UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Sigma-delta AD konvertori u mikrokontrolerima serije MSP430

DIPLOMSKI RAD

Mentor:
Prof. dr Nenad Jovičić

Kandidat:
Jovana Savić

Sadržaj

1. Uvod

U ovom radu su obrađeni sigma-delta AD konvertori koji se nalaze u porodici MSP430 na konkretnom primeru mikrokontrolera MSP430i2041. Cilj ovog rada je ispitati princip rada sigma-delta AD konvertora iz teorijske i praktične perspektive. U skladu sa tim napravljen je i projekat koji ga prati. Projekat se sastoji iz programa za mikrokontroler kome je na ulaz AD konvertora prosleđen signal iz generatora signala, koji rezultat konverzije preko UART-a šalje računaru i programa za računar koji date podatke prikazuje u realnom vremenu i upisuje u fajl i potom dodatno obrađuje.

Sigma-delta AD konvertori su konvertori koje odlikuje visoka rezolucija, niska cena i potrošnja i pošto je veći deo ovog konvertora baziran na digitalnim komponentama njihova implementacija u integrisanim kolima je značajno pojednostavljena. Činjenica da je najveći deo konvertora digitalan povlači sa sobom i veću otpornost na promenu temperature. Jako bitna karakteristika ovih konvertora je monotonost, to jest, kada ulazni signal raste, raste i izlaz AD konvertora i obrnuto. Ovo je veoma dobra osobina kada se radi sa sistemima u zatvorenoj sprezi. Ne zahtevaju kola kao što su *sample-and-hold* i za filtriranje signala je najčešće dovoljan jedan kondenzator. Mana ovih konvertora je činjenica da modulator radi na mnogo većoj učestanosti od učestanosti odabiranja. Zbog toga se ovi konvertori koriste onda kada je potrebna preciznost, ali kada signali nemaju visoku učestanost.

U drugom poglavlju je dat kratak pregled arhitekture mikrokontrolera MSP430i2041. Prikazane su komponente mikrokontrolera i dat je kratak opis njihovih mogućnosti. U poglavlju 2.1. je malo detaljnije objašnjen modul koji omogućava UART komunikaciju, obzirom da je korišćen u samom projektu. AD konvertor je detaljno obrađen u narednim poglavljima.

Poglavlje 3. se bavi svim onim što je značajno za digitalnu obradu signala – diskretizacijom signala koja je neophodna da bi se on predstavio u digitalnom sistemu, načinima predstavljanja signala kao i greškama koje su posledica ograničenja samih digitalnih sistema. U odeljku 3.1. je prikazano idealizovano odabiranje signala i teorema odabiranja koja daje jedan od suštinskih uslova koji se mora zadovoljiti da bi se signal verno preneo u digitalni sistem. U odeljku 3.2. je objašnjeno realno odabiranje signala i uticaj konačnog broja odbiraka na rezultate odabiranja. Delovi 3.3. i 3.4. se bave predstavljanjem brojeva u digitalnom sistemu. U odeljku 3.3. su predstavljeni različiti binarni kodovi, dok se odeljak 3.4. bavi greškama koje nastaju usled zaokruživanja, odnosno odbacivanja bitova. Odeljak 3.5. opisuje kvantovanje amplitude ulaznog signala i definiše šum kvantovanja. Na kraju, odeljak 3.6. daje određene metode kojima se može minimizovati greška kvantizacije, s tim da je oblikovanje greške kvantizacije izostavljeno jer je objašnjeno u sklopu poglavlja o sigma-delta AD konvertoru.

Poglavlje 4. se bavi AD konvertorima u opštem slučaju. U ovom poglavlju su definisani osnovni parametri AD konvertora koji se mogu naći u specifikacijama proizvođača. Odeljak 4.1. se bavi izvorima statičkih grešaka, dok su u odeljku 4.2. obrađene dinamičke karakteristike konvertora. Konačno, u odeljku 4.3. su pomenute još neke specifikacije koje opisuju AD konvertore, ali se odnose na prateće komponente, a ne na sam AD konvertor.

U poglavlju 5. je obrađena teorija sigma-delta AD konvertora koji se sastoji od sigma-delta modulatora i digitalnog filtra. U odeljku 5.1. je analiziran sigma-delta modulator u vremenskom domenu, dok je u odeljku 5.2. urađena analiza u Z domenu u sklopu koje je objašenjeno i oblikovanje šuma kao još jedan od načina minimizacije greške kvantizacije. Odeljak 5.3. predstavlja sigma-delta modulatore višeg reda. Dalje je u odeljku 5.4. dat pregled dva najčešća digitalna filtra koja se koriste u sigma-delta AD konvertorima koje proizvodi *Texas Instruments* – *sinc* i *widepass band* filtri. U odeljku 5.5. je dat teorijski osvrt na *sinc* filtar - filtar koji se nalazi u mikrokontroleru MSP430i2041, a odeljak 5.6. prikazuje implementaciju *sinc* filtra u VLSI tehnologiji koju je dao proizvođač *Texas Instruments*. Konačno, odeljak 5.6.1. se bavi implementacijom *sinc* filtara u slučaju sigma-delta AD konvertora koji zahtevaju veći broj bitova za predstavljanje rezultata i kako se u tom slučaju vrši zakruživanje.

Poglavlje 6. se bavi sigma-delta AD konverotrom u mikrokontroleru MSP430i2041. U odeljku 6.1. je data blok šema AD konvertora i objašnjen njegov rad. Iako je u prethodnom poglavlju prikazana implementacija sigma-delta modulatora i *sinc* filtra koji se nalazi u ovom AD konvertoru, u odeljku 6.2. je sistematizovana implementacija u konkretnom mikrokontroleru i date su još neke specifikacije. U odeljku 6.3. su prikazani i objašnjeni različiti režimi rada AD konvertora. U odeljku 6.5. je dat kratak pregled registara AD konvertora i signala koji se nalaze u njima, dok je u odeljku 6.6. dat pregled karakteristika ovog AD konvertora.

Poglavlje 7. daje objašnjenje projekta koji prati ovaj rad. Odeljak 7.1. objašnjava program za mikrokontroler, dok je u odeljku 7.2. opisan softver koji vrši obradu podataka na računaru. Rezultati testiranja su predstavljeni u odeljku 7.3.

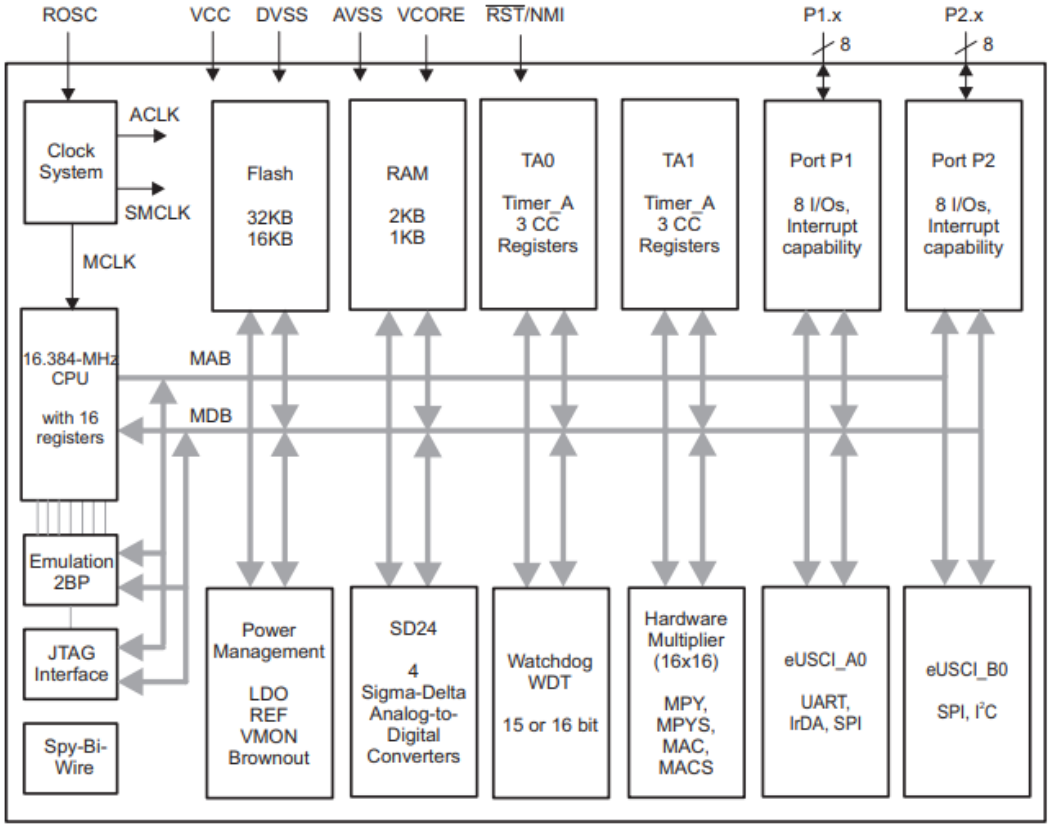
Zaključak rada je dat u poglavlju 8.

U poglavlju 9. je dat pregled korišćene literature, a u prilogu kôd koji predstavlja suštinski deo projekta.

2. Arhitektura mikrokontrolera msp430i2041

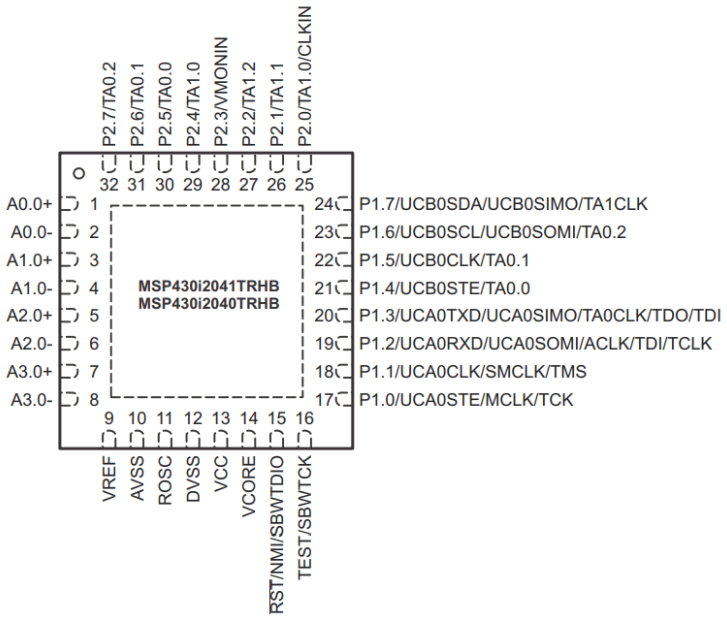
Mikrokontroleri msp430i204x, msp430i203x i msp430i202x čine jednu porodicu. Sve njih karakteriše 16-bitni RISC procesor, signal takta baziran na digitalno kontrolisanom oscilatoru, sistem za praćenje napona napajanja sa ugrađenim referentnim naponom i monitorom napona, dva do četiri 24-bitna sigma-delta AD konvertora, temperaturni senzor, 16-bitni hardverski množač, dva 16-bitna tajmera, jedan eUSCI-A (UART, IrDA, SPI) modul i jedan eUSCI-B (SPI, , I2C) modul, *watchdog* tajmer i do 16 ulazno-izlaznih pinova.

Na slici 2.1. je prikazan funkcionalni blok dijagram mikrokontrolera msp430i2041 i msp430i2040 u RHB pakovanju. Mikrokontroler iz PW paketa ima gotovo isti blok dijagram, jedina razlika je u tome što u ovom paketu na portu P2 postoje 4 ulazno-izlazna pina, umesto 8.



Slika 2.1. *Blok dijagram mikrokontrolera mspe430i2041 i msp430i2040* (Texas Instruments, 2020)

Na slici 2.2. prikazan raspored pinova mikrokontrolera msp430i2041 i msp430i2040 u RHB pakovanju.



Slika 2.2. *Raspored pinova mikrokontrolera msp430i2041 u RHB pakovanju* (Texas Instruments, 2020)

- **Procesor** (eng. *Central Processing Unit – CPU*) je RISC (eng. *Reduced Instruction Set Computer*), sa 27 instrukcija i 7 modova adresiranja. Arhitektura je ortogonalna, odnosno, moguće je koristiti svaku instrukciju sa svakim modom adresiranja. Date su instrukcije koje omogućavaju pristup svim registrima opšte namene, kao i registru programske statusne reči, statusnim registrima i stek pointer registru. Sve operacije sa registrima traju tačno jedan ciklus. Registri su 16-bitni, kao i adresna magistrala i magistrala podataka. Zato je moguće koristiti i podatke veličine bajta i reči. Osim toga, postoje i generatori konstanti koji omogućavaju generisanje šest najčešćih konstanti odgovarajućim instrukcijama, bez potrebe da se pristupa memoriji da bi se dobile njihove vrednosti.

- **Generator takta** može da koristi unutrašnji ili spoljašnji oscilator koji se dovodi na CLKIN pin. Unutrašnji oscilator može da koristi unutrašnji otpornik ili spoljašnji koji se povezuje preko ROSC pina. Unutrašnji oscilator ima frekvenciju 16.384 MHz i generiše četiri taktna signala. Na raspolaganju su ACLK frekvencije 32 kHz i MCLK i SMCLK koji mogu imati od 1.024 MHz do 16.384 MHz u zavisnosti od toga kako se programiraju. Osim ovih izvora takta postoji i jedan taktni signal koji se prosleđuje AD konvertoru i njegova frekvencija je 1.024 MHz.

- **Kolo za kontrolu napona napajanja** (eng. *Power Management Module – PMM*) se koristi za generisanje napona koji se koristi za rad unutrašnjih komponenti koji iznosi 1.8 V od datog napona napajanja koji može biti od 2.2 V do 3.6 V. Ovo kolo takođe obezbeđuje i BOR signal koji je zadužen da kolo postavi u inicijalno reset stanje po dovođenju napona napajanja. Moguće je i pratiti

napon koji je doveden na VMONIN pin i porediti ga sa posebnim unutrašnjim referentnim naponom od 1.16 V i generisati prekid u zavisnosti od vrednosti dovedenog napona. Ovaj unutrašnji referentni napon se koristi i kao referentni napon AD konvertora, kao i za rad integrisanog temperaturnog senzora.

- Mikrokontroler poseduje dve vrste **memorije** – Flash i RAM. Konkretan mikrokontroler poseduje 32 KB flash memorije i 2 KB SRAM memorije. Flash memorija je stalna memorija – njen sadržaj se ne gubi kada nestane napajanje i zbog toga se u njoj čuva program mikrokontrolera. SRAM memorija je memorija kod koje sadržaj nestaje onda kada se izgubi napajanje, ali je znatno brža pa se koristi tokom izvršavanja programa. Dozvoljen je i upis u flash memoriju, ali taj upis se vrši eksplicitno, odnosno pristupa joj se kao posebnoj periferiji uz pomoć kontrolera.

- MSP403i2041 ima 16 **ulazno-izlaznih digitalnih priključaka** koji su ukombinovani u dva paralelna porta. Oni se mogu programirati kao ulazni ili izlazni i mogu generisati prekide, pri čemu se može izabrati da li će se prekid generisati na uzlaznu ili silaznu ivicu. Portovi imaju posebne registre za ulaz, odnosno za izlaz, koji se koriste u skladu sa izabranim smerom komunikacije. Kao što se vidi sa slike 2.2. ovi pinovi se mogu koristiti kao ulazno-izlazni ili kao pinovi nekih drugih komponenti (npr. TX i RX ulazi kada se koristi UART).

- ***Watchdog* tajmer** je supervizorsko kolo koje nadgleda rad procesora. Ovaj tajmer nakon određenog vremena (koje se može isprogramirati) generiše prekid. Odgovor procesora na prekid je, po pravilu, resetovanje ovog tajmera upisom odgovarajuće šifre. Ukoliko mikrokontroler iz nekog razloga ne reaguje na ovaj prekid sistem se automatski resetuje. Ideja je da procesor u određenim vremenskim intervalima resetovanjem ovog tajmera javlja da je sve u redu. Neresetovanje ovog tajmera nije potreban uslov uslov da se zaključi da je došlo do greške, ali uglavnom jeste dovoljan. Ovaj tajmer se isključuje kada se debaguje program, a može se i koristiti kao običan tajmer.

