VEŽBA 3 – Odabiranje

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Urađena Vežba 2 – Generisanje signala
- Odslušana predavanja iz predmeta OAiS DSP na temu Diskretizacija signala

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Osnovne koncepte prevođenja signala iz analognog u diskretni domen i obrnuto.
- Šta je to odabiranje signala i kako se vrši
- Koja su to ograničenja prilikom odabiranja i šta je teorema o odabiranju
- Šta se dešava sa signalom ukoliko se ta ograničenja ne ispoštuju
- Kako frekvencija odabiranja utiče na kvalitet zvuka
- Šta se dešava kada zvuk odabiramo koristeći jednu frekvenciju, a reprodukujemo koristeći drugu

Kroz praktične primere upoznaćete se i sa osnovnim konceptima programiranja za DSP procesore:

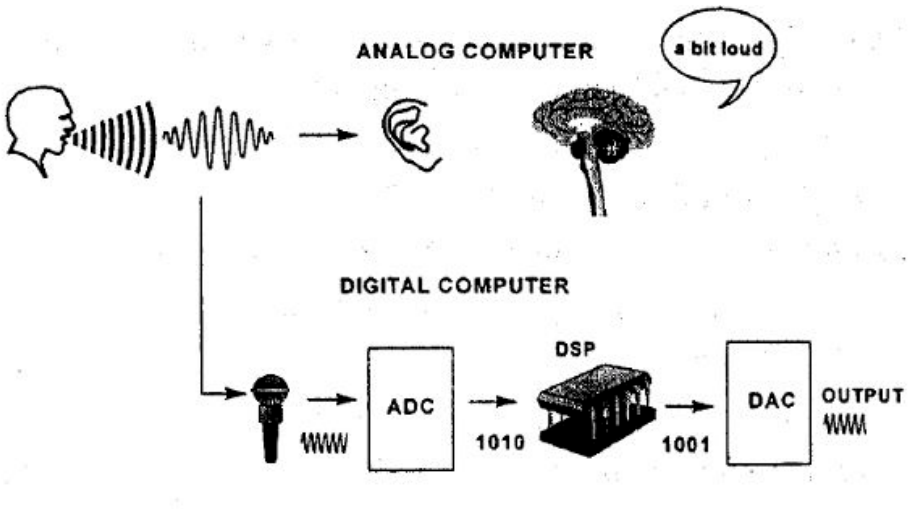
- Koncept kružnih bafera
- Blokovska obrada

Motivacija

Signali u prirodi su uglavnom kontinualne funkcije i predstavljeni su beskonačnim brojem vrednosti. Sa druge strane digitalni računari imaju memoriju ograničenog (konačnog) kapaciteta. Odabiranje (diskretizacija po vremenu) omogućava da se jedan kontinualan signal predstavi sekvencom diskretnih vrednosti. Prilikom odabiranja potrebno je očuvati bitne karakteristike signala u cilju što vernije predstave kontinualnog signala koji posmatramo.

1 TEORIJSKE OSNOVE

Digitalna obrada signala može se vršti na dva načina, a to su obrada signala u realnom vremenu, i obrada signala koja nije u realnom vremenu. Obrada signala koja ne podrazumeva realno vreme vrši se nad signalima koji su ranije prikupljeni i sačuvani u digitalnom obliku. Ovakvi signali obično ne predstavljaju trenutno stanje, i zahtevi za rezultatima obrade ne sadrže vremenska ograničenja. Obrada signala u realnom vremenu podrazumeva čvrste zahteve, i prema hardveru i prema softveru, da svoje zadatke izvrše u određenom vremenskom intervalu. Najčešće obrada signala u realnom vremenu podrazumeva učitavanje signala iz prirode, obradu nad učitanim signalom i njegovu reprodukciju.



Slika 1 – Sistem za digitalnu obradu signala u realnom vremenu

Kao što znamo, svi signali u prirodi su analogni. Dakle ukoliko želimo da vršimo digitalnu obradu signala, pre toga taj signal moramo nekako predstaviti u diskretnom obliku. Proces diskretizacije signala sastoji se iz dva koraka:

- Diskretizacija po vremenu ili **odabiranje**
- Diskretizacija po amplitudi ili **kvantizacija**

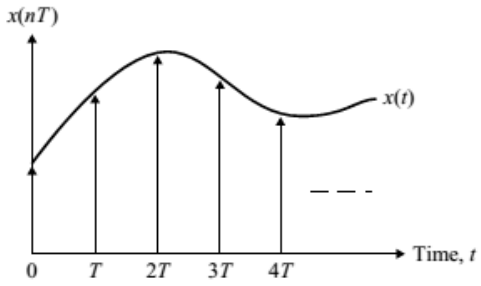
U okviru sistema za digitalnu obradu signala, komponenta koja vrši diskretizaciju signala naziva se analogno-digitalni konvertor (*ADC*). Nasuprot procesa diskretizacije signala postoji i proces za pretvaranje digitalnog signala u analogni, za koji je zadužen digitalno-analogni konvertor (*DAC*). Predmet današnje vežbe jeste diskretizacija signala po vremenu.

