Univerza v Ljubljani
Fakulteta za elektrotehniko

Černivec Gregor

# Spektralni analizator z DSP

Seminarska naloga

pri predmetu
Elektronska vezja

# 1 Uvod

## 1.1 Motivacija

Namen seminarske naloge je izvedba spektralnega analizatorja na razvojni platformi firme Analog Devices. Omogočal bi opazovanje audio spektra v realnem času in tudi enkraten zajem vzorcev, primernih za kasnejšo obdelavo. Glede HW izvedbe je potrebno zagotoviti dovolj hitro komunikacijo med DSP platformo in osebnim racunalnikom. Glede SW izvedbe je potrebno učinkovito integrirati FFT funkcije ter komunikacijo s codecom (A/D – D/A pretvornik na razvojni plošči) v algoritem na DSP platformi, ki v realnem času izvaja spektralno analizo vhodnega signala. Pravtako je potrebno napisati ustrezne rutine na HW nivoju, ki omogočajo komunikacijo med DSP platformo in osebnim računalnikom. Te se izvajajo na osebnem računalniku. In končno je bila v Visual Studiu (Microsoft Visual C++ 6.0) napisana tudi aplikacija, ki v sodelovanju z operacijskim sistemom učinkovito uporablja te rutine. Omogoča prikaz spektralnih črt v realnem času, njihov zajem, shranitev, printanje… Aplikacija na osebnem računalniku da celotnemu sistemu dodatno vrednost.

## 1.2 Funkcionalni opis sistema

DSP platforma spada v najnižji, t.j. low cost, razred razvojnih orodij firme Analog Devices. Vsebuje 16 bitni Digitalni Signalni Procesor ADSP-2181, z inštrukcijskim ciklom 30 ns in 16 kbesedami programskega ter 16 kbesedami podatkovnega pomnilnika, in 16 bitni sigma – delta A/D, D/A pretvornik - AD1847 Stereo SoundPort z maksimalno efektivno vzorčevalno frekvenco 48 kHz.

Ob resetu se iz ROM pomnilnika preko BDMA porta (BDMA – Boot Direct Memory Access) v DSP programski pomnilnik naloži monitor program (firmware). Ta omogoča osnovno komunikacijo med DSP-jem in PC-jem. Na PC-ju mora delovati host program (firmware), s katerim komuniciramo z monitor programom. Host program nam omogoča le zelo primitivno tehniko debuggiranja. Komunikacija poteka serijsko – asinhrono (RS232). Ker DSP nima HW izvedene asinhrone komunikacije, le to preko sinhronega porta programsko emulira monitor program. Ko napišemo program, ga prevedemo in s host programom pošljemo monitorju. Monitor zapiše program v programski pomnilnik in ga nato pokliče kot podprogram z ukazom CALL. V tem primeru je komunikacija med računalnikom in DSP-jem prekinjena, saj programski tok prevzame naš program. Če hočemo npr. ugotoviti vrednosti spremenjenih registrov ali podatkovnega pomnilnika, je potrebno nadzor ponovno dati monitorju. To storimo z ukazom RTS (return from subroutine).
Svoj program imamo lahko tudi v več tekstovnih datotekah, ki se imenujejo moduli. Vsak modul moramo posebej prevesti z assemblerjem. Ta ustvari objektne datoteke, ki jih je nato potrebno z linker programom prevesti v izvršilno .EXE datoteko. Ta je vedno ena sama. To datoteko nato pošljemo monitor programu, ki jo zapiše v programski pomnilnik.
Ko smo zadovoljni s programom, ga zapečemo v EPROM. Z njim zamenjamo boot ROM. Ob resetu se tako v namesto monitorja v programski pomnilnik naloži naš program.

Vrata v analogni svet predstavlja stereo audio codec AD1847. Na splošno je to kar pogosto uporabljano vezje starejših zvočnih kartic na matičnih ploščah. Ta ima 4 multipleksirane analogne vhode in en analogni izhod. Za komunikacijo z DSP-jem codec uporablja sinhrono časovno multipleksirano 2-wire komunikacijo. Vlogo master ima vedno codec, kar pomeni, da codec generira uro in frame sinhronizacijo. Vsak frame je sestavljen iz 32 time slotov, vsak time slot pa iz 16 bitov. Časovno multipleksiranje je še posebno ugodno z vidika DSP-ja, ki s posebno HW metodo – autobuffering omogoča avtomatsko zapisovanje time slotov – vzorcev v določen konec pomnilnika (buffer). Če je ta buffer krožni, bo po zadnjem rezerviranem naslovu spet skočil na prvega. Vse to pa se dogaja popolnoma avtomatsko in

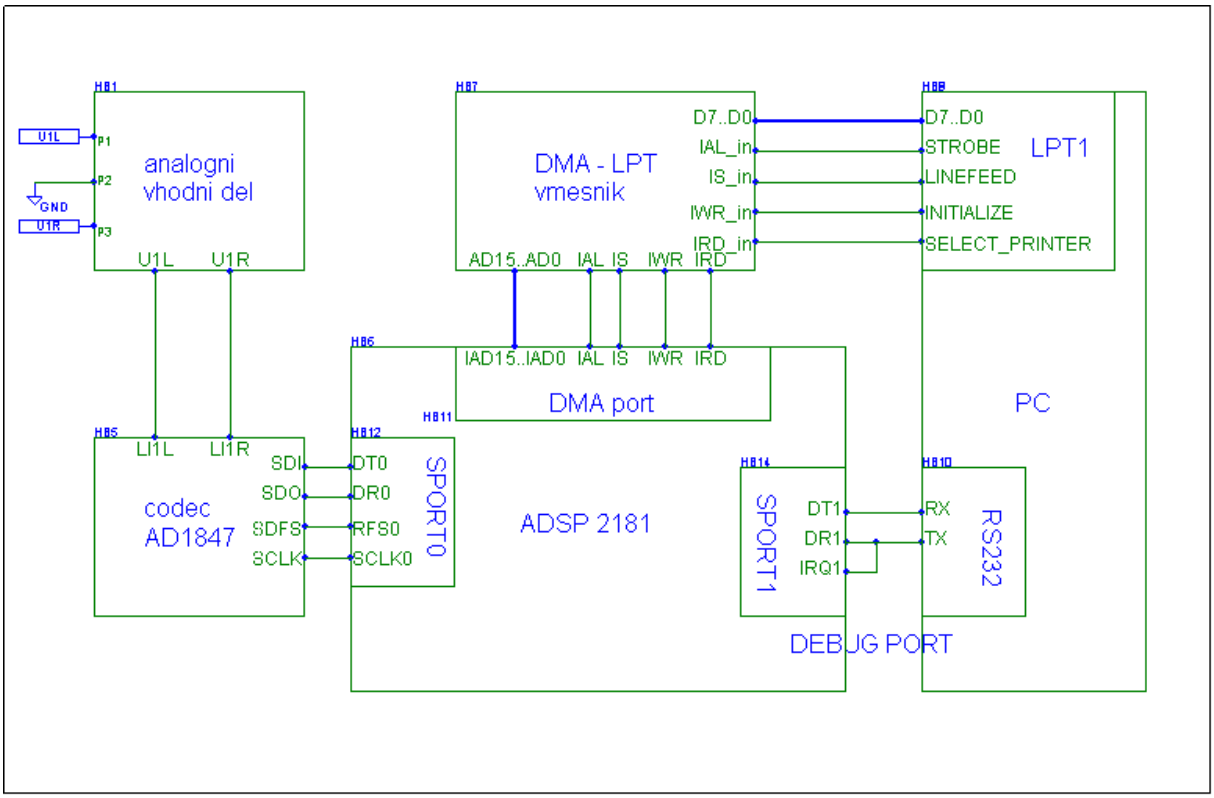
brez programskega posredovanja. Potrebna je samo začetna inicializacija autobufferja in sinhronega porta SPORT0, nato pa vzorci prihajajo v pomnilnik automatsko. To pomeni, da imamo ob vsakem času v pomnilniku za dolžino bufferja tekočih vzorcev vhodnega signala. Vedeti moramo samo, na kateri lokaciji se ob klicu fft rutin nahaja zadnji tekoči vzorec. To je pomembno, ker moramo pred klicem glavne fft rutine vhodne vzorce ustrezno preurediti, da dobimo na izhodnem bufferju frekvenčne vzorce v pravilnem vrstnem redu.