- Ovaj mikrokontroler ima 2 **tajmera** od kojih svaki ima po 3 *capture/control* registara. Ovi tajmeri mogu da koriste sva tri izvora takta. Moguća su dva režima rada – brojački i merenje vremena koje protekne između događaja. U brojačkom režimu rada tajmer broji do određene vrednosti i potom generiše prekid. U *capture* režimu rada se bira događaj koji se meri (uzlazna i/ili silazna ivica signala) i kada se ovaj događaj desi u registar se upisuje trenutna vrednost tajmera i generiše prekid po potrebi. Tajmeri ove porodice mikrokontrolera imaju sjajnu podršku za impulsno-širinsku modulaciju kao vid DA konverzije. Postoje pinovi koji su izlazi tajmera čija logička vrednost može

da se programira u zavisnosti od trenutne vrednosti brojača; tako se, na primer, može obezbediti da je izlaz na visokom, odnosno niskom nivou uvek kada je vrednost brojača veća od neke definisane.

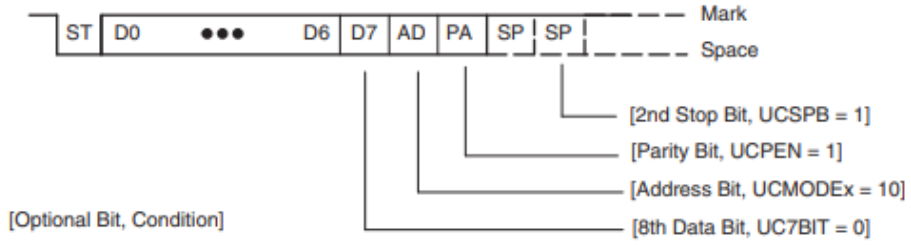
- Ovaj mikrokontroler poseduje jedan **množač**. Ova porodica mikrokontrolera spada u grupu mikrokontrolera koji su specijalizovani za digitalnu obradu signala i množenje je ovde jedna od najčešćih operacija koje su skupe u smislu vremena izvršavanja. Množač mikrokontrolera MSP430i2041 podržava množenje pozitivnih brojeva, kao i brojeva u komplementu dvojke, pri tome operandi mogu biti širine 8 ili 16 bita. Podržana je i MAC (eng. *Multiply and Accumulate*) operacija, brojeva sa i bez znaka.

- Modul **eUSCI_A0** podržava serijsku komunikaciju, i to UART (eng. *universal asynchronous receiver-transmitter*), IrDA (eng. *Infrared Data Association*) i SPI (eng. *Serial Peripheral Interface*). UART i IrDA su vrste asinhrone serijske komunikacije, a je SPI sinhrona. Sva tri protokola imaju posebne žice za prijem i slanje podataka. Kod SPI protokola postoji i žica kojom se obezbeđuje isti takt na obe strane. Bazirana je na *master-slave* principu i posebnom žicom se selektuje odgovarajući slave uređaj.

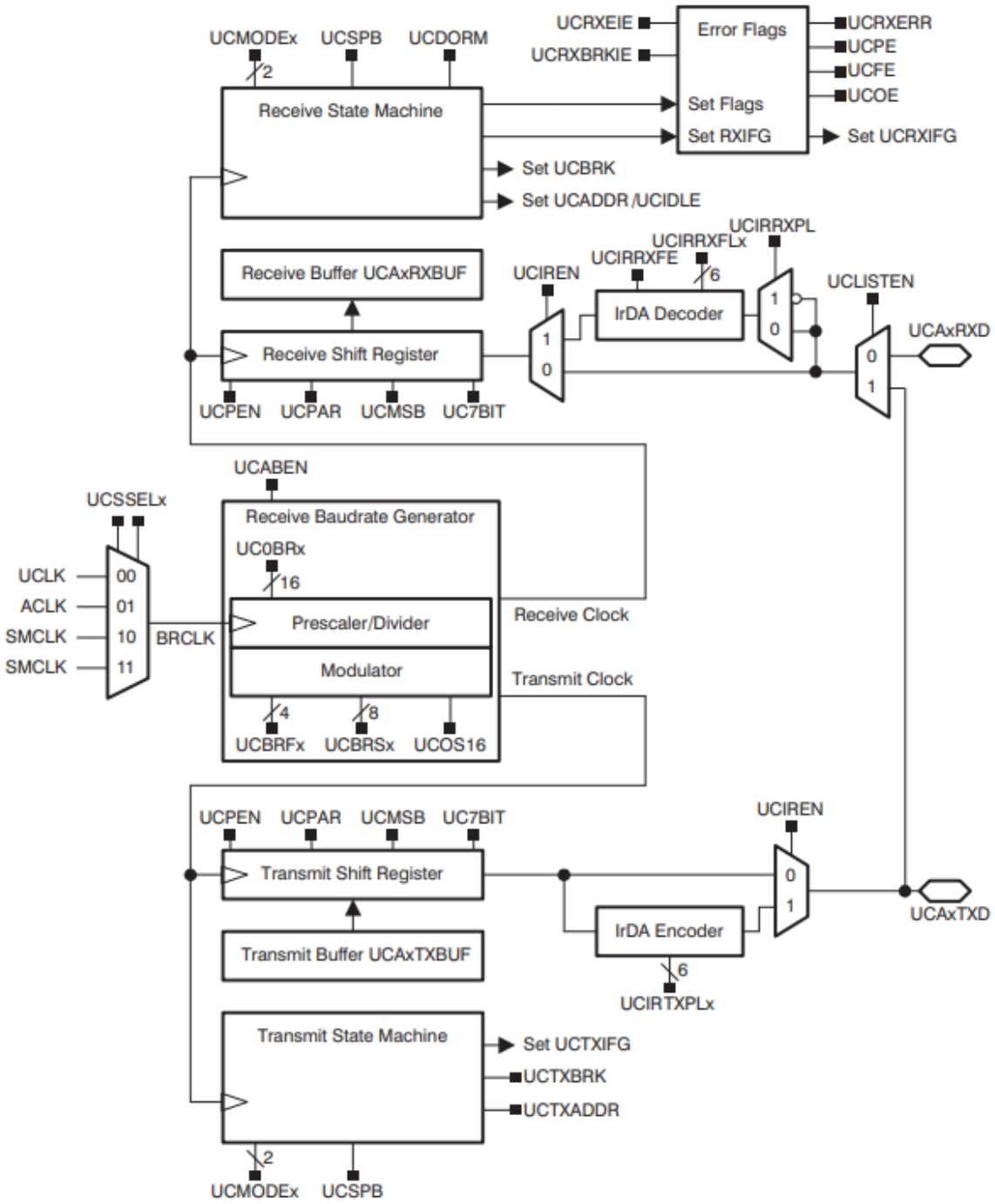
- Modul **eUSCI_B0** podržava SPI i I2C serijsku komunikaciju. *Inter-IC bus* (IIC ili I2C) je sinhrona magistrala koja funkcioniše na *master-slave* principu. Ona koristi dve *single-ended* linije SCL (*Serial Clock Line*) i SDA (*Serial Data Line*) za polu-dupleks komunikaciju. Protokol je razvijen od strane *Philips*-a i široko je rasprostranjen za komunikaciju na maloj daljini između jednog ili više kontrolera i periferijskih uređaja.

2.1. UART

Periferija eUSCI_A0 podržava UART komunikaciju koja je korišćena u projektu za slanje rezultata konverzije na računar. Na slici 2.1.1. je prikazan uopšteni format UART poruke, a na slici 2.1.2. je prikazan izgled ove periferije kada je konfigurisana da radi kao UART.



Slika 2.1.1. *Format UART poruke u MSP430i2041 mikrokontroleru* (Texas Instruments, 2014)



Slika 2.1.2. *Blok šema eUSCI_Ax periferije u UART režimu rada* (Texas Instruments, 2014)

2.1.1. Prijemnik

Gornja trećina slike 2.1.2. prikazuje prijemnik UART-a. Komponente koje čine prijemnik su mašina stanja koja ga kontroliše (eng. *Receive State Machine*), logika za kontrolu flegova grešaka (eng. *Error Flags*), registar u koji se smešta primljeni podatak (eng. *Receive Buffer* – UCAxRXBUF) i pomerački registar u koji se smeštaju podaci po prijemu (eng. *Receive Shift Register*) zajedno sa multiplekserima i IrDA dekoderom koji ga kontrolišu. Konačno, iz dela zaduženog za generisanje takta se prosleđuje takt odgovarajuće frekvencije za prijemnik (eng. *Receive Clock*).

Mašina stanja je kontrolisana sa tri signala – UCMODEx (dvobitni), UCSPB i UCDORM. Signal UCMODEx služi da se izabere asinhroni režim rada prijemnika kada je UCSYNC postavljen na 0. Dostupni su sledeći režimi rada: klasičan UART režim (00b), *idle-line* multiprocesor režim (01b), multiprocesor režim sa adresnim bitom (10b) i UART režim rada sa automatskom detekcijom *baud rate*-a (11b). Signal UCSPB se koristi da se izabere jedan ili dva stop bita (0b i 1b, respektivno). Signal UCDORM bira da li je prijemnik u *sleep* režimu rada (1b) ili ne (0b). Kada prijemnik nije u *sleep* režimu svaka primljena proruka generiše prekid koji obaveštava procesor o primljenoj poruci (i eventualno ga izbacuje iz nekog *low power* režima). Ukoliko je ovaj bit setovan samo karakteri kojima prethodi visoka vrednost napona na liniji (*idle-line*) ili oni gde je adresa odgovarajuća mogu izazvati prekid. Ukoliko je u pitanju režim u kome se automatski detektuje *baud rate* ovaj prekid se aktivira ukoliko je primljeni podatak bio ispraćen odgovarajućom pauzom i sinhronizacijom.

Mašina stanja je zadužena za postavljanje odgovarajućih flegova. Signal ICBRK detektuje pauzu koja je definisana nulama. Pauza se koristi kod automatske detekcije *baud rate*-a, ali je moguće iskoristiti ovaj signal, setovanjem signala ICBRKIE, tako da se generiše prekid kada su primljene sve nule (bez obzira na to šta predstavljaju dati podaci, bilo konkretan podatak, bilo bitove parnosti ili adrese). U režimu rada sa adresnim bitom se šalju poruke koje mogu biti adresa ili podatak. Tada svaka poruka osim podatka koji šalje ima jedan rezervisan bit koji govori i tome da li data poruka predstavlja adresu. Kada je ovaj bit setovan fleg UCADDR je setovan, što označava da je vrednost koja se nalazi u RX registru adresa. Kada se koristi *idle-line* režim ovaj fleg predstavlja UCIDLE signal koji se aktivira nakon što se detektuje 10 jedinica na magistrali označavajući da je linija u *idle* stanju, odnosno na visokom naponskom niovu.

Signal UCRXEIE se koristi da se izabere da li će prijemna poruka biti smeštena u RX registar ukoliko je došlo do greške. Kada je ovaj signal resetovan ukoliko je detektovana greška u stop bitu (eng. *framing error*) ili u bitu parnosti, primljena poruka neće biti upisana u registar. Kada je ovaj signal na jedinici, podatak se upisuje u registar i odgovarajući bit greške je setovan.

Signali koji označavaju da je došlo do greške su UCRXERR, UCPE, UCFE, i UCOE. Pri tome je signal UCRXERR setovan ukoliko je bilo koji od sledeća tri aktivan, odnosno, ovaj signal govori o tome da je došlo do greške uopšte. Signal UCPE je signal koji označava grešku parnosti, odnosno, broj jednica je paran onda kada treba da bude neparan i obrnuto, što znači da je došlo do neparnog broja grešaka u toku prenosa. Signal UCFE označava grešku na stop bitu. Ova greška se javlja kada se detektuje nizak nivo stop bita. U slučaju da se koriste dva stop bita dovoljno je da dođe do greške na jednom da bi ovaj fleg bio setovan. Signal UCOE označava da je u RX bafer upisana nova poruka iako prethodna nije bila očitana. Svi ovi flegovi su automatski obrisani kada se pročita podatak iz RX registra, ali se mogu i softverski obrisati, s tim da se ne preporučuje softversko brisanje UCOE flega jer se u tom slučaju ne garantuje ispravan rad. Prema tome, neophodno je proveravanje grešaka pre očitavanja registra.

Signali koji kontrolišu pomerački registar su signal UCPEN, UCPAR, UCMSB i UC7BIT. Signal UCPEN označava da li se koristi bit parnosti, dok signal UCPAR govori o tome da li je u pitanju bit parne parnosti (1b) ili neparne parnosti (0b). Signal UCMSB se setuje kada se očekuje da je prvi pristigao bit bit najveće težine, dok kada je resetovan, to znači da se bitovi šalju počev od bita najmanje težine. Na osnovu ove informacije se na odgovarajući način primljena poruka smešta u RX registar. Konačno, signal UC7BIT označava dužinu poruke. Podrazumevana dužina (0b) je 8-bitna, a setovanjem ovog signala se dužina poruke postavlja na 7 bita.

Na slici je takođe prikazano da je moguće propustiti signal sa predajnika i vratiti ga prijemniku aktiviranjem signala UCLISTEN. Kao što se vidi na blok dijagramu, signal koji se prima se tretira na isti način bez obzira na to da li je stigao sa strane ili od istog mikrokontrolera. Signal UCIREN se koristi da bi se propustio rezultat IrDA dekodera.

2.1.2. Generator takta

Signalom UCSSELx je moguće izabrati taktni signal koji će se koristiti u UART-u. Moguće je koristiti jedan od tri taktna izvora koji postoje u samom mikrokontroleru ACLK, SMCLK i MCLK ili neki spoljašnji izvor takta UCLK. U zavisnosti od *baud rate*-a koji se želi postići i frekvencije takta, signali UC0BRx, UCBRFx, UCBRSx i UCOS16 se biraju prema sledećem algoritmu.

1. $N = \frac{f_{BRCLK}}{baud\ rate}$
2. Ako je N veće od 16 prelazi se na korak 4, u suprotnom se prelazi na korak 3.
3. OS16 = 0, UCBRx = INT(N), nastavlja se sa korakom 5
4. OS16 = 1, UCBRx = INT(N/16), UCBRFx = INT([(N/16) – INT(N/16)] × 16)

5. UCBRSx se određuje na osnovu decimalnog dela rezultata N - INT(N) iz tabele date u *User Guide*-u.

Što je frekvencija BRCLK veća u odnosu na *baud rate*, to je verovatnoća greške manja.

2.1.3. Predajnik

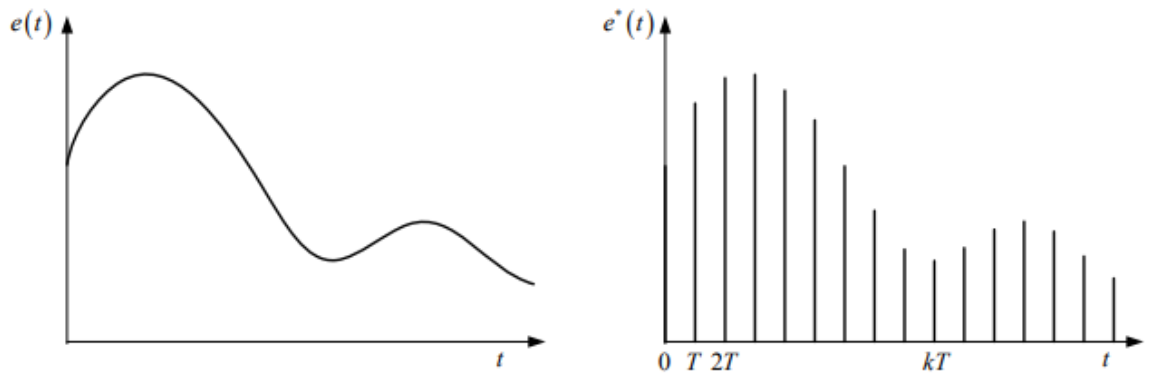
Na donjem delu slike 1.1.2. je prikazan UART predajnik. On se sastoji iz pomeračkog registra, bafera u koji se smešta podatak koji se šalje (UCAxTXBUF), mašine stanja predajnika i IrDA enkodera. Signali koji postoje i kod prijemnika imaju istu funkciju i kod predajnika. Signal UCTXIFG je signal prekida od strane prijemnika i označava da je prijemnik spreman da primi novu poruku koju treba upisati u UCAxTXBUF. UCTXBRK je signal koji služi da se generiše pauza u *idle-line* multiprocesorskom režimu rada koja označava kraj skupa poruka, dok se signal UCTXADDR označava da je sledeći karakter koji se upiše u TX bafer potrebno poslati nakon pauze koja traje bar 11 karaktera.