1.1 Diskretizacija po vremenu - odabiranje

Diskretizacija po vremenu je proces koji nam omogućava da se jedan kontinualni signal $x(t)$ predstavi sekvencom digitalnih vrednosti $\{x_n\}$ koje predstavljaju vrednosti signala u ekvidistantnim momentima vremena $x_n = x(nT_s)$, gde je T_s period odabiranja.

Na slici 1 je prikazan sistem za obradu audio signala. Kao ulazni senzor imamo mikrofon koji predstavlja analognu komponentu koja pretvara zvučni talas u električni signal. Postoje različite vrste mikrofona, međutim, ono što je zajedničko za sve jeste da u svakom trenutku vrednost jačine električnog signala

oslikava trenutnu jačinu zvučnog talasa koji dopire do mikrofona. U datom sistemu odabiranje signala po vremenu se svodi na čitanje vrednosti jačine električnog signala na izlazu iz mikrofona na svakih T_s.



Slika 2 – Primer analognog signala *x(t)* i njegove diskretne prezentacije *x(nT)*

Period odabiranja možemo izraziti i kao: $T_S = \frac{1}{f_S}$ gde f_s predstavlja frekvenciju odabiranja. Frekvencija odabiranja nam govori sa koliko odbiraka će biti predstavljen jedan sekund signala u diskretnom domenu.

1.2 Teorema o odabiranju

Prilikom izbora frekvencije odabiranja postoji kriterijum koji se mora ispoštovati. Ovaj kriterijum opisan je teoremom o odabiranju koja glasi:

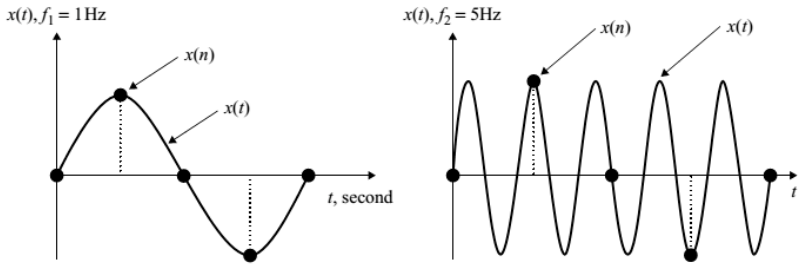
Frekvencija f_s odabiranja signala x(t) treba da je dva puta veća od najveće frekvencije prisutne u signalu da bi se taj signal mogao rekonstruisati bez izobličenja iz signala x_s(t) dobijenog odabiranjem signala x(t).

$$f_S \geq 2f_M$$

Granična frekvencija određena frekvencijom odabiranja $f_N = \frac{f_S}{2}$ naziva se Nikvistova frekvencija.

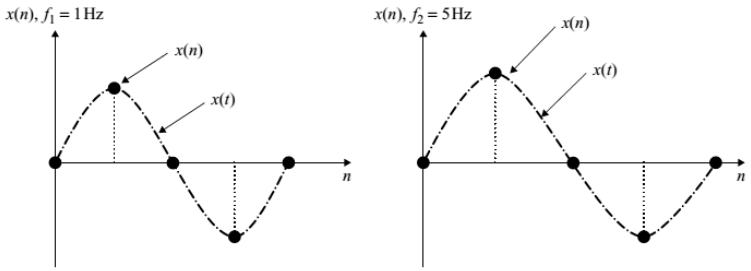
Šta se dešava ukoliko signal koji se odabira sadrži frekvencije više od Nikvistove?

Uzmimo za primer dva sinusna signala frekvencija f_1=1Hz i f_1=5Hz. Po teoremi odabiranja minimalna frekvencija odabiranja za ova dva signala bila bi 10Hz. Umesto toga izvršićemo odabiranje frekvencijom 4Hz.



Slika 3 – Originalni analogni signali

Na slici 3 prikazani su dati signali u vremenskom domenu, sa obeleženim vrednostima odbiraka. Slika 4 sadrži prikaz vrednosti odbiraka diskretnog signala i analogni signal rekonstruisan iz datih odbiraka.



Slika 4 – Odbirci signala i rekonstruisan analogni signal

Kao što možemo zaključiti sa datih slika, originalni signal se može rekonstruisati za sinus frekvencije f_1 = 1Hz. Međutim za signal frekvencije f_2 = 5Hz, dobijeni rekonstruisani signal je takođe sinusni signal frekvencije 1Hz. Za ovakvu pojavu kažemo da je došlo do preklapanja signala frekvencije f_2 u signal frekvencije f_1.

Preklapanje (eng. *Aliasing*) predstavlja izobličenje signala usled neispunjenosti uslova teoreme odabiranja, koje se manifestuje preklapanjem spektralne komponente signala koja je van opsega $f \leq \frac{f_s}{2}$ u neku od spektralnih komponenti u datom opsegu.

U većini realnih sistema za digitalnu obradu signala prilikom A/D konverzije, pre koraka odabiranja signal se propušta kroz pojasno propusni filtar, koji odseca sve spektralne komponente signala van opsega definisanog teorijom odabiranja. Ovakav filtar naziva se „anti-aliasing“ filtar. Upotreba „anti-aliasing“ filtra rešava problem preklapanja neželjenih frekvencija u opseg koji je od značaja, međutim za posledicu ima pojavu da sve spektralne komponente signala više od Nikvistovog kriterijuma budu eliminisane iz signala.