Komunikacija z osebnim računalnikom je paralelna – asinhrona in poteka prek DMA porta DSP procesorja. DMA ali Direct Memory Access pomeni dostop (branje in pisanje) do podatkovnega ali programskega pomnilnika DSP processorja, pri čemer procesor obremenimo z dodatnim ciklom za vsako prebrano ali napisano besedo. Samo izvajanje glavnega programa je neprekinjeno. Kontrola DMA porta je izključno zunanja in poteka preko 4 kontrolnih linij iz LPT porta osebnega računalnika. Podatki se prenašajo preko osmih podatkovnih linij LPT porta osebnega računalnika. Izhodne frekvenčne komponente fft rutina zapisuje v določen konec podatkovnega pomnilnika, neodvisno od tega pa do njih preko LPT – DMA dostopa tudi aplikacija na osebnem računalniku.

Aplikacija na osebnem računalniku kliče funkcijo, ki dostopa do registrov paralelnega porta LPT1. Ta funkcija predstavlja nek gonilnik -– driver za LPT1 port. Kot parameter sprejme začetni in končni naslov DMA bufferja DSP procesorja, vrne pa frekvenčne vzorce, ki so sedaj zapisani v novem bufferju, v RAM-u osebnega računalnika. Ta buffer nato uporabi aplikacija za izris spektralnih črt. Funkcija se kliče periodično vsakih 20 ms, kar zagotavlja dovolj veliko real – time dinamiko.

# 2 Glavni del

## 2.1 Opis delovanja podsklopov vezja s shematskim prikazom podsklopov



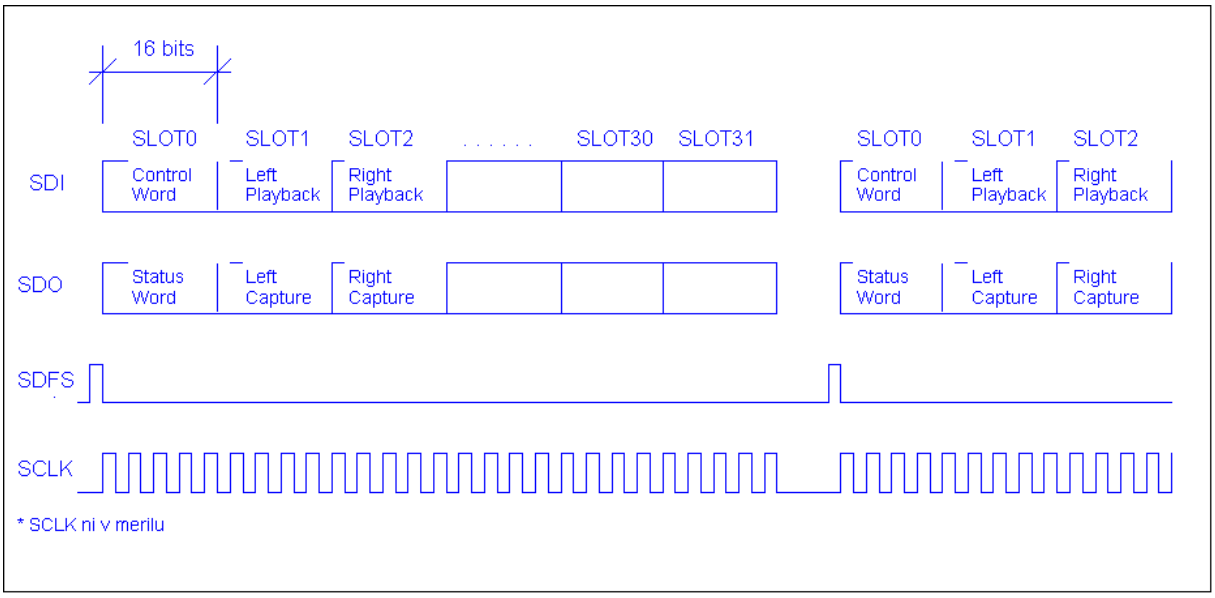

### 2.1.1 Analogni vhodni del

Sestavljata ga levi in desni kanal. Na vsakem kanalu je aktivni pasovno prepustni filter 1. reda. Spodnjo mejno frekvenco ima pri 200 Hz, zgornjo pa pri 19 kHz. Shema analognega dela je prikazana v dodatku.

### 2.1.2 Codec AD1847

Za vhode v codec je priporočena kapacitivna ločitev in nizkopasovni filter 1. reda, kar je rešeno z analognim vhodnim delom. Problem prekrivanja spektrov (alliasing efekt) je rešen z nadvzorčenjem, pri čemer je vhodni signal vzorčen s frekvenco 2MHz, nato pa je spekter decimiran. Tako so možne različne efektivne vzorčevalne frekvence, ki so programabilne od 5.5 kHz do 48 kHz.
Ob začetku delovanja je potrebno audio codec ustrezno inicializirati. Programerski model temelji na sinhroni serijski 2-wire komunikaciji. V frame-u je 32 time-slotov. Nov vzorec A/D pretvorbe se pojavi na vsak frame. V frame-u pa so izkoriščeni samo prvi trije 16-bitni time-sloti. Ostali time-sloti so neaktivni, kar pomeni, da jih ne beremo. V teh prvih treh time-slotih so locirani direktni registri, ki jih uporabljamo za nadzor codeca, dostop do indirektnih registrov in branje vzorcev. Izhod serijske komunikacije SDO (Serial Data Output) je sinhroniziran s frame signalom SDFS (Serial Data Frame Synchronization). Prvi time-slot vsebuje statusno besedo (Status Word), ki nam pove ali je codec v procesu inicializacije ali v procesu normalnega delovanja ali pa je tik pred prekinitvijo serijske komunikacije. Drugi time-slot vsebuje 16-bitni zajem levega kanala (Left Capture). Tretji time-slot vsebuje 16 bitni

zajem desnega kanala (Right Capture). Ostalih 29 time-slotov je neaktivnih. Tudi vhod SDI (Serial Data In) je sinhroniziran s frame signalom. Prvi time-slot vsebuje kontrolno besedo (Control Word). Kontrolna beseda vsebuje naslovno in podatkovno polje, s katerima inicializiramo indirektne registre codeca. Drugi time-slot vsebuje 16 bitni podatek za levi kanal D/A pretvornika (Left Playback). Tretji time-slot vsebuje 16 bitni podatek za desni kanal D/A pretvornika (Right Playback). Vendar pa ne uporabljamo teh dveh time-slotov, saj nas v primeru spektralne analize zanima le A/D pretvorba. Princip delovanja serijske komunikacije prikazuje spodnja slika. Shema codeca je prikazana v dodatku.