3. Diskretizacija signala

Da bi rad sa podacima u digitalnim sistemima bio moguć neophodno je diskretizovati date signale u vremenu i po amplitudi. Ovu obradu vrši AD konvertor, pri čemu se najčešće signal prvo odabira u vremenu, a potom vrši kvantovanje amplitude. U ovom poglavlju će biti razmatrano odabiranje signala u vremenu, kvantizacija amplitude i njihovi efekti. Osim toga, biće prikazani i različiti brojni sistemi koji se koriste i razmotrene njihove osobine.

3.1. Odabiranje signala

Posmatrajmo kontinualni signal koji se diskretizuje u vremenu, tako što se u ekvidistantnim vremenskim trenucima uzima njegova vrednost. Ovo je prikazano na slici 3.1.1. gde je $x(t)$ kontinualni signal, a $x^*(t)$ isti taj signal diskretizovan u vremenu sa periodom odabiranja T.



Slika 3.1.1. *Kontinualni signal i signal dobijen diskretizacijom sa periodom odbiranja T*

Između ova dva signala postoji sledeća veza:

$$x^*(t) = \begin{cases} x(t); t = kT, k = 0, 1, 2, \dots \\ 0; \qquad inače \end{cases} \tag{3.1.1}$$

Iako je ovde prikazan signal koji je kauzalan, izvođenje se može proširiti i na signale koji nisu kauzalni. Uvedimo signal $i(t)$ koji predstavlja beskonačnu povorku Dirakovih impulsa:

$$i(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \tag{3.1.2}$$

Tada se signal $x^*(t)$ može napisati kao proizvod kontinualnog signala $x(t)$ i signala $i(t)$[1]:

$$x^*(t) = x(t)i(t) \tag{3.1.3}$$

Kako množenje u vremenskom domenu odgovara konvoluciji u frekvencijskom domenu, Furijeova transformacija dobijenog signala se može pisati kao:

$$X^*(j\omega) = \frac{1}{2\pi} X(j\omega) * I(j\omega) \tag{3.1.4}$$

Odredimo Furijeovu transformaciju signala $i(t)$. Ovaj signal je očigledno periodičan, pa se može razviti u Furijeov red.

$$i(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} \tag{3.1.5}$$

gde je :

$$\omega_0 = \frac{2\pi}{T}; a_k = \int_{-\frac{T}{2}}^{\frac{T}{2}} i(t)dt = \frac{1}{T}; k = 0, \pm 1, \pm 2, \dots \tag{3.1.6}$$

Prema tome, signal $i(t)$ je:

$$i(t) = \frac{1}{T} \sum_{k=-\infty}^{\infty} e^{jk\omega_0 t} \tag{3.1.7}$$

Furijeova transformacija datog signala se dobija korišćenjem veze između Furijeovog reda i Furijeove transformacije:

$$F(j\omega) = \sum_{k=-\infty}^{\infty} 2\pi a_k \delta(\omega - k\omega_0) \tag{3.1.8}$$

Na osnovu ove veze tražena Furijeova transformacija je:

$$I(j\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_0) \tag{3.1.9}$$

[1] Do ovakvog modelovanja odabiranja se dolazi jer je neophodno održati energiju signala. Kako se vreme uzimanja uzorka skraćuje energija signala teži nuli. Ako se pretpostavi da površina ispod svakog pravougaonog impulsa ostaje jednaka jedinici kada vreme zadrške teži nuli, pravougaoni impulsi postaju Dirakovi delta impulsi, a energija signala ostaje ista.

Furijeova transformacija diskretizovanog signla je:

$$X^*(j\omega) = \frac{1}{2\pi} X(j\omega) * I(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\lambda) I\big(j(\omega - \lambda)\big) d\lambda \tag{3.1.20}$$

Zamenom prethodno dobijene vrednosti $I(j\omega)$ u gornji izraz dobijamo:

$$X^*(j\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\lambda) \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\omega - \lambda - k\omega_0)\, d\lambda \tag{3.1.21}$$

Ukoliko integral i suma zamene mesta dobijamo:

$$X^*(j\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} X(j\lambda) \delta(\omega - \lambda - k\omega_0) d\lambda \tag{3.1.22}$$

Važi:

$$\int_{-\infty}^{\infty} f(t)\delta(t - T) = f(T); \quad \delta(-x) = \delta(x) \tag{3.1.23}$$

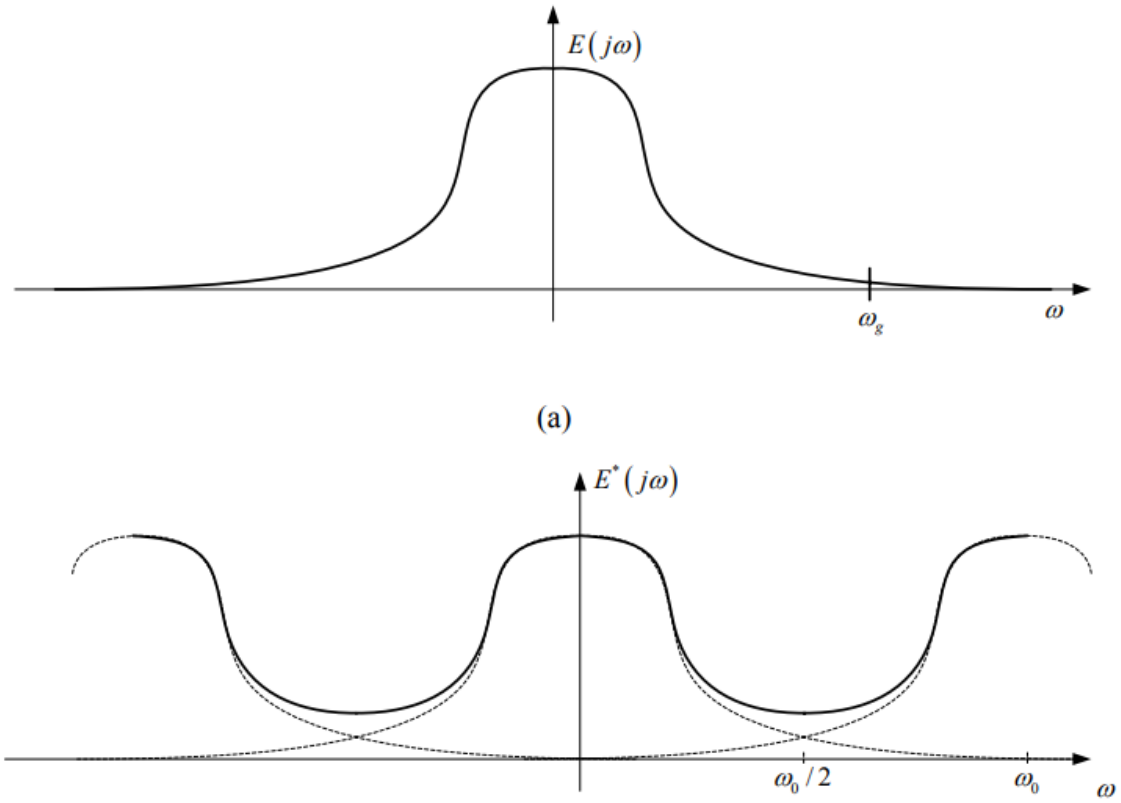
Korišćenjem ovih osobina izraz (4.1.22) se dalje transformiše u:

$$X^*(j\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X\big(j(\omega - k\omega_0)\big) \tag{3.1.24}$$

Ovaj rezultat je jako važan jer pokazuje da je spektar diskretizovanog signala sastavljen od sume spektara originalnog signala pomerenih za učestanost $k\omega_0$, gde je k ceo broj. Na slici 3.1.2. je prikazan spektar originalnog signala i spektar signala dobijenog njegovom diskretizacijom, pri tome je sa ω_g predstvljena granična učestanost kontinualnog signala, odnosno ona učestanost iza koje snaga signala postaje zanemarljiva.

Sa slike se jasno vidi da, ukoliko želimo da spektar dobijenog signala bude što verodostojniji originalnom spektru, ne treba dodavati komponente koje nemaju zanemarljivu snagu, odnosno treba da važi:

$$\omega_g < \omega_0 - \omega_g \implies \omega_0 > 2\omega_g \tag{3.1.25}$$



Slika 3.1.2. *Spektar kontinualnog signala i spektar diskretizovanog signala*

Izraz 3.1.25 se može napisati i kao:

$$f_0 > 2f_g \tag{3.1.26}$$

Dobijeni rezultat je u literaturi poznat kao Šenonova teorema, ili teorema odabiranja. Učestanost $2f_g$, koja je najniža učestanost kojom se signal može odabirati bez gubitka informacija, se naziva Nikvistova učestanost[2].

Ukoliko ovaj uslov nije zadovoljen dolazi do preklapanja spektra (eng. *aliasing effect*). Tada spektar diskretizovanog signala više ne odgovara originalnom i informacije o originalnom signalu se gube.

Teorema odabiranja u nekim slučajevima može biti suviše stroga. U izvođenju je pretpostavljeno da signal izgleda kao da je filtriran NF filtrom. Ukoliko signal ima nenulti spektar u opsegu učestanosti $[\omega_1, \omega_2]$,

[2] U literaturi se može naći i definicija Nikvistove učestanosti kao učestanosti koja je jednaka graničnoj učestanosti signala.

odnosno, ukoliko se radi o uskopojasnom signalu, može se pokazati da je najmanja učestanost odabiranja data izrazom:

$$\omega_0 > \frac{2\omega_2}{[\omega_2/(\omega_2 - \omega_1)]} \tag{3.1.27}$$

gde je uglastim zagradama ozmačena operacija ceo deo.

Konačno, potrebno je uspostaviti vezu između spektra diskretnog i kontinualnog signala. Vremenski kontinualni signal $x(t)$ se pretvara u niz odbiraka $x[k]$, tako da važi $x[k] = x(kT)$. Signal $x^*(t)$ je dat formulom 3.1.3., i zamenom signala $i(t)$ dobija se:

$$x^*(t) = x(t)\sum_{k=-\infty}^{\infty}\delta(t - kT) = \sum_{k=-\infty}^{\infty}x(kT)\delta(t - kT) \tag{3.1.28}$$

Sa jedne strane imamo Furijeovu transformaciju diskretnog signala $x[k]$ datu izrazom:

$$X(j\Omega) = \sum_{k=-\infty}^{\infty}x[k]e^{-j\Omega k} \tag{3.1.29}$$

Sa druge strane, Furijeova transformacija izraza 4.1.28 daje rezultat:

$$X^*(j\omega) = \int_{-\infty}^{\infty}\left[\sum_{k=-\infty}^{\infty}x(kT)\delta(t - kT)\right]e^{-j\omega t}dt = \sum_{k=-\infty}^{\infty}x(kT)\int_{-\infty}^{\infty}\delta(t - kT)e^{-j\omega t}dt \tag{3.1.30}$$

$$X^*(j\omega) = \sum_{k=-\infty}^{\infty}x(kT)\,e^{-j\omega kT} = \sum_{k=-\infty}^{\infty}x[k]\,e^{-j\omega kT} \tag{3.1.31}$$

Poređenjem izraza 4.1.29 i 4.1.31 nalazimo vezu između spektra diskretnog signala i signala dobijenog impulsnim odabiranjem:

$$\Omega = \omega T = \frac{\omega}{f_0} = 2\pi\frac{f}{f_0} = 2\pi F \tag{3.1.32}$$

3.2. Realno odabiranje signala i diskretna Furijeova transformacija

Da bi se signal mogao predstaviti na računaru neophodno je izvršiti pre svega njegovo odabiranje u vremenu. U praktičnim realizacijama, uzorci signala se uzimaju u vrlo kratkim ali konačnim intervalima vremena. To, naravno, dovodi do modifikacije spektra u odnosu na slučaj odabiranja povorkom Dirakovih impulsa, ali taj uticaj je skoro neprimetan i može se zanemariti. Tako da je i u praktičnim primenama, na prvi pogled, ukoliko je učestanost signala ograničena i frekvencija odabiranja veća od Nikvistove

učestanosti, spektar signala gotovo nepromenjen u opsegu od značaja, odnosno, teorija dozvoljava skoro egzaktnu rekosnstrukciju signala. U realnosti, potpuno tačna rekonstrukcija nije moguća obzirom da zahteva idealni filtar koji nije kauzalan, a samim tim nije ni fizički ostvariv.

Drugo ograničenje koje se javlja u digitalnoj obradi signala je činjenica da on treba da bude smešten u memoriju, što sa sobom povlači ograničen broj odbiraka signala. Dakle, da bi postojala teoretska mogućnost rekonstrukcije signala, on mora biti ograničen u frekvencijskom, ali i u vremenskom domenu.

Neka je dat signal $x(t)$ takav da ima nenulte vrednosti u intervalu $t_1 < t < t_2$. Vrednost signala se neće promeniti ako se signal pomnoži pravougaonom funkcijom koja je jednaka jedinici na istom intervalu. Prema tome, može se pisati:

$$x(t) = x(t)\,\mathrm{rect}\left(\frac{t - t_0}{\Delta t}\right) \tag{3.2.1}$$

gde je $t_0 = (t_1 + t_2)/2$ i $\Delta t = t_2 - t_1$.

Tada je Furijeova transformacija signala $x(t)$:

$$X(j\omega) = X(j\omega) * \Delta t\,\mathrm{Sinc}\left(\frac{\omega \Delta t}{2\pi}\right) e^{-j\omega t_0} \tag{3.2.2.}$$

Kako je *sinc* funkcija frekvencijski neograničena konvolucija sa njom je takođe neograničena. Prema tome, gornji izraz je tačan jedino ako je i spektar signala $x(t)$ neograničen. Odavde zaključujemo da signal ne može biti ograničen i u vremenu i po frekvenciji.

Na osnovu ovoga se vidi da se signal u opštem slučaju ipak ne može potpuno verno prikazati na računaru.

Ukoliko je potrebno prikazati spektar signala na računaru, tada se ponovo pojavljuje ograničenje u smislu nemogućnosti prikazivanja kontinualnih vrednosti. Računanje Furijeove transformacije nije moguće, pa se ona modifikuje, odnosno vrši se odabiranje u spektralnom domenu, i koristi se diskretna Furijeova transformacija (eng. *Fast Fourier Transfrom – FFT*). Diskretna Furijeova transformacija radi sa signalima koji su ograničeni u vremenu tako što podrazumeva da se data sekvenca periodično ponavlja.

Diskrena Furijeova transformacija je data izrazom:

$$X[k] = \sum_{i=0}^{N-1} x[i] e^{-\frac{j2\pi ki}{N}}, k = 0, 1, \dots, N-1 \tag{3.2.3}$$

gde je N broj broj odbiraka u frekvencijskom domenu.

Inverzna Furijeova transformacija se računa po formuli:

$$x[i] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi ki}{N}}, i = 0, 1, \dots, N-1 \quad (3.2.4)$$

Rezolucija Diskretne Furijeove transformacije, odnosno inkrement učestanosti za koji se ona računa, je $\Delta F = F_S/N$.

3.3. Predstavljanje brojeva u binarnom sistemu

U gotovo svim savremenim računarskim sistemima se koristi binarna predstava brojeva. U zavisnosti od tačke koja deli ceo i razlomački deo broja razlikujemo sisteme sa fiksnom tačkom i sisteme sa pokretnom tačkom. U oviru ovih sistema postoje različite varijante predstavljanja brojeva koje se uglavnom razlikuju po tome kako predstavljaju negativne brojeve. Obzirom da se prilikom AD konverzije dobijaju celobrojni rezultati, u ovom odeljku će biti objašnjeno predstavljanje celih brojeva.

Najjednostavniji način prestavljanja pozitivnih brojeva u binarnom sistemu je takozvani prirodni binarni kôd. Prirodan broj N u takvom sistemu je dat formulom:

$$N = \sum_{i=0}^{M} b_i 2^i = b_M b_{M-1} \dots b_0 \quad (3.3.1)$$

gde je $b_i \in \{0,1\}$. Krajnji levi bit se naziva bit najveće težine (eng. *Most Significant Bit – MSB*), a krajnji desni bit najmanje težine (eng. *Least Significant Bit – LSB*).