1.3 Podešavanje frekvencije odabiranja kod TMS320C5535

Kao što je prikazano u okviru vežbe 1, TMS320C5535 eZdsp razvojna ploča poseduje TLV320AIC3204 audio kodek koji sadrži *ADC* i *DAC*. Podešavanje željene frekvencije odabiranja vrši se konfiguracijom ove komponente. Konfiguracija kodeka vrši se upisom odgovarajućih vrednosti na predefinisane lokacije u njegovoj memoriji. Memorijska mapa koja sadrži opis značenja i mogućih vrednosti za svaku memorijsku lokaciju data je u pratećem dokumentu koji sadrži tehnički opis TLV320AIC3204 kodeka. U okviru biblioteke za rad sa razvojnom pločom realizovane su funkcije za inicijalizaciju kodeka, inicijalizaciju komunikacije sa kodekom i promenu frekvencije odabiranja i pojačanja ulaznog signala. Funkcije za inicijalizaciju nemaju parametre i date su sa:

- `void aic3204_hardware_init();`
- `void aic3204_init();`

Funkcija za postavljanje, odnosno promenu frekvencije odabiranja i pojačanja ulaznog signala data je sa:

- `unsigned long set_sampling_frequency_and_gain(unsigned long, unsigned int);`

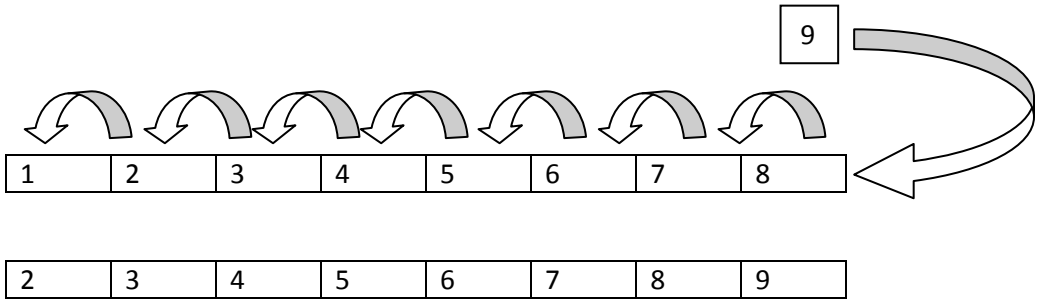
Prvi parameter funkcije predstavlja željenu frekvenciju odabiranja u Hz. Podržane su sledeće vrednosti: 48000, 24000, 16000, 12000, 9600, 8000, 6857. Pre poziva ove funkcije

Pre poziva ove funkcije neophodno je izvršiti inicijalizaciju kodeka. Dat je primer sekvence poziva koji rezultuje uspešno inicijalizovanim i konfigurisanim TLV320AIC3204.

```
aic3204_hardware_init();

aic3204_init();

set_sampling_frequency_and_gain(FS_24kHz, GAIN_IN_dB);
```

1.4 Dodatak 1: Kružni bafer

U oblasti digitalne obrade signala česta je potreba za čuvanjem prethodnih N vrednosti datog signala (jedan od najčešćih primera jeste filtar sa konačnim odzivom). Pomenute vrednosti možemo čuvati upotrebom prostog linearnog bafera tako što ćemo nakon dobijanja svakog odbirka trenutne vrednosti bafera pomeriti za jedno mesto u levo, i novopridošli odbirak smestiti na kraj.

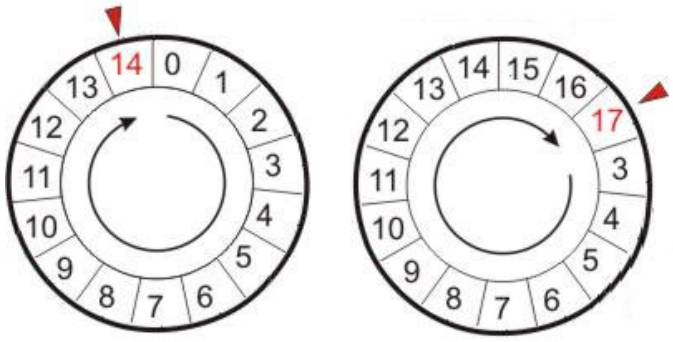


Slika 5 – Pomeraj kod smeštanja elemenata na kraj niza

Ovakav pristup može biti veoma neefikasan u slučaju velikih bafera, jer za svaki primljeni odbirak, vrši se N-1 pomeranja (gde je N veličina bafera).

Drugi pristup, znatno efikasniji jeste korišćenje tzv. kružnog ili cirkularnog bafer.