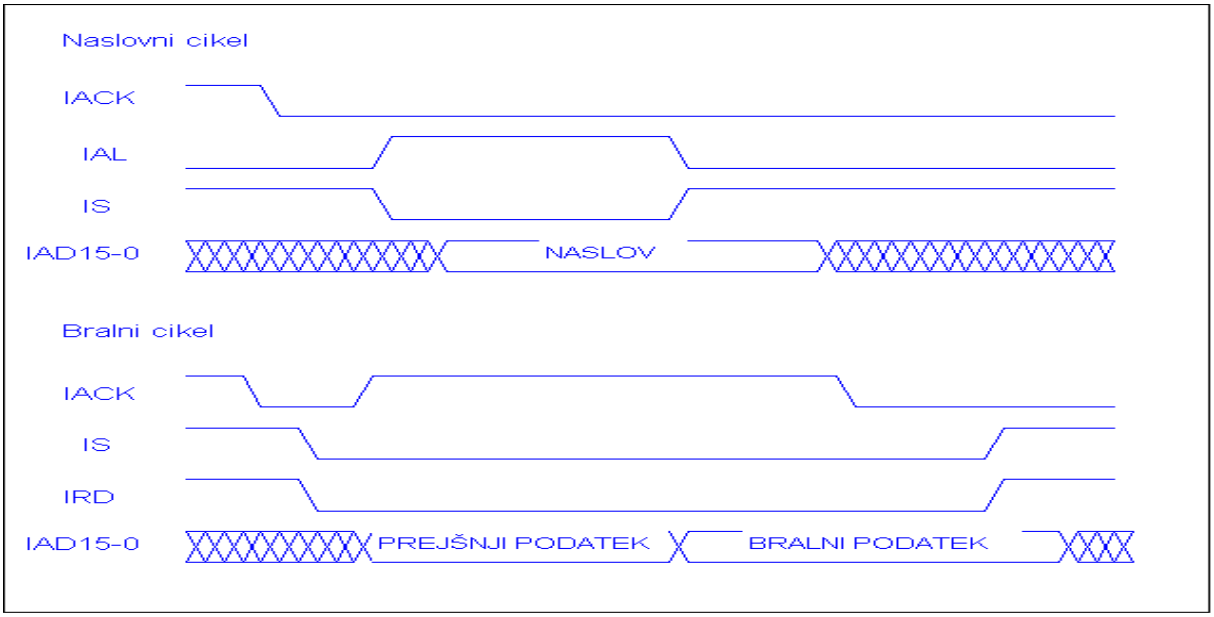

### 2.1.3 ADSP 2181

Za komunikacijo s codecom DSP procesor uporablja sinhroni serijski port SPORT0. SPORT0 uporablja podobne kontrolne in podatkovne signale kot codec: DT – serial Data Transmit, DR – serial Data Receive, RFS0 – Receive Frame Sinchronization, SCLK0 – Serial Clock. DSP processor deluje kot Slave. To pomeni da signala SCLK0 in RFS0 generira Master – codec. Oba dela SPORT0 oddajni in sprejemni uporabljata t.i. double buffering. Ko je 16 bitni sprejemni shift register napoljnjen, se vrednost prenese v programsko dostopen podatkovni serijski register RX0 in ob tem se generira tudi serijska prekinitev SPORT0_RX. Pri oddajanju pa se po prvem prenesenem bitu oddajnega shift registra generira serijska prekinitev SPORT0_TX. Sedaj lahko v programski oddajni register TX0 zapišemo novo vrednost za oddajo, ki se bo avtomatsko prenesla v shift register, ko bo le-ta prazen. V multichannel načinu vsaka sprejeta oz. oddana beseda pomeni svoj kanal. Uporabljamo časovni multipleks. Sedaj je potrebna tudi sprejemna frame sinhronizacija RFS0, ki nam označi začetek prenosa bloka podatkov – frame. Frame je razdeljen na 32 manjših enot – time-slotov, pri čemer en time-slot zapolni cel sprejemni ali oddajni register – 16 bitov. Kanale, ki nas ne zanimajo, onemogočimo. Omogočimo torej samo kanale – time-slote 0,1 in 2.

Najboljši način za prenos podatkov je, nadgradnja sinhrone komunikacije z uporabo autobufferinga. Namesto, da generiramo prekinitev za vsak sprejeti time-slot, je veliko ugodneje, da prekinitev generiramo šele po sprejetih time slotih, ki so za nas zanimivi; 0,1 in 2. Ti time-sloti se avtomatsko zapišejo v dodeljeni del podatkovnega pomnilka, ki je v tem primeru 3 lokacije dolg buffer. Seveda to omogoča arhitektura DSP-procesorja, saj je programsko posredovanje mogoče šele ob prekinitvi, t.j. ko so sprejeti vsi trije time-sloti.

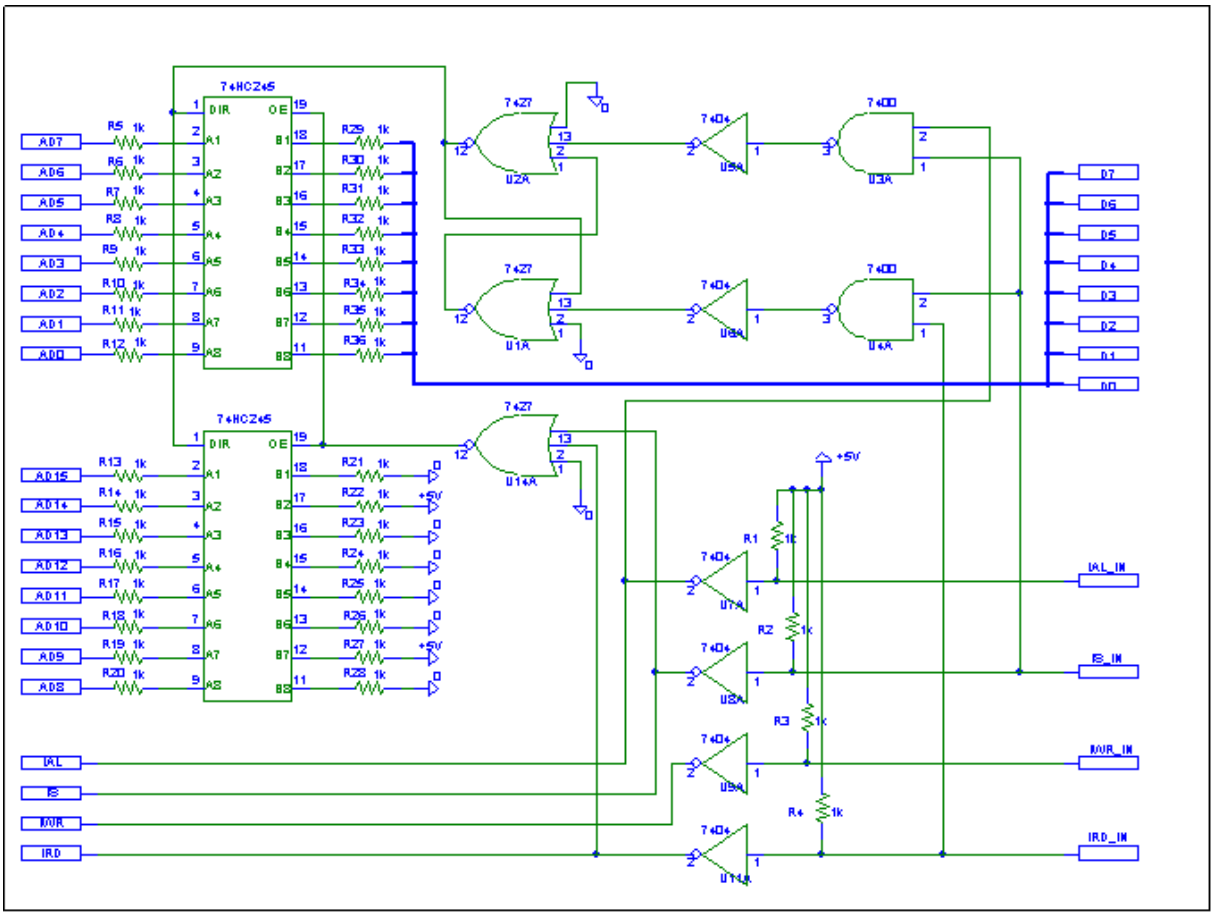
Ostali time-sloti se ignorirajo, saj smo te kanale onemogočili. Postopek se ponovi ob ponovni frame sinhronizaciji.

DMA dostop do pomnilnika se izkaže kot najugodnejši, saj ne potrebuje nobenega posredovanja programa. IDMA je paralelni port, ki nam omogoči DMA dostop. Vsi kontrolni signali so vhodni, t.j. IDMA port krmili neka druga naprava, v našem primeru LPT port, osebnega računalnika. Izhodni je samo signal IACK (IDMA Acknowledge), ki signalizira zasedenost DMA. Dostop do notranjega pomnilnika se vedno izvaja v dveh korakih. V prvem koraku na vodilo zapišemo naslov. Ob negativni fronti signala IAL (IDMA port Address Latch enable) se naslov zapiše v notranji kontrolni register. Ta je programsko dostopen, kar pomeni, da začetni naslov lahko poda tudi DSP procesor. Drugi korak je branje ali pisanje podatka na isto vodilo. To se zgodi ob aktivnem bralnem – IRD ali aktivnem pisalnem IWR signalu. Če beremo npr. blok podatkov – buffer, ni potrebno ponovno izvajati prvega koraka, ker se naslov avtomatsko poveča ob vsakem DMA prenosu. Potrebno je samo ustrezno postavljati IWR ali IRD signal. Prvi in drugi korak za branje prikazuje spodnja slika. Shema DSP procesorja je prikazana v dodatku.