Ukoliko je potrebno predstaviti i negativne, odnosno označene brojeve, to je moguće uraditi uvođenjem još jednog bita koji će predstavljati znak broja. Danas se koriste četiri koda za predstavljanje celih brojeva: znak plus amplituda, pomereni binarni kôd, komplement jedinice i komplement dvojke.

Sistem znak plus amplituda je najjednostavniji. U ovom sistemu krajnji desni bit predstavlja znak broja, a ostatak apsolutnu vrednost. U ovom sistemu postoje dve nule (100...00 i 00...00) i sabiranje nije jednostavno, pa se jako retko koristi u praksi.

Pomereni binarni kôd se dobija iz prirodnog binarnog koda tako što se najvećem pozitivnom broju pridružuje predstava koja se sastoji iz svih jedinica, a najnegativnijem predstava koja se sastoji iz svih nula. Ovaj kôd je ekvivaletan prirodnom binarnom kodu koji je sabran sa konstantom i na osnovu toga je i dobio ime. Mane ovog koda su komplikovano izvođenje računskih operacija kao i činjenica da je MSB kod negativnih brojeva jednak nuli što je u suprotnosti sa konvecijom. Međutim, ukoliko se koristi ovakva predstava to značajno pojednostavljuje izvođenje AD i DA konvertora.

Komplement jedinice podrazumeva da se pozitivni brojevi predstavljaju sa bitom najveće težine 1 kao kod znak plus amplituda koda, ali se negativni brojevi dobijaju komplementiranjem predstave njihove apsolutne

vrednosti. Mane ovog sistema su postojanje pozitivne i negativne nule kao i to što se prilikom sabiranja, ukoliko dođe do prenosa na MSB poziciji on mora sabrati sa LSB bitom.

Komplement dvojke je najčešće korišćeni sistem.U ovom sistemu, predstava pozitivnih brojeva je ista kao i u komplementu jedinice, ali se negativni brojevi dobijaju oduzimanjem pozitivnog od broja 2. Ova operacija je ekvivalentna komplementiranju pozitivnog broja i dodavanju jedinice dobijenom broju. Ovo značajno olakšava sabiranje i oduzimanje koje su i najčešće operacije. Oduzimanje se svodi na sabiranje sa komplementom kod koga je ulazni prenos jednak jedinici. Druga velika prednost, koja je od velikog značaja, ovog sistema je zatvorenost operacija sabiranja i oduzimanja, odnosno ukoliko je rezultat u opsegu koji se može predstaviti biće tačan čak i onda kada je došlo do prekoračenja u međurezultatima. Ova osobina se naziva i modularna aritmetika (eng. *modular arithmetics*, *wrap-around arthmetics*), i tu važi:

$$\mathrm{mod}(a \pm b) = \mathrm{mod}(a) \pm \mathrm{mod}(b) \tag{3.3.2}$$

Na kraju treba pomenuti još jednu dobru osobinu koda komplementa dvojke. Već je rečno da je pomereni binarni kôd veoma zgodan kada se koriste AD i DA konvertori. Veza između ovog koda i komplementa dvojke je veoma jednostavna. Naime, komplementiranjem bita najveće težine u pomerenom binarnom kodu se dobija broj u sistemu komplementa dvojke. Ova osobina značajno olakšava sintezu AD i DA konvertora.

3.4. Greške zbog odsecanja i zaokruživanja binarnih brojeva

Prilikom izvođenja računskih operacija u digitalnim sistemima često postoji potreba da se binarni broj skrati, odnosno predstavi manjim brojem bita. Ova potreba često postoji i kod sigma-delta AD konvertora, ali su razlozi za to detaljnije diskutovani u poglavlju 5. Ovu analizu je dovoljno uraditi za pozitivne brojeve jer se uvek može preći u pomereni binarni kôd i komplementiranjem dobiti broj u sistemu komplementa dvojke.

Binarni broj se može skratiti zaokruživanjem ili odsecanjem, odnosno odbacivanjem, bitova. Očigledno je da se u ovom slučaju dešava greška, ali osobine greške u ova dva postupka nisu iste. Neka je sa $Q(x)$ označena operacija kvantovanja signala x koja od signala predstavljenog sa L bitova daje signal gde je najnižih B bitova skraćeno.

Greška kvantovanja je tada:

$$\varepsilon = x - Q(x)2^B \tag{3.4.1}$$

Iz formule 3.4.1. se vidi da je širina greške 2^B[3].

Ukoliko se radi odsecanje (eng. *truncating*) to podrazumeva odbacivanje najnižih B bitova. Ukoliko se radi zaokruživanje (eng. *rounding*), tada se vrši odsecanje ukoliko je MSB koji se odbacuje 0, ondnosno, vrši se odsecanje i dodavanje jedinice na LSB koji ostaje ako je MSB koji se odbacuje 1.

Greška kvantovanja se najčešće modeluje kao beli šum, odnosno smatra se da ima uniformnu raspodelu. Širina te raspodele je ista i u slučaju odbacivanja i u slučaju zaokruživanja i data je formulom:

$$E = 2^B \tag{3.4.2}$$

Srednja vrednost greške zavisi od toga da li se bitovi odbacuju ili se radi zaokruživanje i data je izrazom:

$$\mu = \begin{cases} \frac{1}{2}E & odbacivanje \\ 0 & zaokruživanje \end{cases} \tag{3.4.3}$$

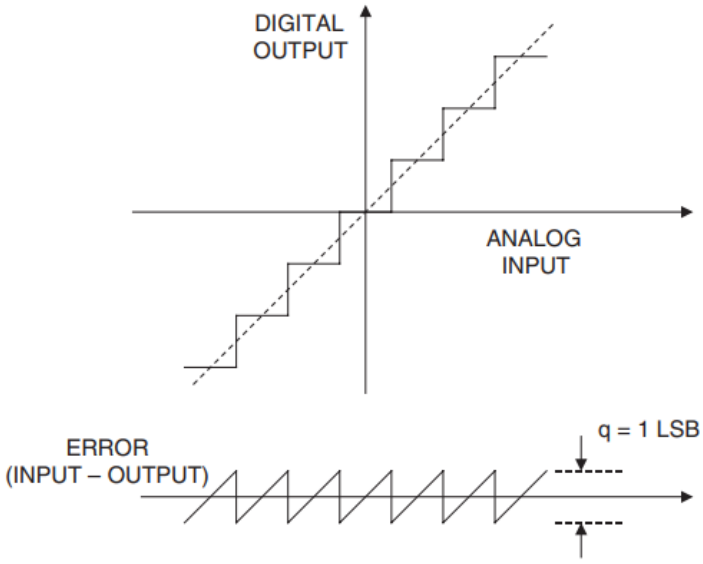
Varijansa je u oba slučaja ista i iznosi:

$$\sigma^2 = \frac{E^2}{12} \tag{3.4.5}$$

3.5. Kvantovanje ulaznog signala

U digitalnim sistemima je potrebno predstaviti signal konačnim brojem bita. Zbog toga se na ulazu sistema nalaze AD konvertori čija je uloga da analogni signal pretvore u digitalni. Osim prenosne funkcije AD konvertora koja utiče na frekvencijske karakteristike signala i vremenskog odabiranja koje se radi, na signal utiče i činjenica da se on mora predstaviti određenim brojem bita. Na ovaj način se javljaju greške kvantovanja ili kvantizacije koje je nemoguće izbeći, a zbog kojih se deo informacija o signalu nepovratno gubi. Kvantovanje signala se najčešće vrši postupkom zaokruživanja, odnosno analogna vrednost signala se pridružuje najbližem kvantizacionom nivou. Na slici 3.5.1. je prikazana kriva kvantovanja ulaznog signala, a ispod nje greška kvantovanja.

[3] U kontekstu celih brojeva ova greška nije veća od $2^B - 1$, ali u AD konverziji ovi celi brojevi predstavljaju realne brojeve pošto se rezultat skalira sa referentnim naponom, pa je gornja granica greške kao prilikom zaokruživanja realnih brojeva.



Sika 3.5.1. *Kvantovanje signala sa zaokruživanjem na najbližu vrednost i greška kvantovanja* (Analog Devices Inc., Engineeri, 2005)

Ukoliko se koristi N bitova, tada postoji 2^N kvantizacionih nivoa. Ukoliko je sa FSR označen opseg punog ulaznog napona (eng. *full scale range*) koji je $FSR = V_{max} - V_{min}$, tada je rezolucija kvantizatora data formulom:

$$LSB = q = \frac{FSR}{2^N} \tag{3.5.1}$$

Greška kvantovanja ε je u opsegu:

$$-\frac{q}{2} < \varepsilon < \frac{q}{2} \tag{3.5.2}$$

Odnos signala i šuma (eng. *signal noise ratio – SNR*) je metrika koja se najčešće koristi za prikazivanje uticaja greške kvantizacije. Prilikom analize greške kvantizacije, šum kvantizacije se najčešće modeluje kao beli aditivni šum, to jest, smatra se da je greška kvantizacije uniformno raspodeljena u opsegu datom formulom 3.5.2. Takođe, smatra se da je greška kvantizacije unifromno raspodeljena na celom spektru. Jako je bitno napomenuti da načini kojima se smanjuje šum kvantizacije, uključujući onaj koji se koristi kod sigma-delta AD konvertora koriste ovakav model šuma. Dakle, ukoliko model šuma, odnosno greške kvantizacije ne odgovara realnom stanju, ne može se očekivati smanjenje šuma korišćenjem ovih metoda.

Dalje izvođenje važi ukoliko greška kvantovanja može da se predstavi kao beli aditivni šum sa uniformnom raspodelom i ako ona nije korelisana sa sa signalom koji se konvertuje. Sukcesivne greške su korelisane ako je nivo signala mali. Takođe, ukoliko je korak kvantovanja preveliki, ne možemo govoriti o uniformnoj raspodeli greške – neophodno je da korak bude dovoljno mali da se sukcesivne aproksimacije razlikuju za nekoliko nivoa.

Odnos signal/šum se definiše kao:

$$SNR(\text{dB}) = 10 \log \frac{P_x}{P_n} \tag{3.5.3}$$

gde je P_x snaga ulaznog signala, a P_n snaga šuma.

Snaga slučajne promenljive čija je srednja vrednost nula je jedanaka njenoj varijansi. Takođe je poznato da je varijansa unifromne raspodele jednaka kvadratu širine raspodele podeljenom na 12. Korišćenjem ove činjenice nalazi se snaga šuma:

$$P_n = \sigma_\varepsilon^2 = \frac{q^2}{12} \tag{3.5.4}$$

Smatrajući da je signal normalizovan tako da je FSR jednako 1 i zamenom gore dobijene vrednosti u formulu 3.5.3 dobija se:

$$SNR(\text{dB}) = 10 \log P_x + 6.02\, N + 10.79 \tag{3.5.5}$$

Odavde se vidi da svaki bit koji se doda poboljšava odnos signal/šum za 6 dB. Ovaj odnos se takođe poboljšava za istu vrednost ukoliko se snaga signala poveća dva puta. Prema tome, što je amplituda signala veća (dokle god je u granicama AD konvertora), to je manji uticaj šuma kvantizacije. Zbog ovoga većina AD konvertora ima i diferencijalni pojačavač na svom ulazu koji će ulazni signal pojačati tako da se što bolje iskoristi opseg AD konvertora.

Ukoliko je na ulazu sinusoidalni signal koji zauzima ceo opseg kvantizatora njegova jednačina je:

$$x(t) = \frac{q2^N}{2} \sin(2\pi f t) \tag{3.5.6}$$

jer je $FSR = q2^N$.

U ovom slučaju se i dalje može smatrati da je šum uniformno raspodeljen, pa se za snagu šuma može koristiti jednačina 3.5.4. Poznato je da je snaga sinusoide jednaka polovini kvadrata amplitude, pa na osnovu toga dobijamo sledeći izraz za odnos signal/šum:

$$SNR(\text{dB}) = 10\log\frac{q^2 2^{2N}}{8} - 10\log\frac{q^2}{12} = 6.02\,N + 1.76\ \text{dB} \tag{3.5.7}$$

Iako je ova analiza dosta pojednostavljena, pokazuje se da je ona sasvim zadovoljavajuća u većini praktičnih slučajeva. Zbog toga se ova formula koristi za definisanje efektivnog broja bitova AD konvertora.

Realni AD konvertori pored šuma kvantizacije unose dodatne greške koji menja vrednost SNR-a. Ova realna vrednost se koristi da se definiše efektivni broj bitova na bazi formule 3.5.7. Efektivni broj bitova (eng. *effective number of bits – EONB*) je broj bitova koji se za dati odnos signal/šum uklapa u odnos signal/šuma idealnog kvantizatora:

$$ENOB = \frac{SNR - 1.76\ \text{dB}}{6.02\ \text{dB}} \tag{3.5.8}$$

3.6. Minimizacija greške kvantovanja

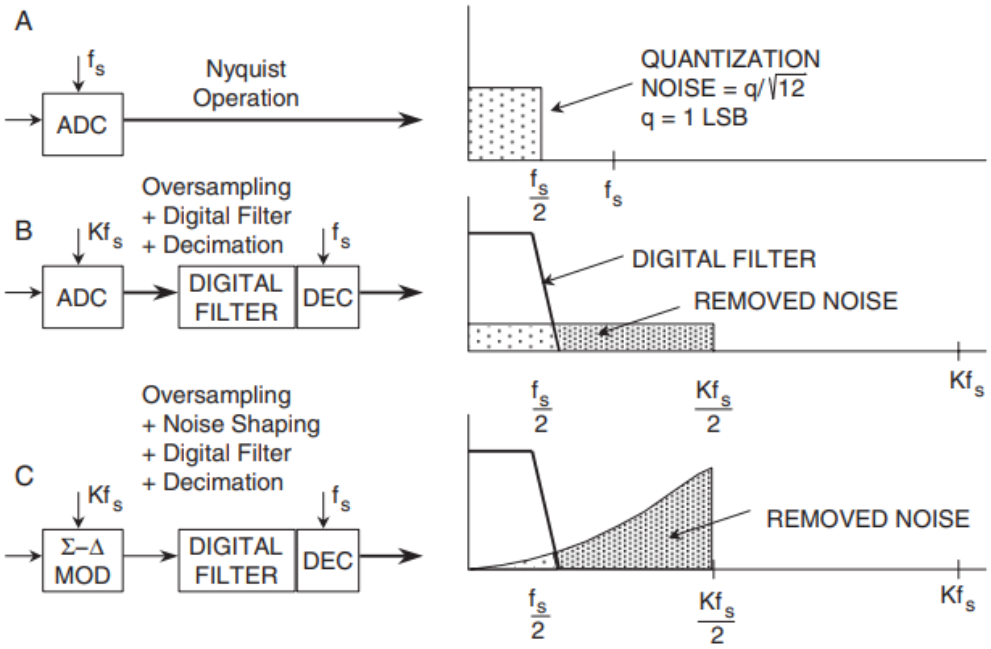
Iako se greška kvantizacije ne može izbeći ni u teoriji, njen uticaj se na nivou većeg broja odbiraka može popraviti. Postoje dva osnovna načina kojima se minimizuje greška kvantizacije. Ukoliko se smatra da je greška uniformno raspodeljena u spektru, tada se ona može minimizovati povećanjem frekvencije odabiranja signala. Drugi način za minimizaciju greške rešava problem greške kvantizacije koja se ne može modelovati belim šumom. Tada se na signal namerno dodaje šum (eng. *dithering*) čime se obezbeđuje da greška bude ravnomerno raspoređena.

Kao što je već objašnjeno u prethodnom odeljku, model greške u kome je ona ravnomerno raspoređena u Nikvistovom opsegu je sasvim zadovoljavajući u većini slučajeva. Ukoliko se frekvencija odabiranja poveća, tada se i opseg te greške povećava, a njena snaga na opsegu od interesa smanjuje. Zbog toga je odabiranje signala sa većom frekvencijom (eng. *oversampling*) osnovna ideja za minimizaciju greške kvantizacije. Ideja je prikazana na slici 3.6.1.

Ukoliko se signal sempluje sa učestanošću Kf_s i potom propusti kroz digitalni NF filtar efektivan broj bitova se povećava, odnosno smanjuje se uticaj šuma na signal, jer najveći deo šuma ostaje izvan opsega. Pošto je ukupna snaga šuma smanjena K puta, to je poboljšanje SNR-a dato formulom:

$$\Delta SNR = 10\ \log K\ \text{dB} \tag{3.6.1}$$

Za faktor $K = 4$ poboljšanje iznosi 6.02 dB, što odgovara jednom bitu više. Prema tome, da bi se dobilo poboljšanje od N bitova, potreban je faktor $K = 4^N$.