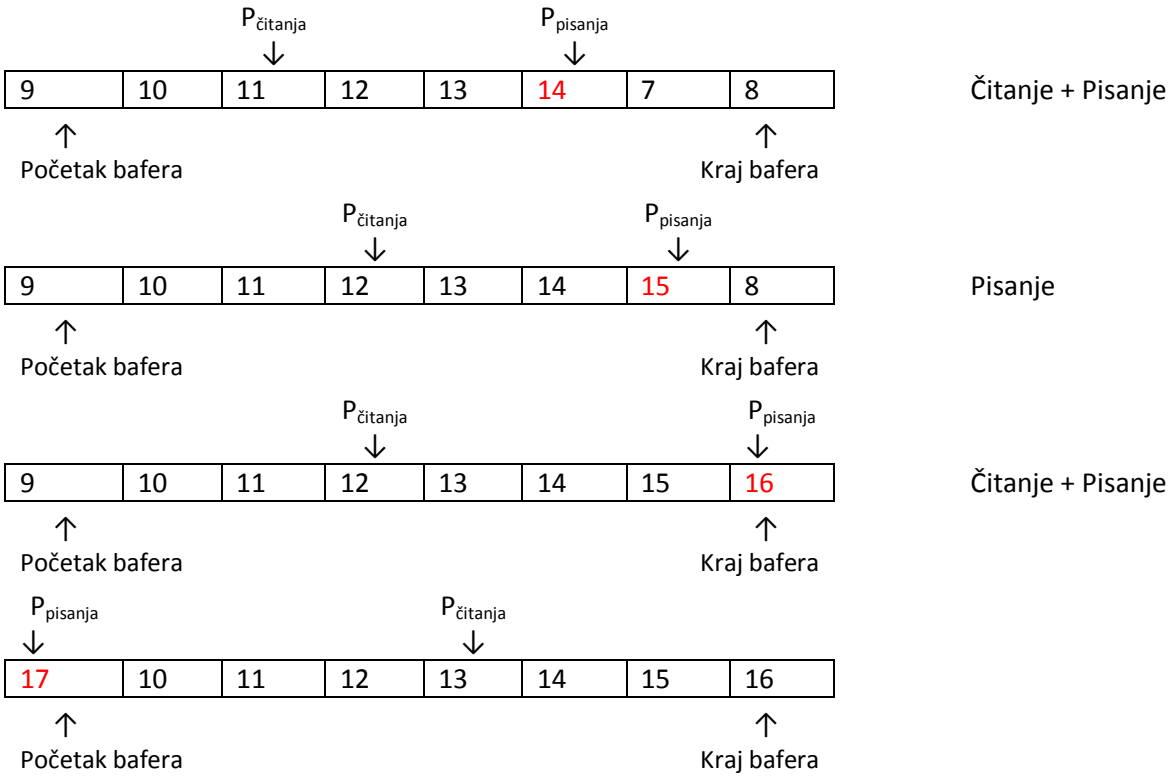
Koncept kružnog bafera zasniva se na posmatranju alociranog memorijskog prostora kao krug.



Slika 6 – Prikaz kružnog bafera

Kružni bafer je realizovan tako što uz alocirani memorijski prostor pamtimo i dva pokazivača, pokazivač upisa i pokazivač čitanja. Pokazivač upisa pokazuje na lokaciju koja će biti korišćena za smeštanje naredne vrednosti, a pokazivač čitanja najstariju nepročitanu vrednost u baferu. Pokazivač pisanja uvećava se nakon svake operacije pisanja, a pokazivač čitanja nakon svake operacije čitanja. Pokazivači

se uvećavaju po modulu veličine bafer-a. Dakle pokazivač se uvećava dok ne pokazuje na poslednju alociranu lokaciju u memoriji (pokazivač == kraj bafera). U narednom pokušaju povećanja pokazivača, umesto uvećanja za jednu lokaciju, pokazivaču će biti dodeljena vrednost adrese prve lokacije u alociranom baferu (pokazivač = početna adresa bafera). Na narednoj slici prikazan je primer čitanja i pisanja vrednosti u i iz kružnog bafera.



Slika 7 – Prikaz rada kružnog bafera

1.4.1 Implementacija kružnog bafera u programskom jeziku C

U ovom odeljku dat je primer jednostavne implementacije kružnog bafera. U osnovi implementacije nalazi se struktura koja sadrži opis instance kružnog bafera.

U programskom jeziku C strukture predstavljaju složene promenljive definisane od strane korisnika. Struktura predstavlja skup polja, gde svako polje može biti promenljiva bilo kog tipa, niz ili druga struktura. Strukture se definišu koristeći ključnu reč ***struct,*** nakon toga sledi (opciono) naziv strukture, potom u vitičastim zagradama deklaracije svih polja strukture, i na kraju nazivi promenljivih koje predstavljaju instance date strukture.

```
struct [naziv strukture]
{
    definicija polja;
    definicija polja;
    ...
    definicija polja;
} [naziv jedne ili više promenljivih];
```

Definicija svakog od polja izgleda kao definicija promenljive. Poljima strukture pristupa preko naziva polja. Dva operatora za pristup poljima struktura su: "." i "->". Prvi operator se koristi kada se adresira polje koristeći instancu strukture, a drugi kada se vrši adresiranje koristeći pokazivač na strukturu. Dat je primer definicije strukture i tri načina pristupa njenim elementima. Treći način pristupa podrazumeva dereferenciranje pokazivača pre pristupa elementu strukture.

```
struct struktura1
{
    Int16 x;
    Int16 y;
    Int16 z;

} s;

struktura1* p_s;

s.x = 2;
p_s->y = 3;
(*p_s).z = 4;
```