### 2.1.4 DMA – LPT vmesnik

Ker ima LPT port osebnega računalnika samo 8 podatkovnih in 4 kontrolne linije ni mogoča direktna priključitev LPT porta na DMA port. Seveda tudi ni priporočljiva. Ker 8 bitov pomeni samo 256 naslovnih lokacij, je potrebno višjih 8 bitov fiksno določiti. Potrebno je popolno dekodiranje v naslovnem prostoru. Za priključitev na DMA vodilo sta uporabljena dva 8-bitna dvosmerna bufferja, pri čemer so nivoji višjega naslovnega bufferja stalno določeni. Obračanje smeri bufferjev je izvedeno s preprostim 1-bitnim avtomatom – latch. S tem prihranimo 1 kontrolno linijo, ki jo sedaj lahko uporabimo tudi za pisalne cikle. S tem smo dobili popoln dostop do notranjega pomnilnika, tako da sedaj lahko podatke tudi vpisujemo. Vsi kontrolni signali LPT porta so speljani preko negatorjev, tako da dobimo ostre fronte, primerne za kontrolo DMA porta. Kontrolni signali so tipa open colector. Signal IACK – ackonowledge se ne detektira, čeprav bi se lahko, saj imamo na voljo še 4 vhodne linije. V praksi se izkaže, da detekcija IACK tudi ni potrebna, čeprav pripomore k robustnosti sistema. Shemo prikazuje spodnja slika.



### 2.1.5 LPT1 port

Fizično LPT port osebnega računalnika predstavlja konektor DB25. Razpored pinov prikazuje spodnja slika.

| Pin No (D-Type 25) | Pin No (Centronics) | SPP Signal | Direction In/out | Register | Hardware Inverted |
|---|---|---|---|---|---|
| 1 | 1 | nStrobe | In/Out | Control | Yes |
| 2 | 2 | Data 0 | Out | Data | |
| 3 | 3 | Data 1 | Out | Data | |
| 4 | 4 | Data 2 | Out | Data | |
| 5 | 5 | Data 3 | Out | Data | |
| 6 | 6 | Data 4 | Out | Data | |
| 7 | 7 | Data 5 | Out | Data | |
| 8 | 8 | Data 6 | Out | Data | |
| 9 | 9 | Data 7 | Out | Data | |
| 10 | 10 | nAck | In | Status | |
| 11 | 11 | Busy | In | Status | Yes |
| 12 | 12 | Paper-Out PaperEnd | In | Status | |
| 13 | 13 | Select | In | Status | |
| 14 | 14 | nAuto-Linefeed | In/Out | Control | Yes |
| 15 | 32 | nError / nFault | In | Status | |
| 16 | 31 | nInitialize | In/Out | Control | |
| 17 | 36 | nSelect-Printer nSelect-In | In/Out | Control | Yes |
| 18 - 25 | 19-30 | Ground | Gnd | | |

Podatkovni pini pin2-pin9 so primarno definirani kot izhodni, vendar pa novejše matične plošče polek ostalih standardnih načinov delovanja podpirajo tudi SPP – bidirectional. Kar pomeni obojesmerni Standard Serial Port. Seveda je potrebno ta način nastaviti v BIOS-u. Kontrolni pini so uporabljeni le kot izhodi in so tipa open collector. Če jih uporabljamo kot vhode, jih moramo predtem postaviti v log.1 – t.j. setirati, da jih nato od zunaj potegnemo dol, resetiramo. Ostali pini (pin10, 11, 12 ,13, 15) so statusni in kot taki vhodni. Teh ne uporabljamo. Izhodni in vhodni nivoji so TTL kompatibilni. Večja neznanka pa je skupni tok, ki ga paralelni port še lahko sprejme. Ta znaša okoli 12 mA. Programsko se LPT port izkazuje s tremi registri, ki odgovarjajo fižičnim signalom. Registre prikazuje spodnja slika.

| Offset | Name | Read/Write | Bit No. | Properties |
|---|---|---|---|---|
| Base + 0 | Data Port | Write (Note-1) | Bit 7 | Data 7 (Pin 9) |
| | | | Bit 6 | Data 6 (Pin 8) |
| | | | Bit 5 | Data 5 (Pin 7) |
| | | | Bit 4 | Data 4 (Pin 6) |
| | | | Bit 3 | Data 3 (Pin 5) |
| | | | Bit 2 | Data 2 (Pin 4) |
| | | | Bit 1 | Data 1 (Pin 3) |
| | | | Bit 0 | Data 0 (Pin 2) |
| Base + 1 | Status Port | Read Only | Bit 7 | Busy |
| | | | Bit 6 | Ack |
| | | | Bit 5 | Paper Out |
| | | | Bit 4 | Select In |
| | | | Bit 3 | Error |
| | | | Bit 2 | IRQ (Not) |
| | | | Bit 1 | Reserved |
| | | | Bit 0 | Reserved |
| Base + 2 | Control Port | Read/Write | Bit 7 | Unused |
| | | | Bit 6 | Unused |
| | | | Bit 5 | Enable bi-directional Port |
| | | | Bit 4 | Enable IRQ Via Ack Line |
| | | | Bit 3 | Select Printer |
| | | | Bit 2 | Initialize Printer (Reset) |
| | | | Bit 1 | Auto Linefeed |
| | | | Bit 0 | Strobe |

Inicializacijo registrov, pomembnih bitov in funkciji za setiranje in resetiranje bitov prikazuje spodnja koda. Ta del kode je tudi osnovni gradnik samega gonilnika za LPT.

```
//definicije portov
#define PORTADDRESS 0x378
#define DATA_PORT PORTADDRESS+0
#define STATUS_PORT PORTADDRESS+1
#define CONTROL_PORT PORTADDRESS+2

//definicije bitov
#define IAL 0                   //bit 0 v CONTROL_PORT - STROBE (LPT=pin1)
#define IS  1                   //bit 1 v CONTROL_PORT - LINEFED(LPT=pin14)
#define IWR 2                   //bit 2 ni negiran (LPT=pin16)
#define IRD 3                   //bit 3 v CONTROL_PORT - SELECT PRINTER (LPT=pin17)

int   reg_old, reg_new, bitna_maska, i;

//*****************************************************************************
//Funkciji set in reset bit
//*****************************************************************************
void  set_bit(int reg, int bit)
{
        reg_old=_inp(reg);
        for (i=0,bitna_maska=1;i<bit;i++){
```

```
                if(bit!=0){
                        bitna_maska=bitna_maska*2;
                }
        }
        reg_new=bitna_maska | reg_old;
        _outp(reg,reg_new);
}

void  reset_bit(int reg, int bit)
{
        reg_old=_inp(reg);
        for (i=0,bitna_maska=1;i<bit;i++){
                if(bit!=0){
                        bitna_maska=bitna_maska*2;
                }
        }
        bitna_maska=0xFF-bitna_maska;
        reg_new=bitna_maska & reg_old;
        _outp(reg,reg_new);
}
```

Funkciji set_bit() in reset_bit() sprejmeta kot argumenta naslov registra in bit, ki ga hočemo postaviti ali zbrisati. Z ustreznim zaporedjem funkcij set_bit() in reset_bit(), generiramo ustrezen časovni diagram za krmiljenje DMA porta. Vrednost podatka na vodilu preberemo s funkcijo _inp(data_port), ki nam vrne vrednost podatkovnega registra.