Slika 3.6.1. *Tehnike kojima se može minimizovati greška kvantizacije (gore: običan AD konvertor, sredina: AD konvertor sa većom učestanošću od Nikvistove, dole: AD konvertor sa oblikovanjem šuma)* (Analog Devices Inc., Engineeri, 2005)

Osim što se ovim zahteva značajno veća frekvencija odabiranja, treba uzeti u obzir da digitalni filtri koji rade sa ovakvim signalom treba da rade sa rečima velike širine pre nego što se dođe do konačnog rezultata. Zbog svega ovoga se ovaj jednostavan način redukovanja šuma kvantizacije ne koristi da se dobije značajno poboljšanje.

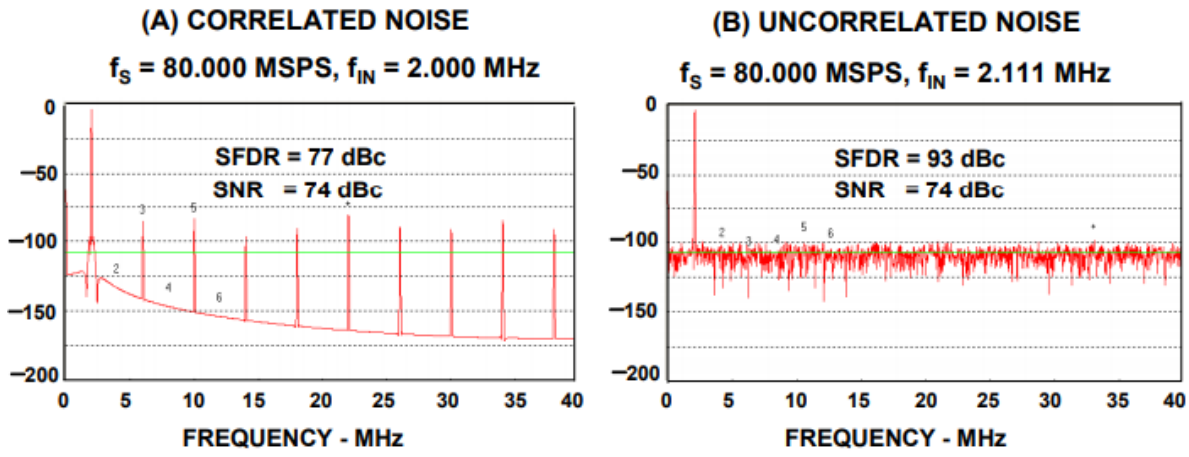
Značajnije poboljšanje bez preterano velike učestanosti odabiranja se koristi u sigma-delta AD konvertoru. Ovo poboljšanje se izvodi oblikovanjem šuma kvantizacije (eng. *noise shaping*). Ova tehnika "gura" šum kvantizacije ka većim učestanostima, pa je šum u opsegu od interesa još više oslabljen u odnosnu na obično korišćenje veće učestanosti odabiranja. Sama tehnika kojom se šum oblikuje je objašnjena u okviru poglavlja o sigma-delta AD konvertoru.

Kao što je već rečeno, greška kvantizacije ne mora uvek biti ravnomerno raspoređena u spektru. Na primer, za periodične signale koji imaju malu amplitudu ovo neće važiti. U ovakvim slučajevima se javlja velika korelacija između signala i šuma i tada je šum u spektru koncentrisan oko harmonika signala gde je i najnepoželjniji. Ovakvi problemi mogu da se reše namernim dodavanjem šuma koji dovodi do toga da šum ponovo bude slučajna promenljiva sa uniformnom raspodelom. U praktičnoj primeni i sam AD konvertor

unosi šum koji se upravo ovako ponaša, pa to ujedno i najčešće rešava ovaj problem kada se konvertuju signali kao što je, na primer, govor.

Takođe, ako je na ulazu AD konvertora sinusiodalni napon čija je frekvencija parni umnožak frekvencije odabiranja tada dolazi do velike korelacije između greške kvantizacije i signala, to jest, greška više nije uniformna i to se vidi kao pojava dodatnih harmonika u spektru. Objašnjenje ovoga je prilično intuitivno, naime signal se odabira uvek u istim vrednostima pa se prave jedne iste greške i one se prave periodično. Na slici 3.6.2. je prikazan spektar sinusoide koji se dobija kvanovanjem kada postoji korelacija šuma i ulaznog signala i spektar koji se dobija kada ista ne postoji.

Ovo su efekti koje treba imati na umu kada se radi sa periodičnim signalima i kada se testira sam AD konvertor, obzirom da je dovođenje sinusoidalnog signala jedan od osnovnih načina testiranja.



Slika 3.6.2. Spektar sinusoide nakon kvantizacije

A) Sinusoida čija je frekvencija parni umnožak frekvencije odabiranja kod koje je šum kvantizacije korelisan sa signalom

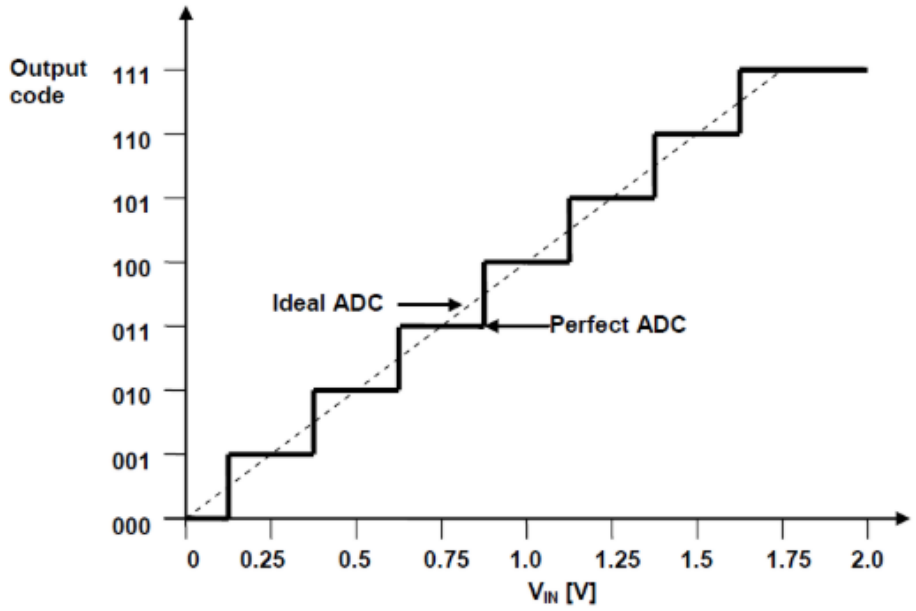
B) Sinusoida čija frekvencija nije celobrojni umnožak frekvencije odabiranja i ne postoji korelacija između šuma kvantizacije i signala (Kester, 2020)

Konačno, kada se konvertuje DC signal, greška se može smatrati uniformno raspodeljenom jedino onda kada je rezolucija AD konvertora dovoljno velika. Ukoliko to nije slučaj, čak i šum koji AD konvertor unosi neće biti dovoljan, naime, dati napon će se preslikavati uvek u istu vrednost, pa će srednja vrednost greške biti razlika između realne i kvantovane vrednosti.

4. Karakteristike A/D konvertora

A/D konvertor je komponenta koja analogni signal (najčešće napon) konvertuje u njemu odgovarajuću digitalnu reprezentaciju. Iako postoje različiti načini za realizovanje AD konvertora, osnovni prinicip je svuda isti – vrši se preslikavanje ulaznog analognog signala u određen broj bitova. Osim diskretizacije amplitude signala, implicitno se, prilikom AD konverzije vrši i diskretizacija signala u vremenu. Većina AD konvertora prvo diskretizuje signal u vremenu time što uzima vrednosti u određenim vremenskim trenucima, a potom se one diskretizuju po amplitudi.

Na slici 4.1. je prikazana prenosna karakteristika idealnog i savršenog AD konvertora.



Slika 4.1. *Prenosna karakteristika idealnog i savršenog 8-bitnog unipolarnog AD konvertora*
(Microchip Developer, 2020)

U prethodnom poglavlju su obrađene sve one greške koje su posledice toga što ne postoji tako nešto kao što je idealni AD konvertor i toga što signal menja svoju prirodu iz kontinualne u diskretnu. U ovom poglavlju će biti predstavljene sve one greške koje su posledica nemogućnosti realizacije savršenog AD konvertora. Postoje dve vrste ovih grešaka – statičke i dinamičke greške. Statičke greške se analiziraju na nivou DC signala, dok se dinamičke analiziraju u frekvencijskom domenu i odnose se na šum koji se u njemu pojavljuje.

Osnovne vrednosti kojima je definisan AD konvertor su napon pune skale:

$$FSR = V_{max} - V_{min} \tag{4.1.1}$$

gde su V_{max} i V_{min} najveći i najmanji napon koji se konvertuju.

Konvertor koji koristi N bitova ima 2^N kvantizacionih nivoa. Najmanja promena koja se može detektovati je rezolucija AD konvertora i to je ujedno vrednost bita najmanje težine:

$$LSB = \frac{FSR}{2^N} \tag{4.1.2}$$

4.1. Statičke karakteristike AD konvertora

Statičke greške su one koje utiču na tačnost konvertovanja DC signala. Postoje četiri vrste ovih grešaka i sve one se izražavaju u odnosu na naponsku rezoluciju AD konvertora - LSB. To su greška ofseta, greška pojačanja, integralna nelinearnost i diferencijalna nelinearnost. Ove karakteristike govore o tome koliko se dati AD konvertor razlikuje od savršenog AD konvertora.

4.1.1. Greška ofseta

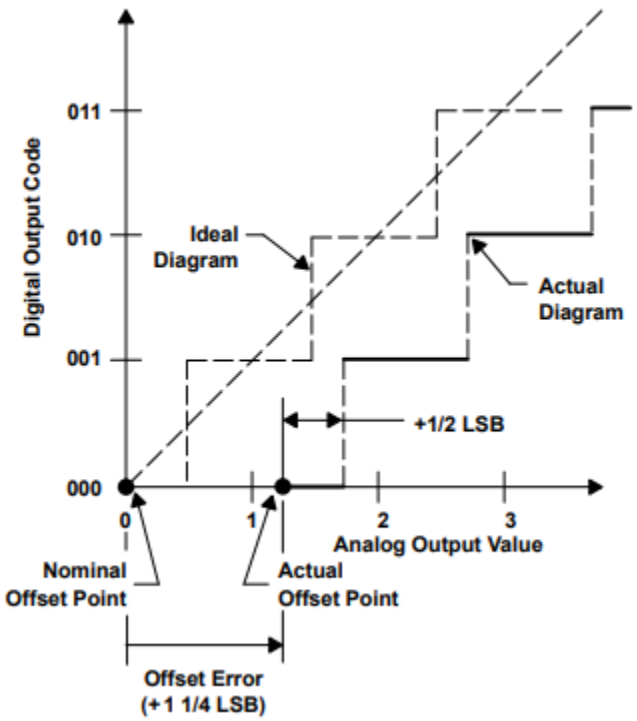
Greška ofseta se definiše kao pomeraj stvarne prenosne karakteristike AD konvertora u odnosu na savršenu prenosnu karakteristiku. Znak pomeraja ofseta zavisi od toga kako je proizvođač definisao ofset, *Texas Instruments*, na primer, ofset definiše kao vrednost u LSB za koju je karakteristika pomerena u odnosu na savršeni AD konvertor, gde je pomeraj udesno pozitivan ofset, a ulevo negativan.

Pomeranjem karakteristike udesno, to jest, uvođenjem pozitivnog ofseta, analogne vrednosti u opsegu $[V_{RefLO}, V_{RefLO} + |offset|]$ ne mogu da se iskoriste, odnosno da se razlikuju jer se preslikavaju u nulu. Sa druge strane, napon za koji se dobija najviši binarni kod se povećava za vrednost ofseta, pa je realni maksimalni napon koji se može konvertovati manji.

Pomeranjem karakteristike ulevo, odnosno uvođenjem negativnog ofseta, se dešava obrnuta situacija. Može se desiti da najniži binarni kodovi ne budu iskorišćeni, jer se ispostavi da su vrednosti napona koje im odgovaraju manje od minimalne ulazne.

Na slici 4.1.1.1. je prikazana pozitivna greška ofseta.

Ova greška se jednostavno rešava i u samom softveru dodavanjem ili oduzimanjem vrednosti ofseta.



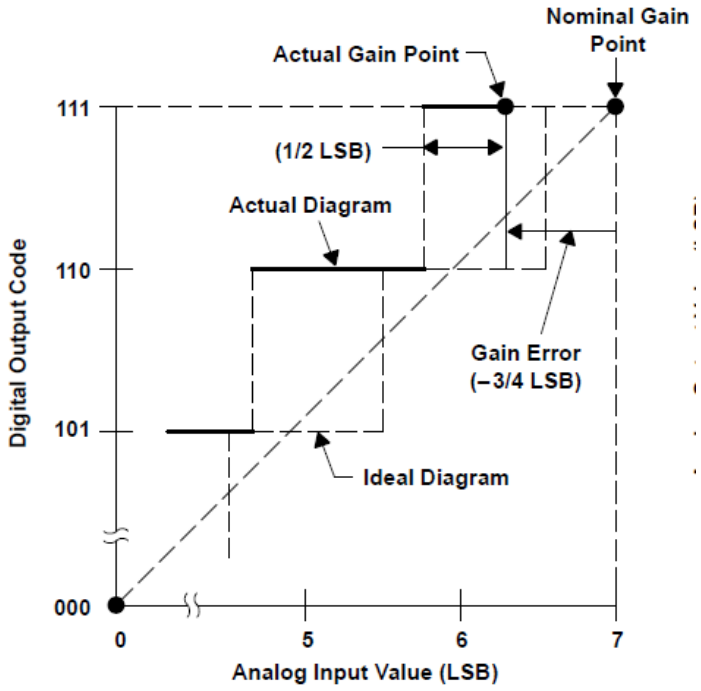
Slika 4.1.1.1. *Greška ofseta AD konvertora* (Texas Instruments, 1995)

4.1.2. Greška pojačanja

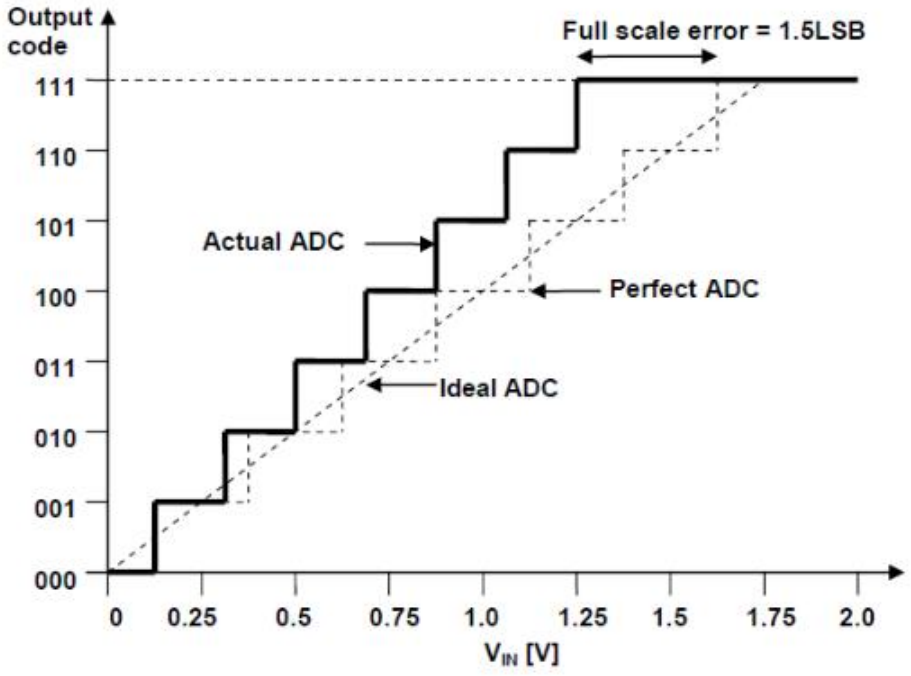
Greška pojačanja predstavlja razliku između sredine poslednjeg stepenika prenosne karakteristike realnog i savršenog AD konvertora (nakon kompenzovanja greške ofseta realnog konvertora). Linije koja spajaju sredine stepenika idealnog i realnog AD konvertora imaju određene nagibe, a ova greška govori o tome koliko se oni razlikuju.