Koristeći ključnu reč *typedef* moguće je definisati novi tip u okviru programskog jezika C. Jednom definisan tip koristeći ovu ključnu reč može se u daljem kodu koristiti na isti način kao osnovni tipovi. Dat je primer definisanja novog tipa koristeći strukturu iz prethodnog primera.

```
typedef struct struktura1
{
    Int16 x;
    Int16 y;
    Int16 z;

} strukt_tip;

strukt_tip s1;
```

Kao što je već pomenuto, instanca kružnog bafera u predloženoj implementaciji predstavljena je instancom strukture *CircularBuffer,* koja sadrži polja: niz za smeštanje podataka (*data*), indekse čitanja (*read_indx*) i pisanja (*write_indx*) i veličinu bafera (*size*).

```
typedef struct CircularBuffer_
{
    Int16 *data;
    Int16 read_indx;
    Int16 write_indx;
    size_t size;
} CircularBuffer;
```

Pored ove strukture date su funkcije za inicijalizaciju bafera (*circular_buffer_init*), čitanje (*circular_buffer_read*) i pisanje (*circular_buffer_write*) elemenata u bafer. Kod ove implementacije ukoliko pokazivači pokazuju na istu lokaciju smatra se da je bafer prazan i nije moguće čitanje podataka

iz njega. Ukoliko nakon upisa novog elementa pokazivači pokazuju na istu lokaciju smatra se da je bafer prepunjen, i pokazivač čitanja se uvećava za jedan, kako bi pokazivao na najstariji element u baferu. Sve tri funkcije su parametrizovane, struktura *CircularBuffer* se prosleđuje kao parametar funkcije. Odatle sledi da je moguće ove funkcije koristiti nad različitim instancama kružnog bafera.

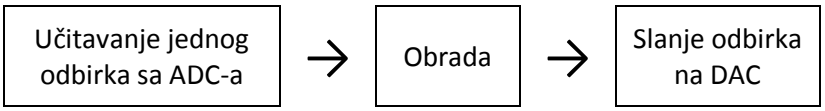
```
void circular_buffer_init(CircularBuffer* circular_buffer, Int16* buffer, size_t
size)
{
    memset(buffer, 0, size*sizeof(Int16));
    circular_buffer->data = buffer;
    circular_buffer->size = size;
    circular_buffer->read_indx = 0;
    circular_buffer->write_indx = 0;
}

Int16 circular_buffer_read(CircularBuffer* circular_buffer)
{
    Int16 ret;
    if(circular_buffer->read_indx != circular_buffer->write_indx)
    {
        ret = circular_buffer->data[circular_buffer->read_indx];
        circular_buffer->read_indx++;
        circular_buffer->read_indx = circular_buffer->read_indx % (circular_buffer-
>size);
        return ret;
    }
    else
        return 0;
}

void circular_buffer_write(CircularBuffer* circular_buffer, Int16 value)
{
    circular_buffer->data[circular_buffer->write_indx] = value;
    circular_buffer->write_indx++;
    circular_buffer->write_indx = circular_buffer->write_indx % (circular_buffer-
>size);
    if(circular_buffer->read_indx == circular_buffer->write_indx)
    {
        circular_buffer->read_indx++;
        circular_buffer->read_indx = circular_buffer->read_indx % (circular_buffer-
>size);
    }
}
```

Priloženu implementaciju moguće je dodatno optimizovati dodavanjem ograničenja da je veličina bafera stepen broja 2. U tom slučaju umesto operatora „%“ moguće je koristiti logički operator i nad bitima „&“.

1.5 Dodatak 2: Blokovska obrada

Tipična obrada signala u realnom vremenu podrazumeva učitavanje vrednosti odbirka na ulazu u sistem, vršenje obrade i slanje vrednosti obrađenog odbirka na izlaz iz sistema. Međutim u ovom slučaju vreme

trajanja obrade ograničeno je periodom odabiranja T_s (=1/f_s). Ukoliko bi vreme trajanja obrade bilo veće od perioda odabiranja, dolazi do preskakanja odbiraka i izobličenja signala.



Slika 8 - Tipična obrada u okviru DSP sistemima

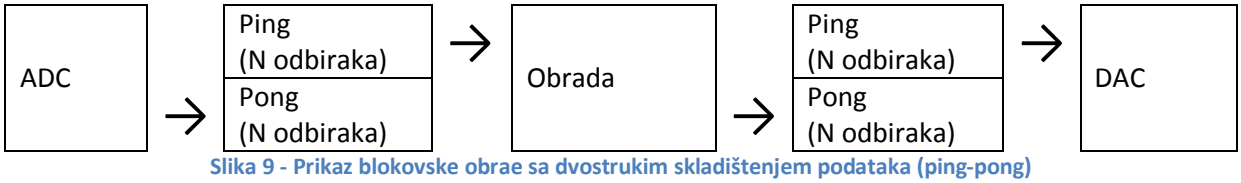
To je jedan od razloga zbog čega se u oblasti digitalne obrade signala često koristi pristup blokovske obrade signala. Blokovska obrada podrazumeva učitavanje bloka od N odbiraka, vršenje obrade nad blokom i prosleđivanje obrađenih podataka na izlaz. U tom slučaju dobija se period od NT_s da se obradi N odbiraka.