## 2.2 Analiza delovanja

Delovanje lahko razdelimo v groben na dva, med seboj neodvisna dela. Programski tok na DSP procesorju in programski tok na osebnem računalniku. V programskem toku DSP procesorja se v neskončni zanki izvajata branje vzorcev iz codeca in spektralna analiza le teh. Branje vzorcev je izvedeno s pomočjo prekinitve serijskega sinhronega vmesnika SPORT0 DSP procesorja. Ko je 3 lokacije globok avtobuffer napoljen, se izvede prekinitvena rutina, ki prebere vzorce in jih spravi v nov buffer. Prekinitveno rutino prikazuje spodnja koda. Najprej izvedemo prenos podatkov iz sprejemnega bufferja – rx_buff. Preberemo levi in desni kanal. Nato pa v vhodni buffer fft rutine prenesemo samo vzorec desnega kanala.

```
input_samples:
        ena sec_reg;                    {shadow register bank }
        mr0 = dm (rx_buf + 1);          {levi kanal}
        mr1 = dm (rx_buf + 2);          {desni kanal}
        dm (i4, m4) = mr1;              {samo desni kanal}
rti;
```

Vsi podatkovni prenosi v DSP procesorju temeljijo 'generatorju podatkovnih naslovov' – DAG (Data Address Generator). Sam DSP procesor ima dva taka podatkovna adresna generatorja DAG1 in DAG2. Vsak izmed teh generatorjev pa lahko neodvisno tvori naslove za 4 bufferje. Indeksni register i4 vsebuje začetni naslov vhodnega fft bufferja, modifikacijski register pa ima vrednost 1, kar pomeni, da se po vsakem podatkovnem prenosu indeksni register poveča za eno lokacijo. Ob vsaki prekinitvi se v vhodni fft buffer tako vpiše nov vzorec. Vhodni buffer fft rutine pa mora biti dvojni, saj se vrednosti vzorcev med njihovo spektralno analizo ne smejo spreminjati. Čas prenosa 1024 vzorcev , kolikor je velik vhodni fft buffer znaša približno 21.3 ms. Ostali del procesorskega časa vzame spektralna analiza 1024 vzorcev, ki nam za rezultat da novih 1024 vzorcev. Informacijo o spektru nosi prvih 512 vzorcev, ostali vzorci so zrcalna slika prvih. Sama spektralna anliza poteka po enačbi in uporablja decimacijo v

$$X[k] = \sum_n x[n] \cdot W_N^{k \cdot n}$$

času.
Pri uporabi fft rutin je potrebno biti pozoren, kako podajamo podatke fft rutini. Ker uporabljamo decimacijo v času DIT, je potrebno podati vzorce (njihove naslove) v reverznem bitnem redu (naslovni bit 9 postane bit 0, bit 8 gre v bit 1 itd…). Samo na ta način dobimo izhodne frekvenčne vzorce v pravilnem vrstnem redu. Preden gredo časovni vzorci v obdelavo, jih oknimo s Hammingovim oknom. S tem dosežemo večje slabljenje stranskih snopov in čistejšo spektralno sliko. Sama fft rutina je tipa Radix2, kar pomeni, da se vzorci razdelijo na dva dela, nad katerima se vrši DFT (Diskretna Fourierjeva transforamcija). Čas izračuna za 1024 vzorcev znaša približno 2.22 ms. V izhodnem bufferju fft rutine dobimo 1024 frekvenčnih vzorcev, oziroma spektralno sliko vhodnega signala. Frekvenčna skala je linearna z razdelkom 48kHz/1024 vzorcev. Rezultat Fourierjeve transformacije je zapisan v dveh bufferjih – realnem in imaginarnem s po 1024 vzorci. Za prikaz močnostnega spektra je potrebno sešteti kvadrata obeh nizov. Če bi hoteli amplitudni prikaz, moramo rezultat še koreniti, vendar bi ta operacija zahtevala dosti procesorskega časa (Metoda fiksne točke, Taylorjeva aproksimacija…), zato se zadovoljimo s prikazom močnostnega spektra.

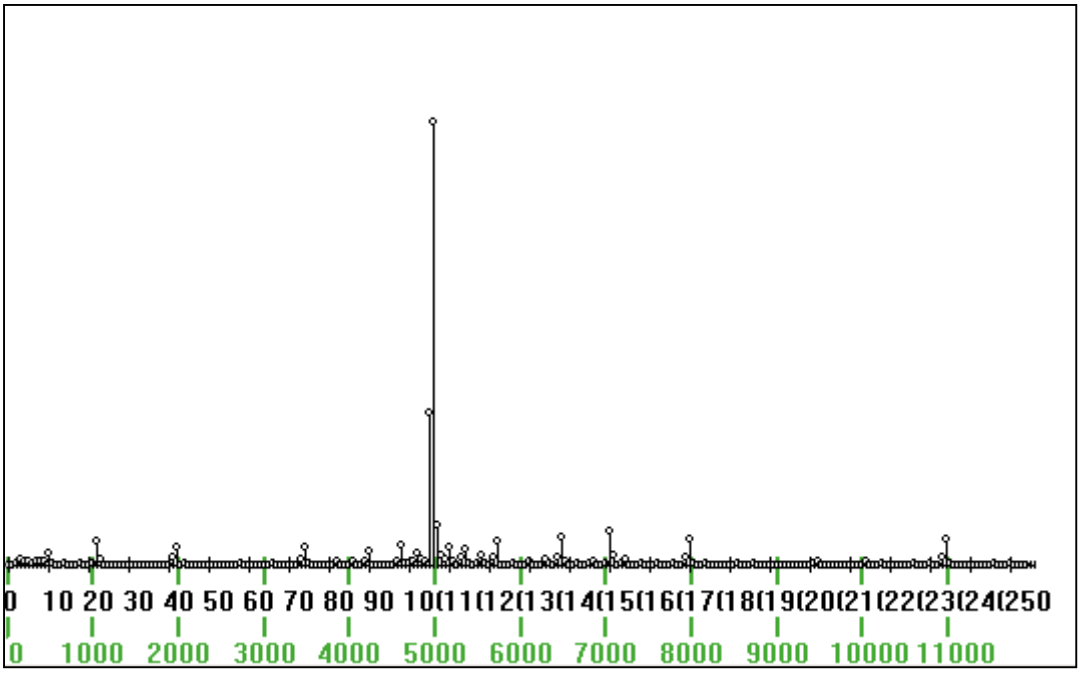
Bistvo programskega toka na osebnem računalniku je timer operacijsega sistema, ki v časovnih intervalih 24 ms kliče gonilnik LPT porta. Ob klicu gonilnik izvede časovni diagram na kontrolnih signalih za vpis naslova v DMA port in branje 256 vzorcev. S tem dobimo informacijo o spektralni sliki od 0 do 11.9 kHz. Koda gonilnika je prikazana spodaj. Uporabniški vmesnik pa je delo Aleša Černivca in je opisan v dodatku.

```
diagram_burst_DSP_branje()
{
        int naslov=0;
        set_bit(CONTROL_PORT,IRD);                      //branje neaktivno
        set_bit(CONTROL_PORT,IS);                       //chip disable

        //for (naslov=m_nNaslov1;naslov<=m_nNaslov2;naslov++){
                reset_bit(CONTROL_PORT,5);                      //smer LPT data je izhodna
                reset_bit(CONTROL_PORT,IS);                     //chip enable, direction_IN condition
                set_bit(CONTROL_PORT,IAL);                      //latch condition, direction_IN
                _outp(DATA_PORT,naslov);
                reset_bit(CONTROL_PORT,IAL);                    //latching
                set_bit(CONTROL_PORT,IS);                       //chip disable
                for (naslov=m_nNaslov1;naslov<=m_nNaslov2;naslov++){
                        set_bit(CONTROL_PORT,5);                //smer LPT data je vhodna
                        set_bit(CONTROL_PORT,IWR);              //data_buffer_enable_OFF
                        reset_bit(CONTROL_PORT,IS);             //chip enable, direction_OUT
                        reset_bit(CONTROL_PORT,IRD);            //read, direction_OUT
                        reset_bit(CONTROL_PORT,IWR);            //data_buffer_enable_ON
                        nbuffer[naslov]=_inp(DATA_PORT);
                        set_bit(CONTROL_PORT,IRD);              //end read
                        set_bit(CONTROL_PORT,IS);               //chip disable

        }
}
```