Na slici 4.1.2.1. je prikazana ova greška.

Uvodi se još i greška pune skale (eng. *full scale error*) koja predstavlja razliku između vrednosti za koju izlaz AD konvertora dobija maksimalnu vrednost u slučaju savršenog i realnog konvertora. Ova greška je prikazana na slici 4.1.2.2.



Slika 4.1.2.1. *Pozitivna greška pojačanja* (Texas Instruments, 1995)



Slika 4.1.2.2. *Greška pune skale* (Microchip, 2020)

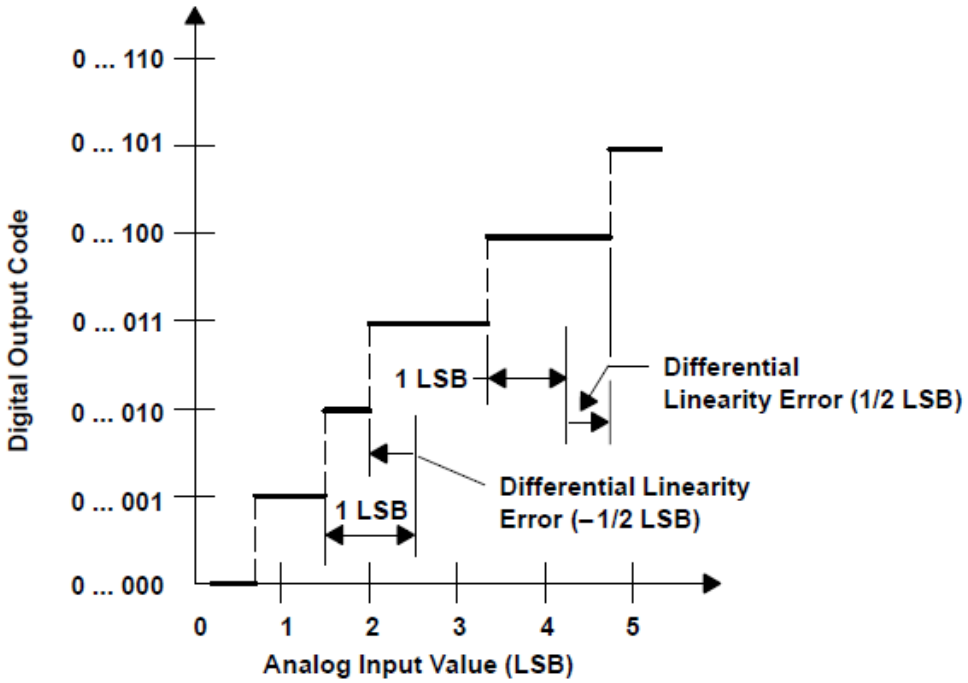
4.1.3. Diferencijalna nelinearnost

U slučaju idealnog AD konvertora širina svakog stepenika na prenosnoj karakteristici je LSB. Odstupanje širine stepenika od idealne je diferencijalna nelinearnost AD konvertora. Kada se govori o diferencijalnoj nelinearnosti posmatra se opseg u kome se ova greška nalazi.

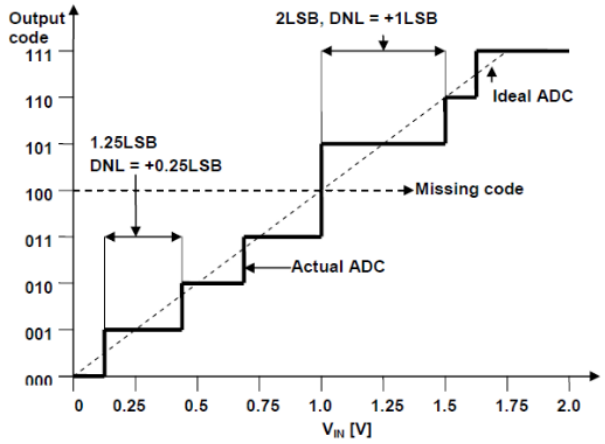
Na slici 4.1.3.1. je prikazana karakteristika realnog AD konvertora i na njoj je obeležena ova greška. Za konvertor dat na slici greška diferencijalne nelinearnosti je $\pm LSB/2$.

U vezi sa ovom greškom je i greška nedostajućeg koda. Naime, AD konvertor može imati karakteristiku u kojoj ne postoji ulazni napon za koji se dobija određeni kôd. Takav AD konvertor je prikazan na slici 4.1.3.2.

AD konvertor koji ima diferencijalnu grešku koja je manja ili jednaka težini najnižeg bita (LSB) sigurno koristi sve izlaze. Ukoliko postoji greška neodstajućeg koda, njena ozbiljnost je obrnuto srazmerna rezoluciji AD konvertora.



Slika 4.1.3.1. *Greška diferencijalne nelinearnosti* (Microchip, 2020)

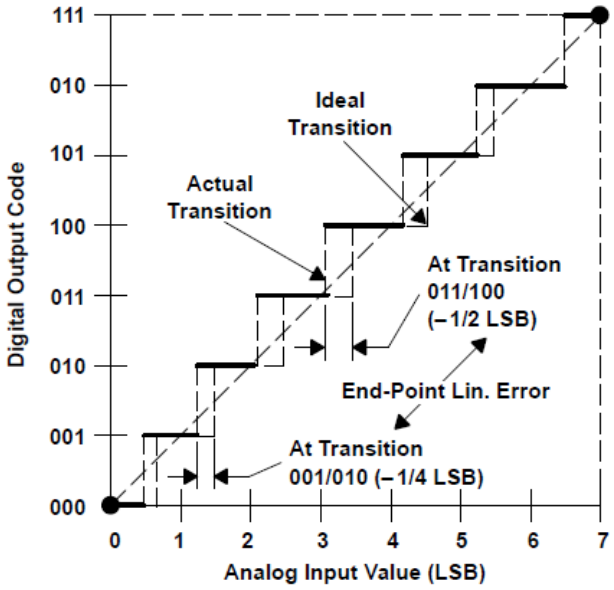


Slika 4.1.3.2. *AD konvertor sa nedostajućim kodom* (Microchip, 2020)

4.1.4. Integralna nelinearnost

Integralna nelinearnost se naziva i samo linearna greška i govori o tome koliko se prenosna karakteristika AD konvertora razlikuje od prave savršenog AD konvertora. Ova prava se formira nalaženjem linije koja sa minimalnom greškom aproksimira tačke na kojima se dešava promena koda (eng. *best fit*) ili se formira spajanjem krajnjih tačaka prenosne funkcije (kada su greške pojačanja i ofseta kompenzovane). Drugi način se najčešće koristi. Odstupanja od ove linije se mere u tačkama u kojima se dešava promena koda. Ova greška se zove integralna nelinearnost jer se greške na prethodnim nivoima gomilaju.

Na slici 4.1.4.1. je ilustrovana ova greška.

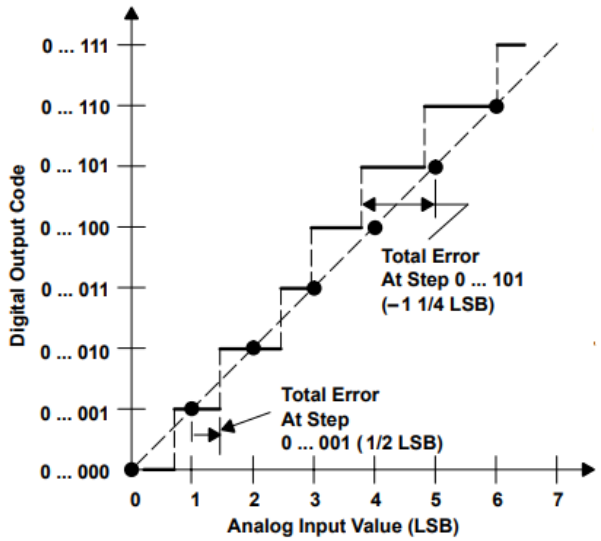


Slika 4.1.4.1. *Integralna nelinearnost AD konvertora* (Texas Instruments, 1995)

4.1.5. Ukupna greška AD konvertora

Ukupna greška AD konvertora je ukupna greška koja se javlja kao posledica svih prethodno navedenih statičkih grešaka. Ova greška govori o tome kako se AD konvertor ponaša u najgorem slučaju, to jest, ako se ne uvede kompenzacija za neke od grešaka.

Na slici 4.1.5.1. je prikazano kako se ova greška određuje.



Slika 4.3.5.1. *Ukupna greška AD konvertora* (Texas Instruments, 1995)

Na osnovu specifikacija AD konvertora je moguće odrediti maksimalnu moguću grešku koja može da se javi kada se vrši konverzija signala. Greška ofseta se jednostavno rešava u softveru dodavanjem, odnosno oduzimanjem konstante. Greška pune skale se takođe može jednostavno rešiti množenjem rezultata konverzije korekcijom. Ozbiljnost ovih grešaka takođe zavisi i od same primene AD konvertora. Tako, na primer, ukoliko je mereni signal deo povratne sprege, greška diferencijalne nelinearnosti ima mnogo veći značaj nego inače.

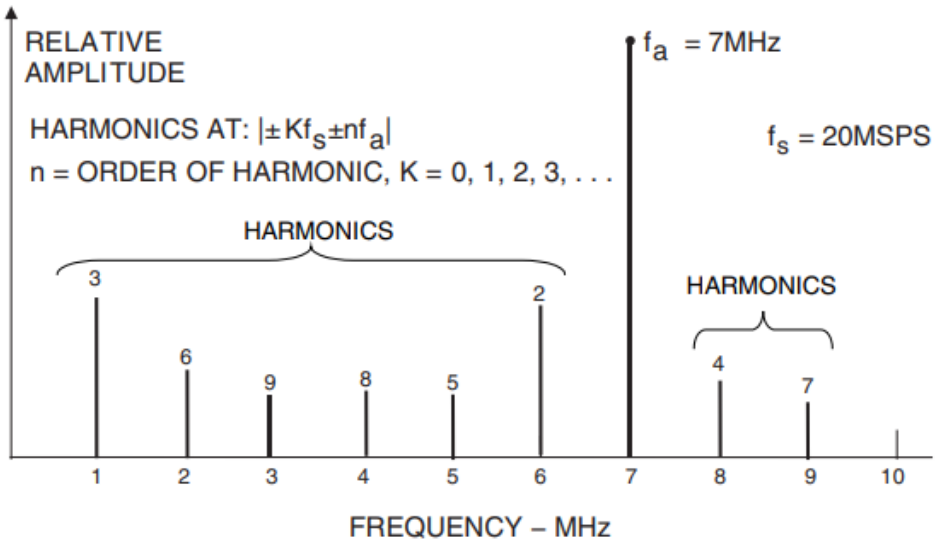
4.2. Dinamičke karakteristike AD konvertora

Dinamičke greške se mere u frekvencijskom domenu i odnose se na greške koje se javljaju kada je na ulazu AD konvertora AC signal. Za razliku od statičkih grešaka koje su jasno posledica realizacije samog AD konvertora i ne uključuju greške koje uvodi sam proces kvantizacije signala, kod dinamičkih grešaka to nije slučaj. Ove greške potiču i od kvantizacije, implementacije AD konvertora, ali i svih drugih izvora kao što je termički šum.

4.2.1. Harmonijska izobličenja

Harmonijska izobličenja (eng. *harmonic distortion*) su u opštem slučaju izobličenja koja nastaju na učestanostima koje su celobrojni umnožak frekvencije referentnog signala. Za merenje ovih izobličenja se najčešće na ulaz AD konvertora dovodi sinusoidalni signal i potom se primenom FFT-a posmatra šum koji se javlja na učestanostima koje su celobrojni umnožak frekvencije te sinusoide. Ova izobličenja se javljaju kao posledica nelinearnosti AD konvertora.

Na slici 4.2.1.1. su prikazane lokacije harmonika za signal učestanosti 7 MHz i frekvenciju odabiranja 20 MHz. Harmonici se nalaze na lokacijama $|\pm Kf_s \pm nf_a|$ gde je f_a frekvencija signala, K ceo broj, a n je red harmonika.



Slika 4.2.1.1. *Lokacije harmonijskih izobličenja za signal čija je frekvencija 7 MHz i učestanost odabiranja 20 MHz* (Analog Devices Inc., Engineeri, 2005)

Harmonijska izobličenja se mogu dati kao snaga harmonika u decibelima koji se javljaju kada se na ulaz AD konvertora stavi sinusoida maksimalne amplitude (uglavnom 0.5 dB i 1 dB ispod pune skale) i najčešće

se daje prvih nekoliko harmonika. Kod nekih proizvođača se može naći i najgori harmonik (eng. *worst harmonic*), odnosno snaga najjačeg harmonika.

Ipak, mnogo češće korišćene metrike su totalno harmonijsko izobličenje (eng. *total harmonic distortion - THD*) i totalno harmonijsko izobličenje sa šumom (eng. *total harmonic distortion plus noise – THDN*).

Totalno harmonijsko izobličenje se računa po formuli:

$$THD = 20\log\left(\frac{\sqrt{V_2^2 + V_3^2 + \cdots + V_n^2}}{V_1}\right) \tag{4.2.1.1}$$

gde su $V_2, V_3, \ldots, V_n$ harmonici i najčešće se uzima prvih pet.

Totalno harmonijsko izobličenje plus šum se računa kao totalno harmonijsko izobličenje, s tim da se na sumu harmonika dodaje i šum. Za ovu metriku je neophodno definisati i opseg frekvencija na kojima se računa šum. Ukoliko je ovaj opseg ceo opseg AD konvertora, odnosno obuhvata sve frekvencije od DC (ne uključujući DC) do polovine frekvencije odabiranja, tada je ova metrika ekvivalentna meri signal/šum plus izobličenja.

4.2.2. Odnos signal/šum

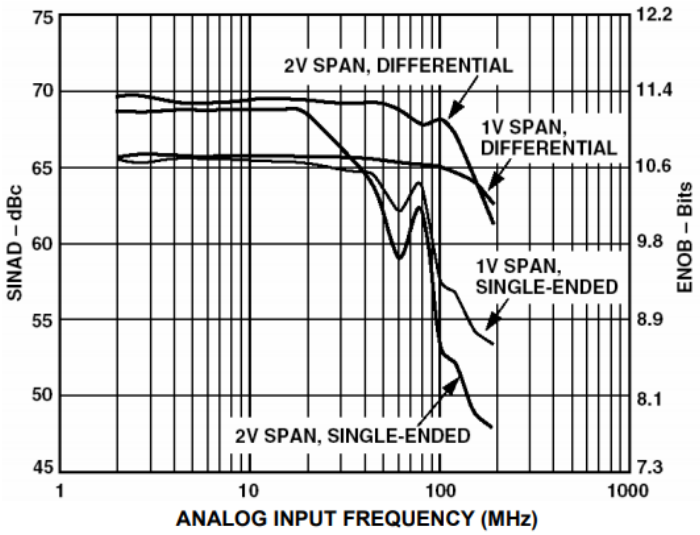
Odnos signala i šuma se takođe predstavlja na nekoliko različitih načina. Najčešće se koristi odnos signal/šum plus izobličenja (eng. *signal to noise plus distortion ratio - SINAD*) ili samo signal/šum (eng. *signal noise ratio – SNR*).

SINAD meri odnos snage signala i svih ostalih komponenti, ne računajući DC komponentu, dok SNR meri isti odnos, ali ne uključuje harmonike.