Većina DSP arhitektura sadrži podršku za direktan pristup memoriji bez posredstva glavnog procesora (eng. *Direct Memory Access* ili *DMA*). DMA funkcioniše tako što DMA kontroleru prosledimo početnu adresu podataka koje želimo da prepišemo, početnu adresu odredišta i količinu podataka. Adresa može biti adresa u memoriji ili adresa perifernog uređaja (u našem slučaju ADC i DAC). Zahvaljujući ovom proširenju moguće je vršiti učitavanje podataka i obradu u paraleli.

Jedna od tehnika koja se može koristiti kako bi se omogućilo vršenje učitavanja odbiraka, obrade podataka i prosleđivanja izračunatih izlaznih odbiraka na DAC u paraleli, jeste tehnika dvostrukog skladištenja podataka ili *Ping Pong* tehnika. Ukoliko se vrši obrada nad blokovima od N odbiraka, potrebno je za svaki transfer zauzeti memoriju za 2*N odbiraka. Zauzeta memorija sastoji se iz dva dela po N odbiraka, *Ping* i *Pong*. Dvostruko skladištenje podataka funkcioniše tako što DMA kontroler čita vrednosti sa ADC konvertora i piše u Ping bafer. U isto vreme vrši se obrada nad odbircima koji se nalaze Pong baferu. Kada se obrada i DMA transfer završe, dolazi do zamene, DMA kontroler piše u Pong bafer, dok se obrada vrši nad odbircima koji se nalaze u Ping baferu. Isto važi i za izlazne odbirke, dok se odbirci izračunati u toku obrade zapisuju u Ping bafer, DMA kontroler vrednosti smeštene u Pong bafer prosleđuje na DAC, i obrnuto.

Prva iteracija:

Druga iteracija:



Slika 9 - Prikaz blokovske obrae sa dvostrukim skladištenjem podataka (ping-pong)

Konfiguracija DMA kontrolera, inicijalizacija prenosa i čitanje i pisanje vrednosti realizovano je u okviru *ezdsp5535_aic3204_dma.c* datoteke. Podrazumevani broj kanala na ulazu i izlazu iz sistema je 2 (levi i desni), iz tog razloga funkcije za čitanje i pisanje kao parametar primaju po dva memorijska niza. Dat je primer inicijalizacije DMA kontrolera, čitanja i pisanja odbiraka.

```
aic3204_hardware_init();

aic3204_init();

aic3204_dma_init();

set_sampling_frequency_and_gain(FS_24kHz, GAIN_IN_dB);

aic3204_read_block(InputBufferL, InputBufferR);

... Obrada ...

aic3204_write_block(OutputBuffer, OutputBuffer);
```

Za čitanje odbiraka iz ping-pong bafera koristi se funkcija:

- `void aic3204_read_block(Int16* buffer_left, Int16* buffer_right);`

koja dobavlja N odbiraka predstavljenih celobrojno sa 16-bita sa levog i desnog kanala.

Za prosleđivanje odbiraka na DAC konvertor koristi se:

- `void aic3204_write_block(Int16* buffer_left, Int16* buffer_right);`

Veličina bloka (*N*) definisana je sa AUDIO_IO_SIZE. Ova vrednost korišćena je prilikom inicijalizacije DMA kontrolera za čitanje i pisanje.

U okviru funkcija za čitanje i pisanje odbiraka je implementirana opisana Ping Pong tehnika.

2 ZADACI

2.1 Zadatak 1

1. Uvući projektni zadatak 1 u radni prostor.
2. Uz pomoć funkcija realizovanih u prethodnoj vežbi generisati signale:
   - Multiton signal u 2000 odbiraka, amplitude 10, početne frekvencije 1 kHz, sa korakom između harmonika *df*=5 kHz, sa frekvencijom odabiranja 48 kHz
   - Logaritamski „sweep“ signal u 2000 odbiraka, amplitude 10, početne frekvencije f_1=1 kHz, krajnje frekvencije f_2=10 kHz, frekvencije odabiranja 48 kHz

Napomena: U prethodnoj vežbi funkcije za generisanje signala realizovane su tako da kao parametar primaju normalizovanu frekvenciju izračunatu sa $\frac{f}{f_s}$, gde je f željena frekvencija, a f_s zadata frekvencija odabiranja.

Generisani Multiton signal predstavlja kako bi izgledao u diskretnom obliku analogni multiton signal zadate frekvencije da je odabran sa frekvencijom odabiranja 48 kHz.

Napravljen je novi signal uzimanjem svakog drugog odbirka generisanog multiton signala.

Novonastali signal predstavlja kako bi izgledao u diskretnom obliku analogni multiton signal zadate frekvencije da je odabran sa frekvencijom odabiranja 24 kHz (odbacivanje svakog drugog odbirka rezultuje duplo većim periodom odabiranja, odnosno duplo nižom frekvencijom).