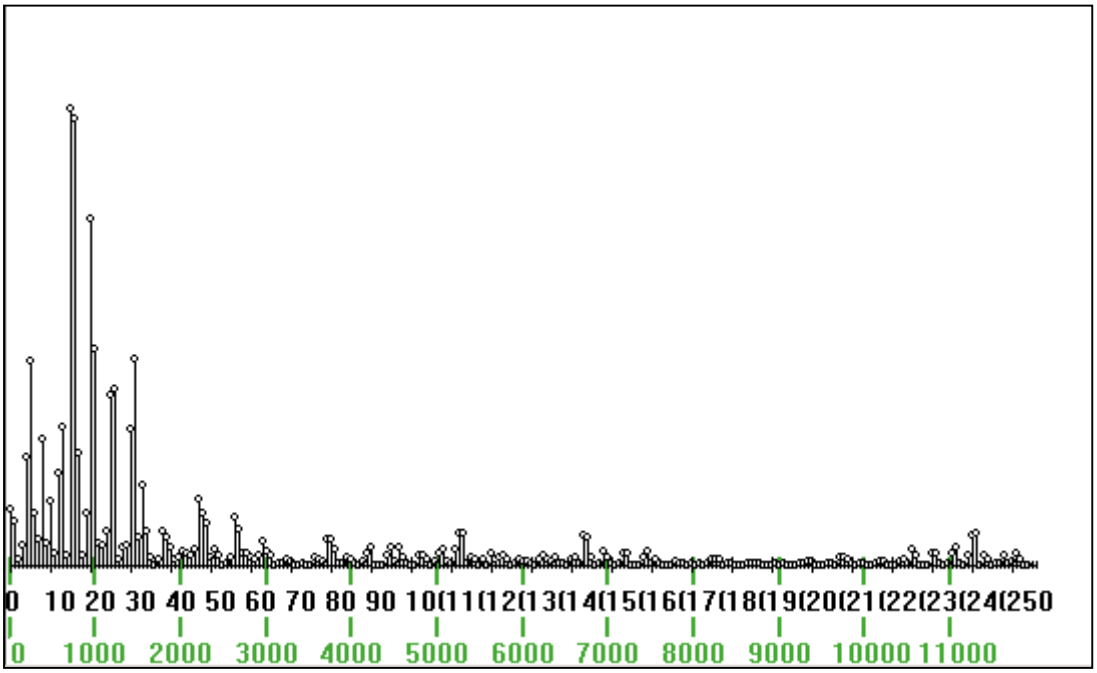
## 2.3 Izmerjene karakteristike

Celoten cikel zajema podatkov, fft pretvorbe in izrisa na osebni računalnik znaša približno 30 ms. Če torej spreminjamo frekvenco ali amplitudo vhodnega signala, bo odziv osebnega računalnika v najslabšem času znašal 30 ms. To pa je čas, ki ga človek skoraj ni sposoben zaznati. (Človeški refleks je približno 80ms). Tako, da se celoten sistem obnaša kot 'real time' oziroma brez opaznih zakasnitev. Informacija o frekvenci je relativno točna. Napaka znaša +- 24 Hz. Informacija o amplitudi je nekoliko slabša. Merilnik ima na vhodu analogni

NF filter, katerega bi bilo potrebno upoštevati v prikazu amplitude. Amplitudna (močnostna) skala tudi ni umerjena. Spodnja slika prikazuje sinusni signal 5 kHz, amplitude nekaj 100 mV.




Glas A :



# 3 Zaključek

## 3.1 Morebitne težave

Do težav pride pri nekaterih operacijskih sistemih (win 2000, win Xp, win NT), ki imajo zaradi varnosti dostop do paralelnega porta zaščten. Pognati je treba preprost gonilnik, ki nam omgoči dostop do paralelnega porta. Težave nastopajo tudi v sami komunikaciji med LPT in DMA portom. Pri komunikaciji se ne uporablja standardne paralelne komunikacije ECP ali EPP, ki hardversko generirata kontrolne signale ampak SPP, ki omogoča programski nadzor kontrolnih signalov. Ko na podatkovno data vodilo pridejo podatki, kar se zgodi pri postavljanju oz. brisanju izbranih kontrolnih signalov, se občasno zgodi glitch. Ta povzroči ponovno branje DMA porta, ki se zato avtomatsko poveča za 1. Glitche pa je tudi hardversko zelo težko odpraviti.

## 3.2 Sklepne ugotovitve

Prikazan spektralni analizator ja poskus izdelave amaterskega low cost instrumenta, ki pa ob dobri umeritvi lahko postane tudi koristno orodje za merjenje akustičnih signalov. Seveda bi se prava uporabnost izkazala ob implementaciji na enem tiskanem vezju. Trenutno je platforma izvedena na 4 slojni tiskanini. Implementacija na dvostranski tiskanini bi bila nekoliko težja, gotovo pa ne nemogoča. Verjetno je en sloj v 4 slojni tiskanini cel zalit z maso, kar je pri mikroprocesorskih sistemih še posebno ugodno. To bi se na dvostranski tiskanini rešilo za zalivanjem z maso na obeh straneh, z ločitvijo analogne in digitalne mase, ki bi se stikali le v eni točki in s pazljivim dizajniranjem analognega dela pri codecu ter enako dolžino zunanjih vodil digitalnega dela.

## 3.3 Možnosti nadgradnje

Nadgradnja obstoječega sistema bo šla v izdelavo nove tiskanine. Na njej bo verjetno enak digitalni procesor, codec pa bo nov. Predvsem tak z mnogo višjim vzorčenjem. Serijska komunikacija med codecom in DSP-jem na tem mestu ne bo predstavljala večjih težav, saj je sedaj zelo slabo izkoriščena – 3 od 32 time slotov. Veliko dela bo vloženega predvsem v izboljšavo komunikacije med DMA in osebnim računalnikom. Tu se sedaj izkazuje več možnosti. Prva je paralelni prenos po hardverskem EPP standardu. To dopušča maksimalno hitrost do 150 kB/s. V najboljšem primeru bi tako znašalo vzorčenje 150 kHz pri 8 bitih. Kar ni posebej zadovoljivo. Druga možnost je USB1 standard, ki omogoča tipične pretoke do 1.5 Mbita. To bi omogočalo vzorčenje nekaj nad 100 kHz pri 16 bitni resoluciji. Ta varianta je nekako mejno zadovoljiva. Toda procesorska moč izbranega DSP-ja je sedaj nekako na meji zmoglivosti. Medtem pa bi USB2 standard že ugodil zahtevam po dovolj visoki frekvenci vzorčenja. Vendar pa bilo potrebno sedaj izbrati tudi močnejši procesor, ki bi lahko 1024 vzorcev obdelal v bistveno manj kot 1ms.

# 4 Dodatek

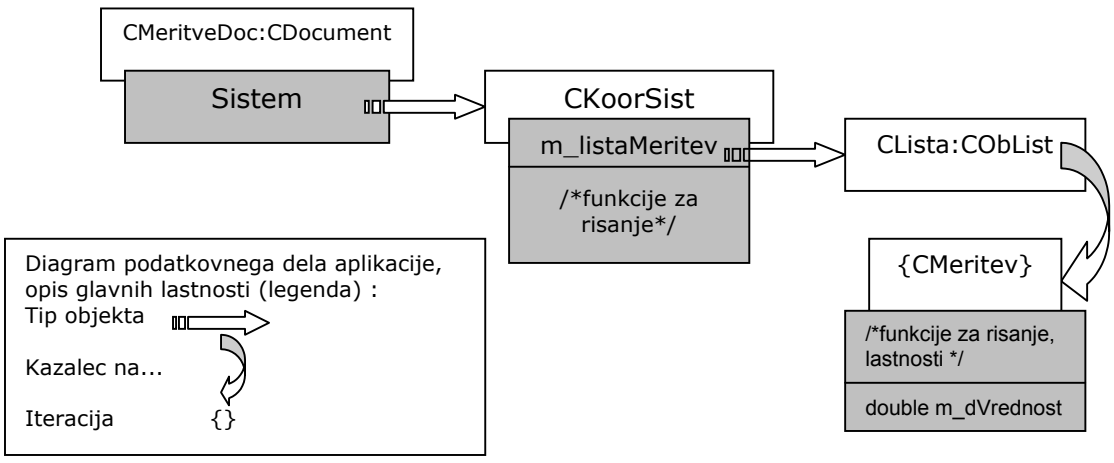
## 4.1 Funkcionalnost in implementacija programa Meritve v1.0 (Aleš Černivec)

Program je napisan v Visual C++ 6.0, in sicer kot MFC (*Mictosoft Foundation Class Library*) aplikacija. Microsoft mi je s svojo knjižnico olajšal delo, tako da sem pridobil predvsem na času razvoja in implementacije aplikacije.