Sa ovim u vezi se definiše i efektivni broj bitova AD konvertora na bazi formule 3.5.8. s tim da se najčešće koristi SINAD, a ne SNR odnos:

$$ENOB = \frac{SINAD - 1.76\ \text{dB}}{6.02\ \text{dB}} \tag{4.2.2.1}$$

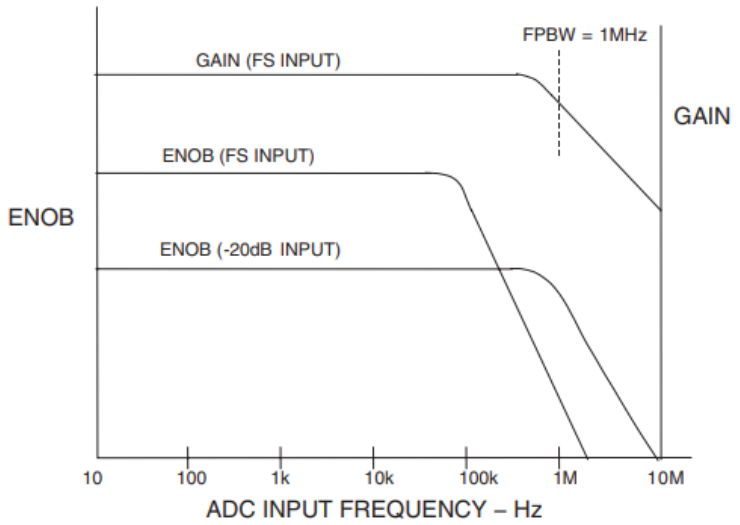
SINAD se smatra jednom od najbitnijih dinamičkih karakteristika AD konvertora obzirom da se uzimaju u obzir sve moguće greške koje se javljaju pa se dobija najrealnija slika o radu konvertora. Takođe, često se prikazuje i zavisnost ove vrednosti od napona i frekvencije ulaznog signala kao što je prikazano na slici 4.2.2.1. Pri tome se koriste i frekvencije koje su veće od Nikvistove da bi se prikazalo kako konvertor radi u slučaju kada se odabiraju signali koji se mogu odabirati manjom frekvencijom od maksimalne (eng. *undersampling*) koji su objašnjeni u odeljku 3.1.



Sila 4.2.2.1. *Primer različitih vrednosti SINAD parametara i broja ENOB za različite frekvencije i amplitude ulaznog signala* (Analog Devices Inc., Engineeri, 2005)

4.2.3. Frekvencijski opseg AD konvertora

Frekvencijski opseg AD konvertora (eng. *analog bandwidth*) definiše učestanost za koju se snaga signala smanji za 3 dB. Nekada se ovaj opseg posebno definiše za signale manje amplitude (eng. *small signal bandwidth – SSBW*), a posebno za signale čija je ampliduta blizu maksimalne (eng. *full power bandwidth – FFBW*). Bitno je napomenuti da ova specifikacija ne definiše frekvenciju do koje AD konvertor radi dobro. Name, SINAD, odnosno ENOB postaju lošiji na učestanostima koje su niže od naznačene. Na slici 4.2.3.1. je prikazan primer u kome se jasno vidi da efektivan broj bitova značajno opada mnogo pre granične učestanosti pojačanja AD konvertora kada je amplituda signala veća.



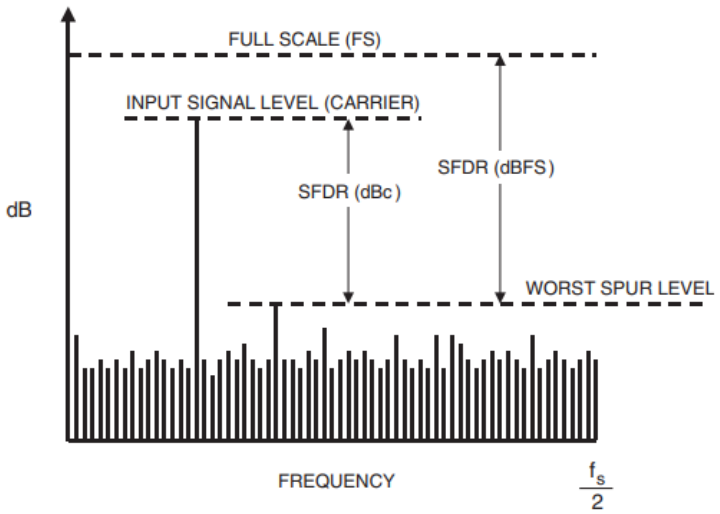
Slika 4.2.3.1. *Efektivan broj bitova AD konvertora u zavisnosti od frekvencije i amplitude ulaznog napona naspram frekvencije na kojoj je signal oslabljen za 3 dB (FPBW)* (Analog Devices Inc., Engineeri, 2005)

4.2.4. Lažni slobodni dinamički raspon AD konvertora

Lažni slobodni dinamički raspon AD konvertora (eng. *Spurious Free Dynamic Range – SFDR*) je jedan od najznačajnijih parametara AD konvertora. Ovaj parametar prikazuje odnos između snage signala i najveće vrednosti u spektru koja je šum.

Na slici 4.2.4.1. je prikazano kako se određuje ovaj raspon. Prikazana su dva načina koja se koriste. Moguće je meriti jačinu ovog lažnog signala u odnosu na napon pune skale ili u odnosu na amplitudu ulaznog signala.

Za signale koji imaju veću amplitudu ovaj lažni signal je najčešće jedan od harmonika, dok za manje amplitude to uglavnom nije slučaj. Dodavanjem šuma moguće je poboljšati ovu karakteristiku, ali ujedno i pogoršati SNR.



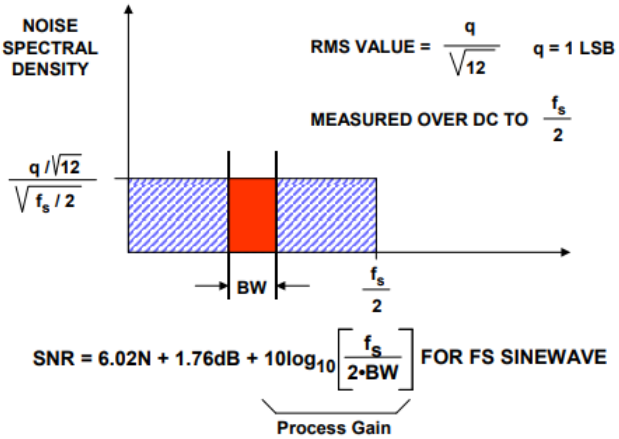
Slika 4.2.4.1. *Merenje lažnog slobodnog dinamičkog opsega*

4.2.5. Povećanje SNR-a usled ograničenog opsega signala

Kada je signal u opsegu koji je manji od Nikvistovog širine BW tada se mogu izabaciti komponente koje su van tog opsega, a sa njima i šum. U tom slučaju se SNR poboljšava i zajedno sa faktorom poboljšanja postaje:

$$SNR = 6.02N \text{ dB} + 1.76 \text{ dB} + 10 \log \frac{f_s}{2\,BW} \tag{4.2.5.1}$$

Na slici 4.2.5.1. je prikazana ova situacija.



Slika 4.2.5.1. *Uticaj ograničenosti spektra signala na poboljšanje odnosa signal-šum* (Kester, 2020)

4.3. Ostale karakteristike AD konvertora

U ovom odeljku će biti objašnjene karakteristike AD konvertora koje se nalaze u specifikacijama proizvođača, ali ne potiču od samog AD konvertora već od pratećih komponenti.

4.3.1. Faktor potiskivanja signala srednje vrednosti

AD konvertor ima najčešće dva ulaza i napon koji se konvertuje je razlika napona na ovim ulazima. Kada na ulazu postoji diferencijalni pojačavač, za njega se definiše faktor postiskivanja signala srednje vrednosti (eng. *common mode rejection ratio – CMRR*). Idealni diferencijalni pojačavač ima izlazni napon opisan formulom:

$$V = A_d(V_+ - V_-) \tag{4.3.1.1}$$

gde je A_d diferencijalno pojačanje. Kod realnog diferencijalnog pojačavača, izlazni napon je dat formulom:

$$V = A_d(V_+ - V_-) + A_s(V_+ + V_-) \tag{4.3.1.2}$$

Faktor potiskivanja signala srednje vrednosti je:

$$CMRR = \left|\frac{A_d}{A_s}\right| \tag{4.3.1.3}$$

Ovaj faktor se još izražava i u decibelima i tada se označava sa CMR (eng. *Common Mode Rejection - CMR*):

$$CMR = 20\ \log CMRR \tag{4.3.1.4}$$

4.3.2. Otpornost AD konvertora na varijaciju referentnog napona

Sposobnost kola da potisne promene napona napajanja (eng. power supply rejection ratio – PSRR) govori o tome kolika greška nastaje zbog nestabilnosti referentnog napona:

$$PSRR = 20\ \log\frac{\Delta V_{CC}}{\Delta V_{out}} \tag{4.3.2.1}$$

Ova karakteristika se najčešće meri tako što se napon napajanja superponira sa sinusoidom i meri varijacija izlaznog napona kola u tom slučaju.

5. Sigma-delta AD konvertor

Ideja sigma-delta AD konvertora[4] je da se ulazni napon konvertuje u povorku pravougaonih impulsa čija je srednja vrednost proporcionalna vrednosti ulaznog napona. Radi se o ideji koja je analogna impulsno-širinskoj modulaciji, samo u suprotnom smeru.

Ideja o sigma-delta modulatoru se prvi put pojavljuje 1962. godine, ali tek u skorije vreme, sa razvojem VLSI tehnologija dobija na značaju. Sa razvojem VLSI tehnologija javlja se jednostavan način za implementaciju kompleksnih kola za obradu digitalnih signala. Zbog toga što se najveći deo ovog konvertora bazira na digitalnim komponentama, veoma jednostavno se i analogni i digitalni deo implementiraju na istom čipu. Ovo je konvertor visoke rezolucije koji je precizan i ekonomičan.

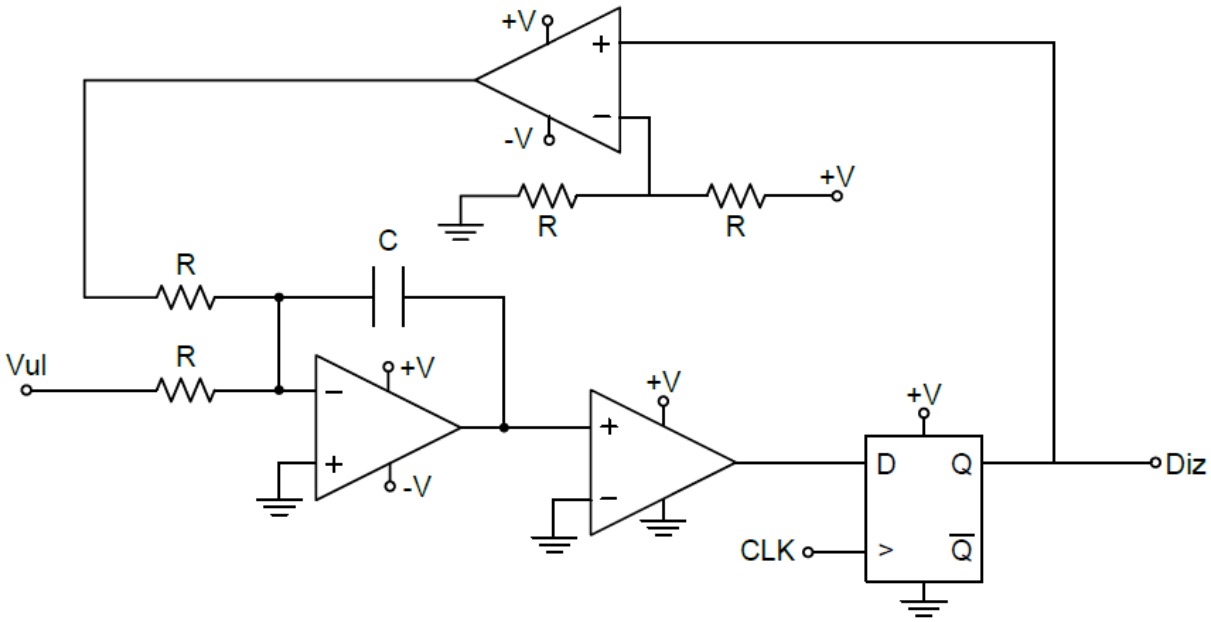
U poglavlju 3.1. je diskutovana teorema odabiranja koja govori o tome kolika je minimalna frekvencija odabiranja signala. Dosta AD konvertora koristi frekvenciju odabiranja koja je blizu Nikvistove učestanosti. VLSI tehnologije omogućavaju mnogo veće brzine i dosta ovih konvertora ne može da iskoristi tu prednost. Sigma-delta AD konvertor može da iskoristi ove prednosti i odabira signal sa mnogo višom frekvencijom. Odabiranje signala većom frekvencijom dovodi do veoma zanimljivog efekta takozvanog oblikovanja šuma kvantizacije koji je jedna od glavnih karakteristika ovog konvertora.

Sigma-delta AD konvertor se sastoji iz dve osnovne komponente – sigma-delta modulatora i digitalnog filtra. Obe komponente su analizirane u ovom poglavlju.

[4] Za ovaj konvertor se još i koristi naziv delta-sigma AD konvertor. Sigma-delta se preporučuje jer bolje objašnjava princip rada sigma-delta modulatora u kome se sumira signal sa razlikom DA konvertora, mada je i bez preporuke češće korišćeni naziv.

5.1. Sigma-delta modulator

Na slici 5.1. je prikazana šema sigma delta modulatora koji predstavlja ulaz ovog AD konvertora koja formira povorku pravougaonih impulsa čija je srednja vrednost proporcionalna ulaznom naponu.



Slika 5.1.1. *Sigma delta modulator* (D. Živković, 1997)

U zavisnosti od toga da li je izlaz flip-flopa u trenutku t bio 1 ili 0, izlaz integratora će u trenutku $t + T_{CLK} - \varepsilon$ biti:

$$V_{int}(t + T_{CLK} - \varepsilon) = V_{int}(t) - \frac{V + V_{ul}}{RC} T_{CLK}, \qquad Q(t) = 1 \tag{5.1.1}$$

$$V_{int}(t + T_{CLK} - \varepsilon) = V_{int}(t) + \frac{V - V_{ul}}{RC} T_{CLK}, \qquad Q(t) = 0 \tag{5.1.2}$$

Komparator koji se nalazi gore ustvari predstavlja jednobitni DA konvertor uz pomoć kog je ostvarena negativna povratna sprega. Pod pretpostavkom da je $V > 0$ i $|V_{ul}| < V$, napon na integratoru opada kada je na izlazu flip-flopa jedinica jer je $V + V_{ul} > 0$ za svaki ulazni napon. Sa druge strane, ukoliko je na izlazu flip-flopa nula, napon na integratoru raste jer je $V - V_{ul} > 0$ za svaki ulazni napon.

Zbog komparatora čiji je jedan ulaz izlaz integratora, izlaz flip-flopa je definisan sledećim formulama:

$$Q(t + T_{CLK} + \varepsilon) = 1, \qquad V_{int}(t + T_{CLK} - \varepsilon) > 0 \tag{5.1.3}$$

$$Q(t + T_{CLK} + \varepsilon) = 0, \qquad V_{int}(t + T_{CLK} - \varepsilon) < 0 \tag{5.1.4}$$

Posmatrajmo šta se dešava sa izlazom ovog kola kada je ulazni napon $V_{ul} = \frac{V}{2}$. Pretpostavimo da je u trenutku $t = 0^+$ napon na izlazu integratora bio manji od nule. To znači da je u početnom trenutku izlaz flip-flopa takođe nula, pa se napon na integratoru u sledećem taktom signalu računa na osnovu formule (5.1.2.) i zamenom vrednosti dobijamo:

$$V_{int}(T_{CLK} - \varepsilon) = V_{int}(0) + \frac{1}{2}\frac{V}{RC}T_{CLK} \tag{5.1.5}$$

Pretpostavimo, dalje, da je ovo dovelo do toga da napon na integratoru postane pozitivan. Tada će izlaz flip-flopa postati visok logički nivo, a izlaz integratora za $t = 2T_{CLK} - \varepsilon$ se računa po formuli (5.1.1.) i dobijamo:

$$V_{int}(2T_{CLK} - \varepsilon) = V_{int}(T_{CLK}) - \frac{3}{2}\frac{V}{RC}T_{CLK} \tag{5.1.6}$$

Zamenom vrednosti $V_{int}(T_{CLK})$ dobijamo:

$$V_{int}(2T_{CLK} - \varepsilon) = V_{int}(0) + \frac{1}{2}\frac{V}{RC}T_{CLK} - \frac{3}{2}\frac{V}{RC}T_{CLK} = V_{int}(0) - \frac{1}{2}\frac{V}{RC}T_{CLK} < 0 \tag{5.1.7}$$

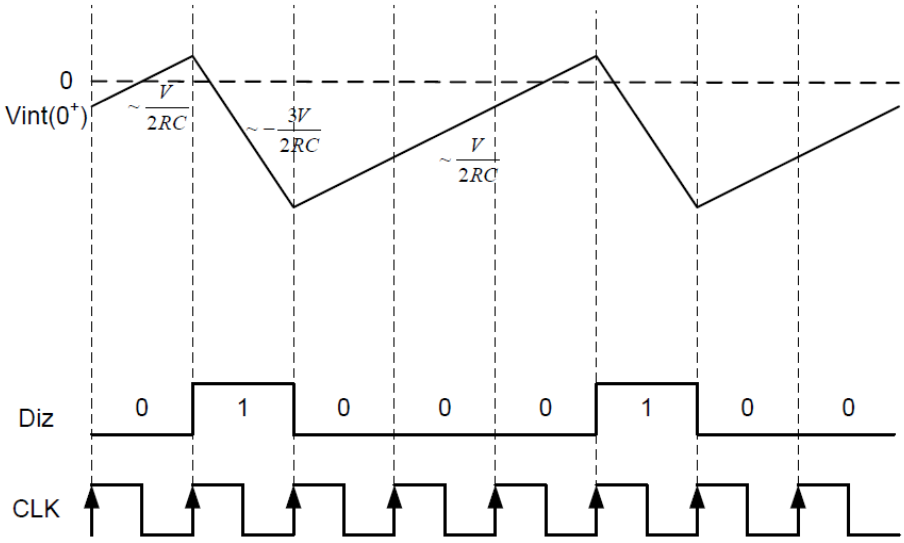
Prema tome, sada se opet menja izlaz flip-flopa, i ponovo napon na integratoru počinje da raste.