3. Prikazati generisani *multiton_fs48kHz* i novonastali *multiton_fs24kHz* signal u vremenskom i frekventnom domenu. Komentarisati rezultate.

Napomena: Za prikaz signala u vremenskom domenu koristiti alat *Single Time,* a za prikaz u frekventnom *FFT Magnitude.* I kod jednog i kod drugog količina podataka (*Acquisition Buffer Size*) treba da odgovara veličini vašeg signala (broju odbiraka). Polje *Dsp Data Type* postavite na *32 bit floating point* ukoliko su odbirci vašeg signala tipa *float* ili *16 bit signed integer* ukoliko su tipa *Int16*. Polje *Starting Address* treba da sadrži adresu početka niza (vrednost ili naziv promenljive). Polje *Sampling Frequency* postaviti na vrednost frekvencije odabiranja koju ste koristili za odabiranje vašeg signala.
Za *Single Time* alat polje koje određuje koliko od pročitanih odbiraka želimo da prikažemo (*Display Data Size*) takođe treba da odgovara veličini signala, jer želimo da prikažemo čitav signal.
Za *FFT Magnitude,* polje *Magnitude Data Plot Style* postaviti na *Bar. FFT Order* postaviti na 10. Ova podešavanja koristiti prilikom iscrtavanja svih signala.

4. Na osnovu generisanog logaritamskog „sweep“ signala napraviti nove signale:
    - Uzimanjem svakog drugog odbirka
    - Uzimanjem svakog trećeg odbirka

Novonastali signali predstavljaju kako bi izgledao „sweep“ signal zadate frekvencije odabran frekvencijama odabiranja 24 kHz i 16 kHz redom.

5. Prikazati sva tri „sweep“ signala u vremenskom i frekventnom domenu. Komentarisati rezultate.

Očekivani izlaz iz zadatka:

- „*screenshot*“ grafika svakog signala za koji je rečeno da ga je potrebno prikazati.
- Tekstualna datoteka koja sadrži komentare.

2.2 Zadatak 2

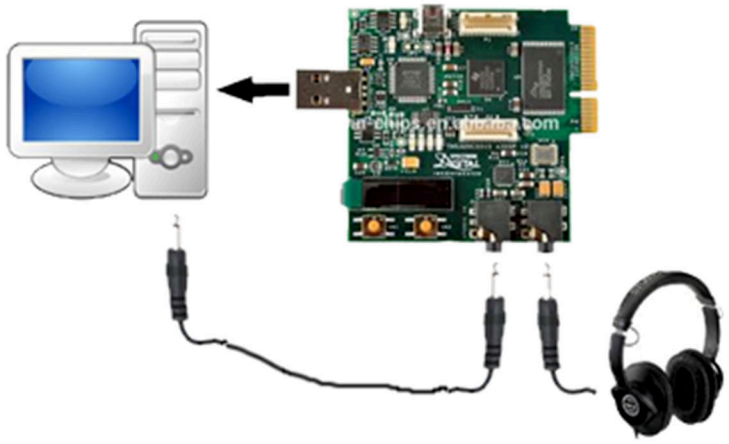
1. Uvući projektni zadatak 2 u radni prostor.

Tok izvršenja datog programa sastoji se iz sledećih koraka:

- Inicijalizacija DSP uređaja
- Inicijalizacija modula zaduženog za AD i DA konverziju (*aic3204*)
- Postavljanje željene vrednosti frekvencije odabiranja i pojačanja

- Preuzimanje po jednog odbirka sa audio ulaza (AD konvertora) i prosleđivanje na audio izlaz (DA konvertor). U okviru promenljivih *buffer_left* i *buffer_right* pamtimo N prethodnih odbiraka, kako bi smo bili u mogućnosti da prikažemo signal u vremenskom i frekventnom domenu upotrebom ugrađenih alata *Code Composer Studo* okruženja.

2. Povezati line-out izlaz vašeg računara na line-in ulaz na DSP uređaju.



Slika 10 - Upotreba računara kao izvor ulaznog signala

3. Na računaru reprodukovati datoteku *dual_sine_6kHz_10kHz.wav* (ova datoteka sadrži zbir dva sinusna signala, frekvencija 6kHz i 10kHz). Podesiti ponavljanje reprodukcije datoteke kada dođe do kraja (*Repeat*).
4. Prevesti projektni zadatak i pokrenuti izvršavanje (zadata vrednos frekvencije odabiranja iznosi 24kHz)
5. Zaustaviti program koristeći tačku prekida. Iscrtati trenutnu vrednost snimljenog signala na jednom kanalu (*buffer_left ili buffer_right*) u vremenskom i frekventnom domenu.
6. Promeniti vrednost frekvencije odabiranja na 16kHz.
7. Prevesti ponovo projektni zadatak, i pokrenuti izvršavanje.
8. Ponoviti korak 5.
9. Ponoviti korake 6, 7 i 8 za frekvenciju odabiranja 8kHz.
10. Komentarisati prikaze signala u vremenskom i frekventom domenu, i subjektivni utisak nakon slušanja zvuka sa različitim frekvencijama odabiranja.
11. Reprodukovati datoteku linear_sweep_20Hz_20kHz.mp3
12. Pokrenuti izvršavanje programa za vrednosti frekvencije odabiranja 24kHz i 16kHz. Na koliko sekundi prestaje da se čuje zvuk sa jednom frekvencijom, a na koliko sa drugom? Zašto?