Glavni del aplikacije predstavlja knjižnica *Vzorci.cpp*, kjer je implementiran razred *CVzorci*. Metode razreda dostopajo neposredno do LPT vmesnika, njegove lastnosti (globalne spremenljivke) pa predstavljajo atribute glavne funkcije *CVzorci::diagram_burst_DSP_branje()*. Razred vsebuje tudi privatni metodi, do katerih pa dostopa le že omenjena glavna funkcija. Tako z uporabo te funkcije dobimo informacije o meritvah, shranjene v spremenljivko *CVzorci::nbuffer*, katere pa moramo še grafično prikazati, in sicer v obliki spektralnih črt, torej spektra.

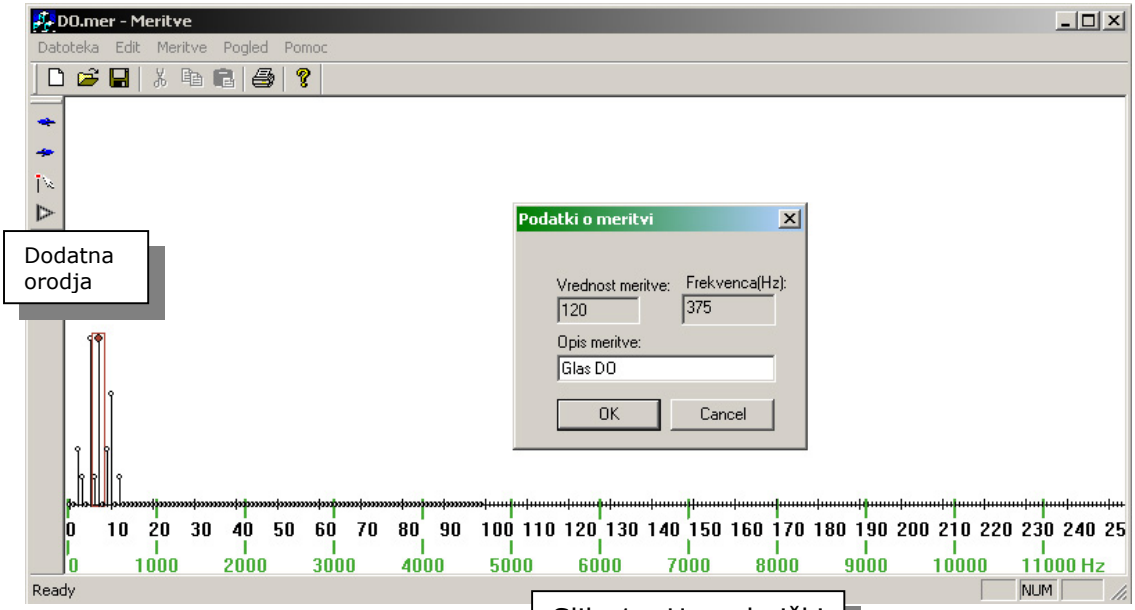
Vsaka spektralna črta je predstavljena kot objekt, ki vsebuje le del informacije o spektru. Ker pa za prikaz celotnega spektra potrebujemo 256 takih spektralnih črt - objektov, sem implementiral listo (izpeljana iz MFC razreda *CObList* ) 256-ih objektov. Vsaki spektralni črti se priredi informacija meritve, ki predstavlja njeno velikost. Ko narišemo vseh 256 meritev, s tem tudi narišemo spekter meritve. Implementacija razredov CMeritev (spektralna črta) in CLista (spekter) se nahaja v podatkovnem delu MFC aplikacije, imenovanemu tudi dokument (*document template*, MeritveDoc.h in MeritveDoc.cpp).

Tako mi je uspelo narisati spekter meritev ob enkratnem zajemu podatkov DSP-ja, kar pa še ni produkt, ki sem ga želel. Želja je bila izdelati aplikacijo, ki je sposobna v realnem času komunicirati z DSP-jem in v realnem času tudi prikazovati spekter. Dosedanji program je bilo treba le še dopolniti. Dodal sem časovnik, ki ga ustvarimo z ukazom *::SetTimer(1, m_nCasovniInterval, NULL)*. Ta časovnik sproži prekinitveno rutino *::OnTimer(UINT nIDEvent)* od izteku *m_nCasovniInterval* milisekund. V prekinitveni rutini narišemo spekter in osvežimo ekran. Tako sem dosegel tekočo animacijo zajemanja spektra, a vendar je nastal ponoven problem. Utripanje slike je onemogočilo popolnoma tekočo animacijo. Problem sem rešil tako, da posamezno spektralno črto nisem risal neposredno na zaslon, temveč v pomnilnik, kjer sem ustvaril celotno sliko spektra in šele nato prikazal sliko na zaslon. Ta tehnika se imenuje listanje (*double buffering*).



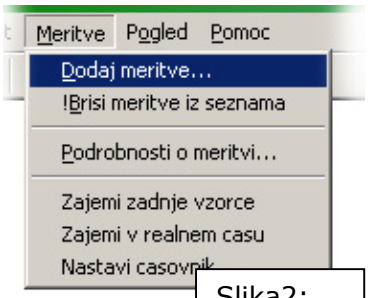
## 4.2 Opis uporabniškega vmesnika

Ker aplikacija dobi svojo vrednost šele z učinkovitim uporabniškem vmesnikom, sem se odločil, da bo ta le najbolj preprost. Odločil sem se za gradnjo vmesnika z enim podatkovnim vmesnikom (SDI aplikacija), ter dvema orodnima vrsticama: standardna




Slika1: Uporabniški vmesnik aplikacije.

orodna vrstica, ter vrstica z dodatnimi orodji za lažje in hitrejše pregledovanje spektra (slika 1). Ko zajamemo vzorec meritev s klikom gumba Zajemi zadnje vzorce na menuju Meritve (slika2), se prikažejo na zaslonu spektralne črte z različnimi višinami, odvisno od vrednosti meritve na določeni frekvenci, ki nam jo je posredoval na LPT vmesnik priključen DSP (na sliki1 je spekter glasu »DO«). Z dvojnim miškinim klikom na določeni spektralni črti, se nam odpre okno Podatki o meritvi (slika1) z raznimi podatki o spektralni črti: Vrednost meritve, Frekvenca meritve in Opis meritve. Slednji podatek lahko spremenimo in tako zaznamujemo določeno spektralno črto z nekim poljubnim opisom. Odprtje okna s podatki o meritvi lahko dosežemo tudi s klikom gumba Podrobnosti o meritvi na menuju Meritve (slika2), vendar mora biti z miškinim klikom že označena spektralna črta. Na menuju Meritve je omogočen tudi zajem meritev v realnem času, in sicer s klikom na gumb Zajemi v realnem času, za katerega smo že pred tem nastavili časovnik (prednastavljena vrednost znaša 15 ms).

