedded systems for block processing and measurments. It contains general description of real time embedded stystems, basic description of tehnologies used in project implementation. Finally, it describes procedures and configurations needed for proper system operation.

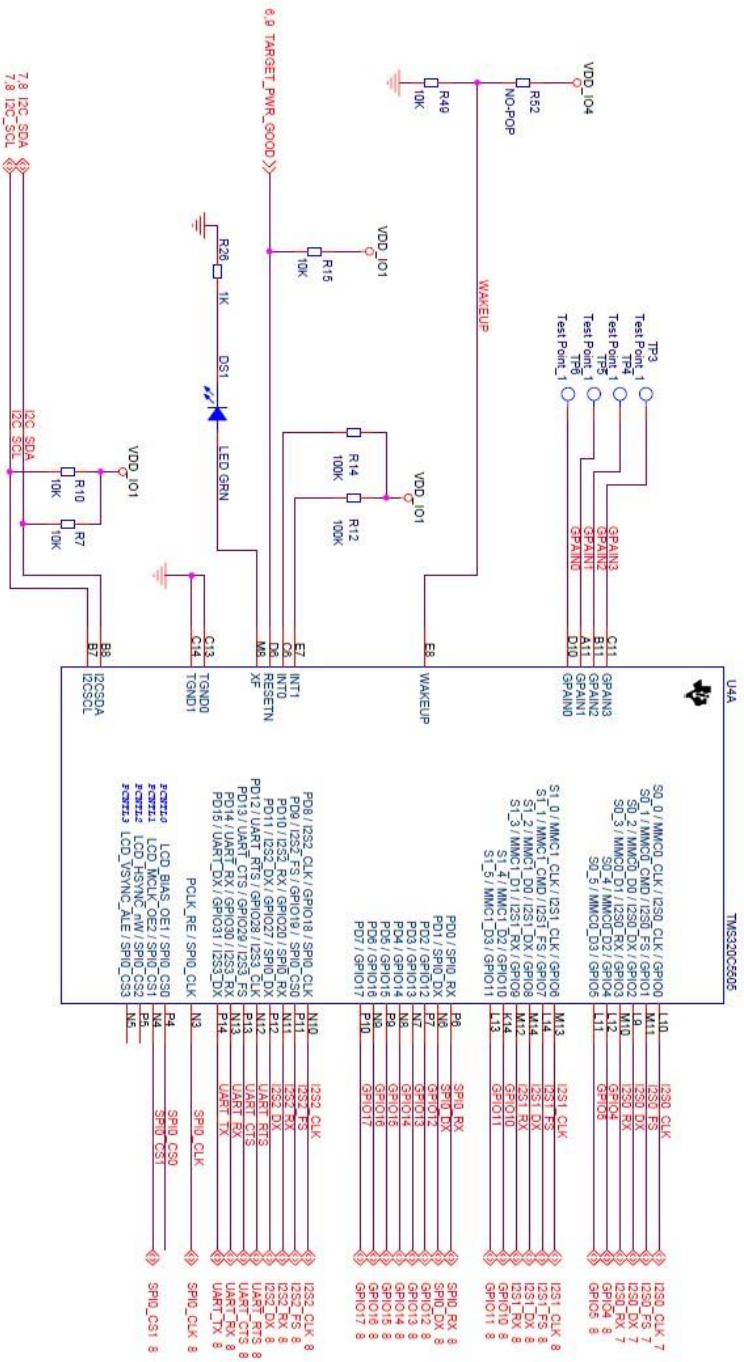
Key words:

1. Digital signal processor
2. Analog-to-digital converter
3. Digital-to-analog converter
4. Audio codec

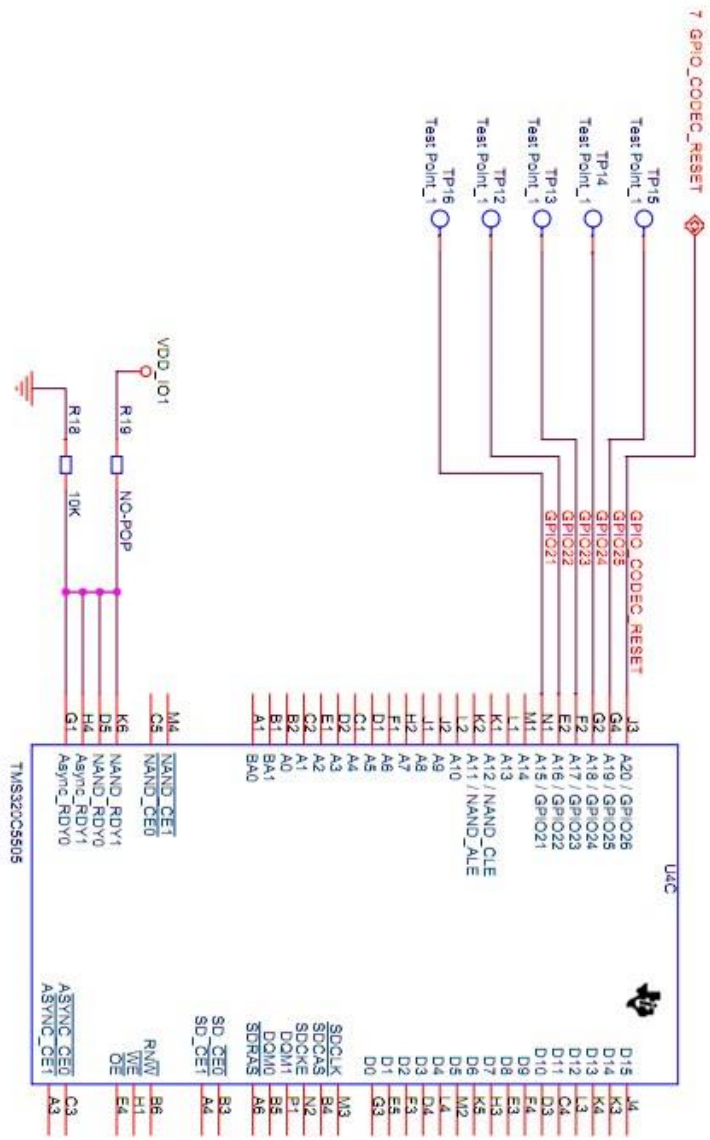
10. Privitak



Shema 1. Audio kodek TLP320AIC3204



Shema 2. procesor TMS320VC5505



Shema 3. procesor TMS320VC5505