2.3 Zadatak 3

1. Uvući projektni zadatak 3 u radno okruženje.

U okviru datog projektnog zadatka realizovano je snimanje zvuka u trajanju od 10 sekundi i reprodukcija snimljenog zvuka. Za smeštanje u memoriju zvučnog signala odabranog frekvencijom odabiranja 16kHz, predstavljenog odbircima veličine 16 bita (Int16) potrebno je:

$$16000\ odbiraka \times 10\ sekundi \times 16\ bita\ po\ odbirku = 320KB$$

S obzirom da je veličina RAM memorije uređaja TMS320C5535 320KB, došlo bi do prepunjavanja memorije. Zbog toga za čuvanje vrednosti signala koristićemo SD karticu. U okviru date biblioteke realizovane su funkcije za inicijalizaciju kartice, čitanje i pisanje podataka na karticu i prekid rada sa karticom.

```
Int16 EZDSP5535_SDCARD_init();
Int16 EZDSP5535_SDCARD_write(Uint32 addr, Uint32 len, Uint16* pWriteBuff);
Int16 EZDSP5535_SDCARD_read (Uint32 addr, Uint32 len, Uint16* pReadBuff );
Int16 EZDSP5535_SDCARD_close();
```

(addr – adresa na kartici, len – dužina bloka u bajtima, pWriteBuff/pReadBuff – pokazivač na bafer sa podacima koje želimo da zapišemo, odnosno u koji želimo da smestimo pročitane podatke).

Proces čitanja i pisanja podataka na SD karticu predstavlja zahtevan zadatak koji traje određeno vreme. Kako bi izbegli kašnjenje i izobličenje signala, korišćena je blokovska obrada

Napomena: Ukoliko kao izvor ulaznog signala koristite mikrofon potrebno je pre pokretanja postaviti vrednost pojačanja na 30dB (*gain* parametar u funkciji *set_sampling_frequency_and_gain*).

2. Prevesti i pokrenuti dati program. Kao izvor ulaznog signala koristiti mikrofon.

Tokom prvih 10s izvršenja programa vrši se snimanje ulaznog signala na SD karticu, nakon toga započinje reprodukcija snimljenog signala.

Cilj ovog zadatka jeste demonstriranje subjektivnog osećaja prilikom slušanja zvuka koji je odabran jednom frekvencijom odabiranja, a zatim reprodukovan koristeći drugu. Da bi se to postiglo potrebno je izvršiti promene objašnjene u naredna dva koraka.

3. U okviru *main* funkcije nakon završetka reprodukcije snimljenog zvuka dodati poziv funkcije za promenu frekvencije odabiranja na duplo manju od frekvencije korišćene za odabiranje ulaznog signala (16kHz/2 = 8kHz). Nakon toga ponoviti petlju u okviru koje je realizovana reprodukcija signala.

Napomena: Nakon promene frekvencije odabiranja potrebno je ponovo inicijalizovati SD karticu (`EZDSP5535_SDCARD_init`) i postaviti vrednost adrese čitanja na početnu (`SD_CARD_START_ADDRESS`).

4. Ponoviti korak 3, s tim da vrednost frekvencije odabiranja treba postaviti na 24kHz.
5. Pokrenuti program i poslušati izlaz.

Šta se dešava sa zvukom kada ga odabiramo jednom frekvencijom, a reprodukujemo drugom? Šta se dešava kada je frekvencija kojom reprodukujemo signal manja, a šta kada je veća od frekvencije odabiranja na ulazu?

2.4 Zadatak 4

1. Izmeniti projektni zadatak 2 tako da se za čuvanje prethodnih N odbiraka signala umesto običnog koristi kružni bafer.

2. Isprobati realizovano rešenje za vrednost veličine bafera od 1024 odbirka.

3 Zaključak

Proces prevođenja signala iz kontinualnog u diskretan domen sastoji iz dva koraka: diskretizacija u vremenu (odabiranje) i diskretizacija po amplitudi (kvantovanje). U okviru ove vežbe obrađen je prvi korak odnosno operacija odabiranja signala. Upoznali ste se sa pojmom frekvencije odabiranja i teoremom koja definiše ograničenja prilikom njenog odabira. Videli ste na primeru kako frekvencija odabiranja utiče na signal.

Prikazani su koncepti upotrebe često upotrebljavanih tehnika u oblasti DSP-a - kružnog bafera i blokovske obrade. Kroz primere ste se upoznali sa nekim od mogućnosti koje nudi biblioteka za rad sa TMS320C5535 eZdsp razvojnom pločom, i to upotreba SD kartice za čuvanje velike količine podataka i upotreba kontrolera za direktan pristup memoriji prilikom čitanja podatka sa audio kodeka.

U narednoj vežbi biće objašnjen drugi korak prevođenja signala iz analognog u diskretni domen, diskretizacija signala po amplitudi.