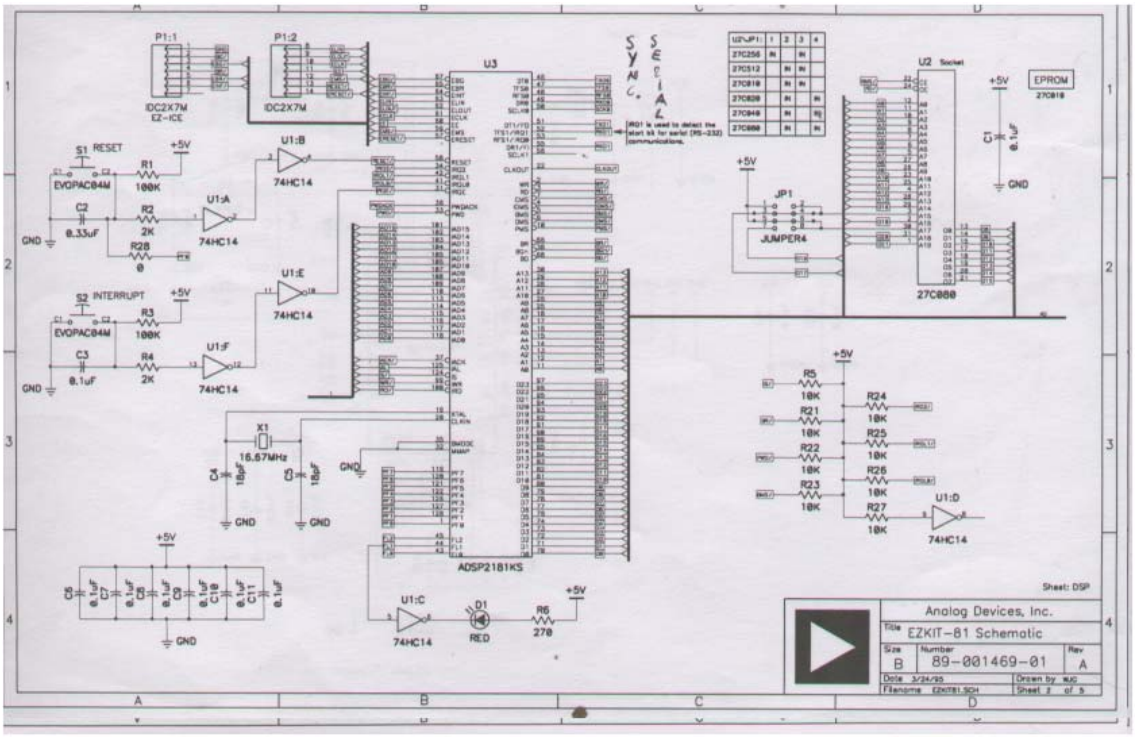



Slika2: Menu Meritve

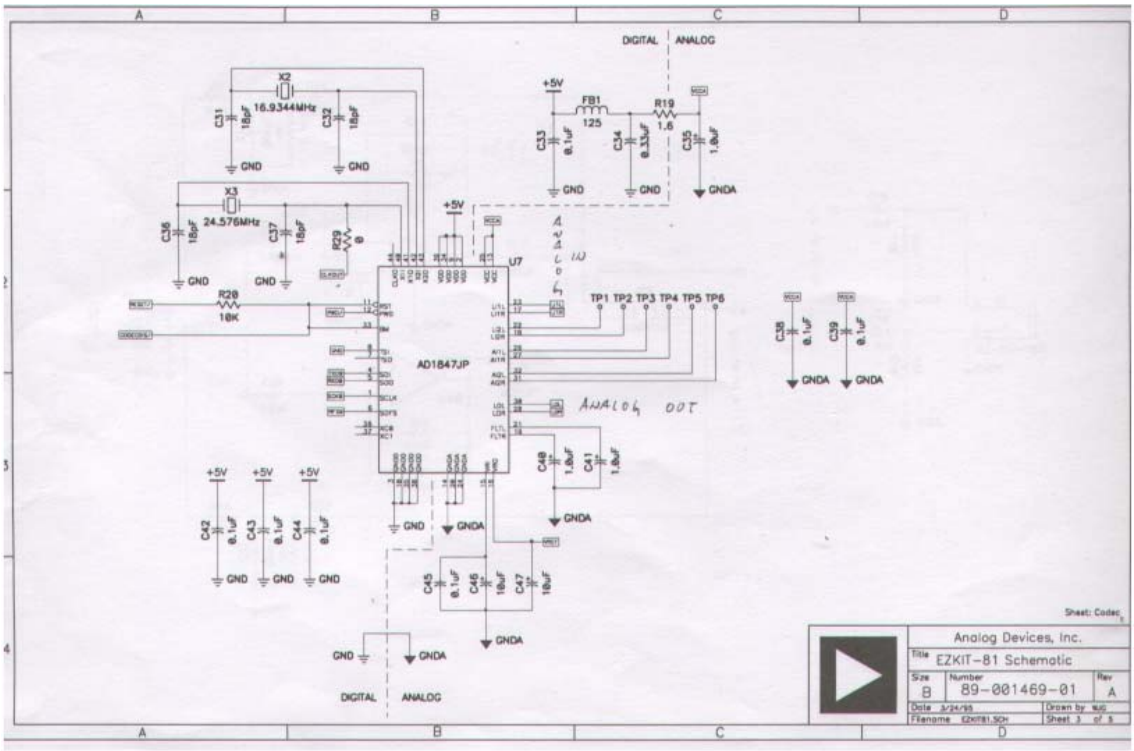
Ko smo zajeli meritev, ga lahko spravimo v datoteko s končnico .MER, ki hrani vse podatke o 256 spektralnih črtah, torej celega spektra. Navsezadnje pa je vredno omeniti tudi, da je omogočeno tiskanje spektra na papir s trenutnim aktivnim tiskalnikom z možnostjo predogleda tiskanja.

## 4.3 Skenirane sheme procesorske platforme, codeca in analognega vhodnega dela

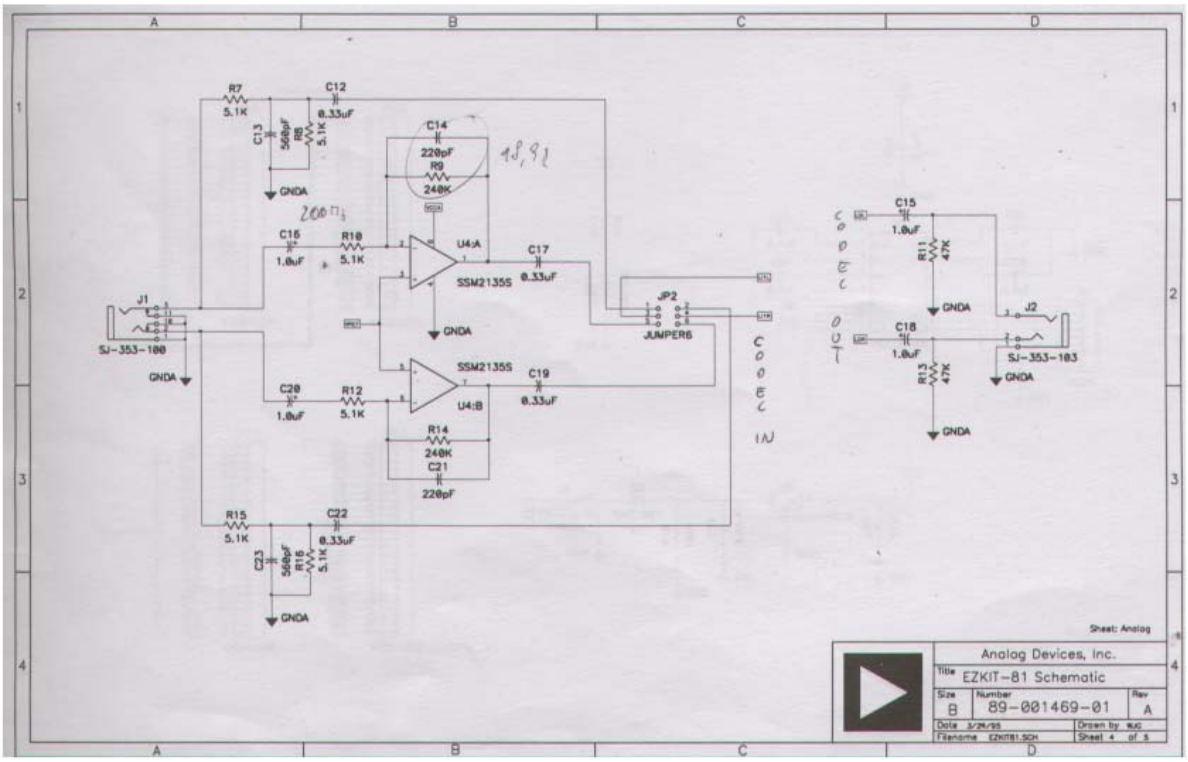
### 4.3.1 Procesorski del:




### 4.3.2 Codec AD1847



### 4.3.3 Analogni vhodni del



# 5 Literatura

ADSP-2100 Family user's manual. Third Edition(9/95). Analog Devices

ADSP-2100 Family (1995). EZ-KIT Lite. Reference Manual. Analog Devices.

Assembler Tools & Simulator Manual. Second Edition(11/94). Analog Devices.

Craig Peacoc (1998) Interfacing the Standard Parallel Port. Htpp://www.senet.com.au~cpeacoc.

Leonardis, S. (1996) Digitalna obdelava signalov. 1. Izdaja. Fakulteta za elektrotehniko.

Engineer to Engineer Note EE-142: Autobuffering, C and FFT on the ADSP-218x. Submited 8/2001 by DTL. www.analog.com/dsp

Engineer To Engineer Note EE-18: Chosing and Using FFTs for ADSP-21xx. Last Modified: 06/18/96. www.analog.com/dsp