Kako napon na izlazu integratora raste tri puta sporije, trebaće tri puta više taktnih signala da se vrati na pozitivnu vrednost. Primetimo da postoje ista maksmalna i minimalna vrednost između koje izlaz integratora osciluje (to su vrednosti dobijene u formulama (5.1.5.) i (5.1.6.). Na osnovu ovoga može se nacrtati grafik prikazan na slici 5.1.2.

Kao što se vidi sa slike, vreme koje izlaz flip-flopa provede na visokom naponskom nivou je proporcionalno zbiru $V + V_{ul}$, dok je vreme provedeno na niskom naponskom nivou proporcionalno njihovoj razlici $V - V_{ul}$.

Dakle, broj intervala sa niskim izlazom je direktno proporcionalan naponu V_{ul}.

Dato izvođenje podrazumeva da ne dolazi do promene režima rada nijednog od datih elemenata. Dobijene formule ne važe ukoliko jedna od njih uđe u zasićenje. Prema tome, potrebno je definisati opseg ulaznog napona za koji modulator radi ispravno.



Slika 5.1.2. *Izlazi integratora i flip-flopa za* $V_{ul} = \frac{V}{2}$ (D. Živković, 1997)

Zbog simetričnog napajanja integratora i simetričnog izlaza DA konvertora i granica zasićenja će takođe biti simetrična. Integrator dostiže maksimalnu vrednost kada je početna vrednost negativna i jako bliska nuli, odnosno $V_{int}(0) = -\varepsilon$, a napon na izlazu raste najvećom brzinom, što se dešava za $V_{ul} = -V$.

Pošto napon raste sve do prve uzlazne ivice takta, maksimalna vrednost izlaznog napona integratora je:

$$V_{int,MAX} = \frac{2V}{RC} T_{CLK} \tag{5.1.8}$$

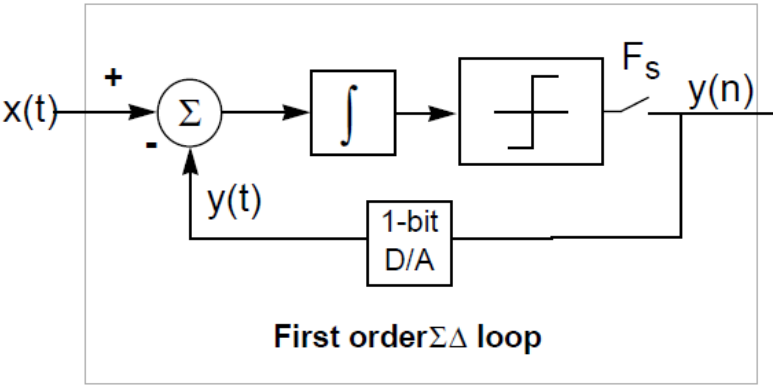
Da bi integrator radio u linearnom režimu, ovaj napon mora biti manji od napona napajanja, odakle se dobija uslov koji mora biti zadovoljen da bi sigma delta modulator radio ispravno:

$$\frac{2V}{RC} T_{CLK} < V \Rightarrow RC > 2T_{CLK} \tag{5.1.9}$$

5.2. Analiza sigma-delta modulatora u Z domenu

Iako je na ulazu sigma-delta konvertora analogni signal, ukoliko se analiza signala ograniči na analizu u diskretnim vremenskim trenucima, signali se mogu analizirati u Z domenu. Ovakva analiza je opravdana jer se signal menja u trenucima koji su umnožak perioda taktnog signala.

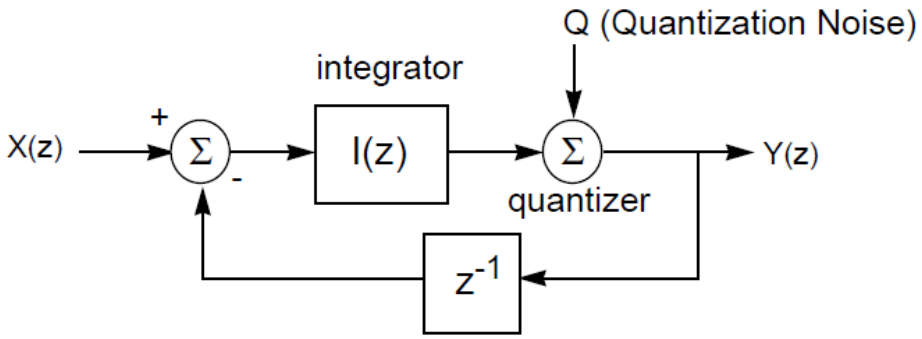
Na slici 5.2.1. je prikazan blok dijagram sigma delta modulatora.



Slika 5.2.1*. Blok dijagram sigma delta modulatora (vremenski domen)* (Sangil Park, 2008)

Potrebno je napomenuti da blok dijagram prikazan na prethodnoj i sledećoj slici ne odgovara u potpunosti šemi prikazanoj na slici 5.1.1. Naime, na šemi DA konvertor konvertuje nizak i visok logički nivo u -V i V, respektivno, dok se sa blok dijagrama dobija konverzija u vrednosti 0, V. Analiza ovako pojednostavljenog blok dijagrama ne dovodi do gubljenja opštosti dobijenog zaključka.

Sa blok dijagrama datog na slici 5.2.1. se lako prelazi u dijagram u Z domenu. DA konvertor se svodi na prosto kašnjenje signala za jednu periodu, dok se kvantizacija može predstaviti kvantizatorom na čiji se izlaz dodaje šum koji predstavlja šum kvantizacije. Ovim dobijamo blok dijagram prikazan na slici 5.2.2.



Slika 5.2.2*. Blok dijagram sigma delta modulatora (Z domen)* (Sangil Park, 2008)

Sa slike 5.2.2. se dobija sledeći izraz:

$$Y(z) = Q(z) + I(z)[X(z) - z^{-1}Y(z)] \tag{5.2.1}$$

Rešavanjem ove jednačine po $Y(z)$ se dobija:

$$Y(z) = X(z)\frac{I(z)}{1 + z^{-1}I(z)} + Q(z)\frac{1}{1 + z^{-1}I(z)} \tag{5.2.2}$$

Prenosna funkcija integratora je data izrazom:

$$I(z) = \frac{1}{1 - z^{-1}} \tag{5.2.3}$$

Zamenom jednačine 5.2.3. u 5.2.2. se dobija:

$$Y(z) = X(z) + Q(z)(1 - z^{-1}) \tag{5.2.4}$$

Posmatrajmo posebno kako se greška kvantizacije ponaša u frekvencijskom domenu. Izlazni signal se može shvatiti kao zbir dva različita signala – stvarnog ulaznog signala i signala koji predstavlja šum, odnosno grešku kvantizacije.

Prenosna funkcija signala greške kvantizacije je tada data izrazom:

$$H_Q(z) = \frac{Y(z)}{Q(z)} = 1 - z^{-1} \tag{5.2.5}$$

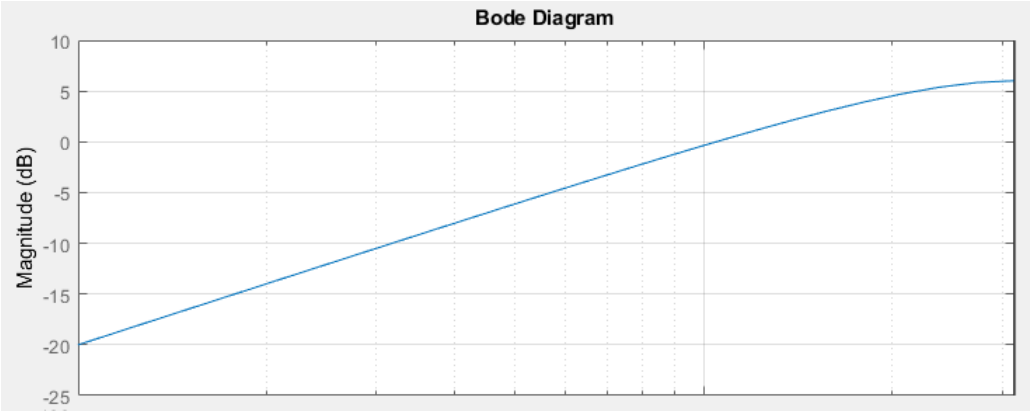
Smenom $z = e^{sT}$ može da se vrati u frekvencijski domen (eng. *star, starred transform*). Dobija se:

$$H(s) = 1 - e^{sT} \tag{5.2.6}$$

Pri tome je T period odabiranja ulaznog signala. Odnosno, $T = 1/F_s$.

Kao što je već objašnjeno u odeljku 3.1., ovaj spektar je periodičan, sa periodom F_s.

Na slici 5.2.3. je prikazana Bodeova amplitudska karakteristika ove prenosne funkcije dobijena korišćenjem MATLAB-a (gde maksimalna učestanost ide do polovine učestanosti semplovanja signala).

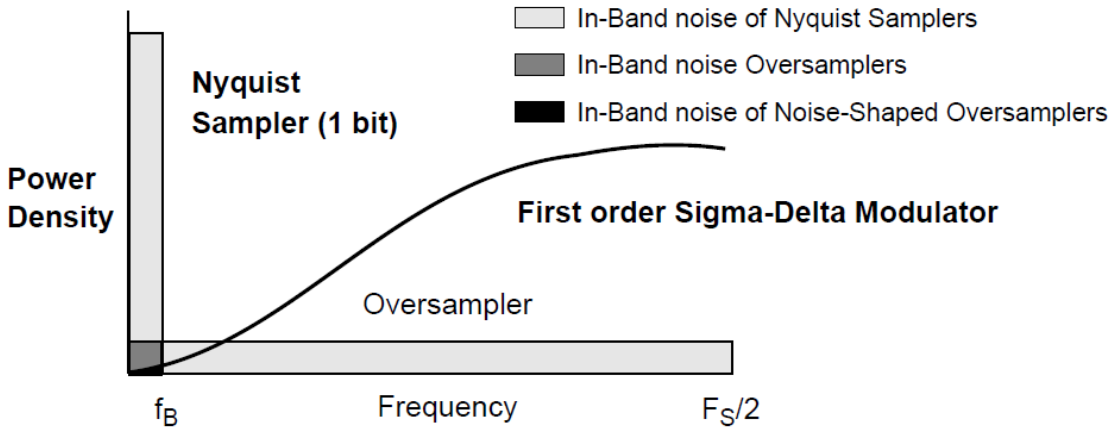


***Slika 5.2.3**. Bodeova amplitudska karakteristika prenosne funkcije greške kvantizacije*

Sa slike se vidi da je greška kvantizacije na nižim učestanostima oslabljena, a pojačana na višim učestanostima. Sigma-delta modulator ima veoma zgodnu osobinu, a to je da "gura" grešku kvantizacije ka višim učestanostima. Ovaj efekat se zove oblikovanje šuma (eng. *noise shaping*).

Sa druge strane, vidi se da je prenosna funkcija ulaznog signala jednaka jedinici, pa se spektar originalnog signala neće promeniti. Ovde se prirodno javlja ideja da se signal odabira sa znatno većom učestanošću od Nikvistove, a da se nakon toga filtira NF flitrom. Na ovaj način se originalni signal ne gubi, ali se izbacuje šum koji se nalazi na višim učestanostima.

Ova ideja je prikazana na slici 5.3.4.



Slika 5.2.4. *Spektar snage šuma prvostepenog sigma delta modulatora* (Sangil Park, 2008)

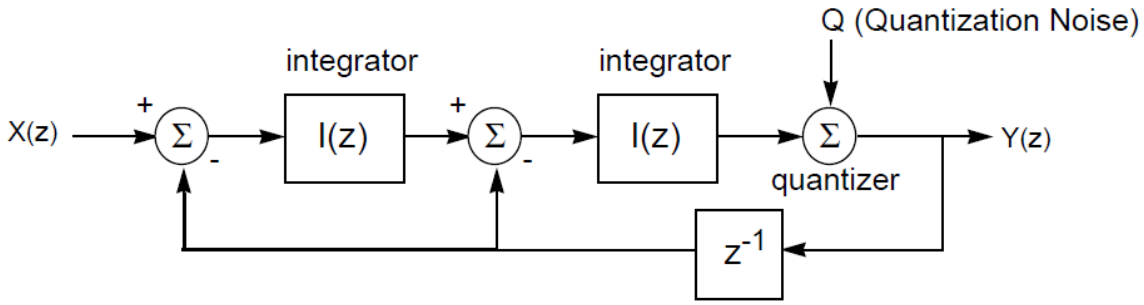
Na slici 5.2.4. je prikazano kako se šum različitih AD konvertora ponaša u frekvencijskom domenu. Sa slike se jasno vidi da konvertori koji rade na Nikvistovoj učestanosti odabiranja imaju maksimalan šum kvantizacije, obzirom da je ukupan šum ravnomerno raspoređen u opsegu učestanosti odabiranja. Kod AD konvertora koji imaju učestanost odabiranja znatno višu od Nikvistove, snaga šuma u nižem delu spektra je, naravno, manja, obzirom da se sada isti taj šum ravnomerno raspoređuje na širem delu spektra. Konačno, prikazana je snaga šuma sigma-delta modulatora i na slici se jasno vidi da je snaga šuma u spektru od interesa u ovom slučaju najmanja. Ovo je plaćeno time što je šum na višim učestanostima pojačan, ali ove učestanosti svakako nisu od interesa.

Ovde je ponovo bitno napomenuti da se ovde koristi pretpostavka da je šum ravnomerno raspoređen u opsegu do učestanosti odabiranja. Ovaj model šuma, pa samim tim i efekat oblikovanja šuma ne važe kada je signal koji se konvertuje približno DC jer je greška kvantizacije tada praktično konstantna. Takođe, ako je je signal periodičan i njegova učestanost je parni umnožak učestanosti odabiranja, ili kada je signal deo

negativne povratne sprege, postoji autokorelacija između signala i šuma, pa nije opravdano pretpostaviti da će on biti ravnomerno raspodeljen na pomenutom opsegu.

5.3. Sigma delta modulatori višeg reda

Već prilično dobre karakteristike sigma delta modulatora se mogu još više poboljšati ako se koristi još jedan integrator kao što je prikazano na slici 5.3.1. Pošto je prenosna funkcija šuma od interesa, radi lakšeg izračunavanja, može se zanemariti signal $X(z)$.



Slika 5.3.1. *Blok šema sigma delta modulatora drugog reda*

Neka je sa $Y_1(z)$ obeležen izlaz prvog integratora. Ovaj izlaz je dat sledećim izrazom:

$$Y_1(z) = -z^{-1}I(z)Y(z) \tag{5.3.1}$$

Izlaz $Y(z)$ je dat izrazom:

$$Y(z) = Q(z) + I(z)[Y_1(z) - z^{-1}Y(z)] \tag{5.3.2}$$

Zamenom vrednosti izraza za $Y_1(z)$ i gornju jednačinu i sređivanjem izraza, dobija se:

$$Y(z) = (1 - z^{-1})^2 Q(z) \tag{5.3.4}$$

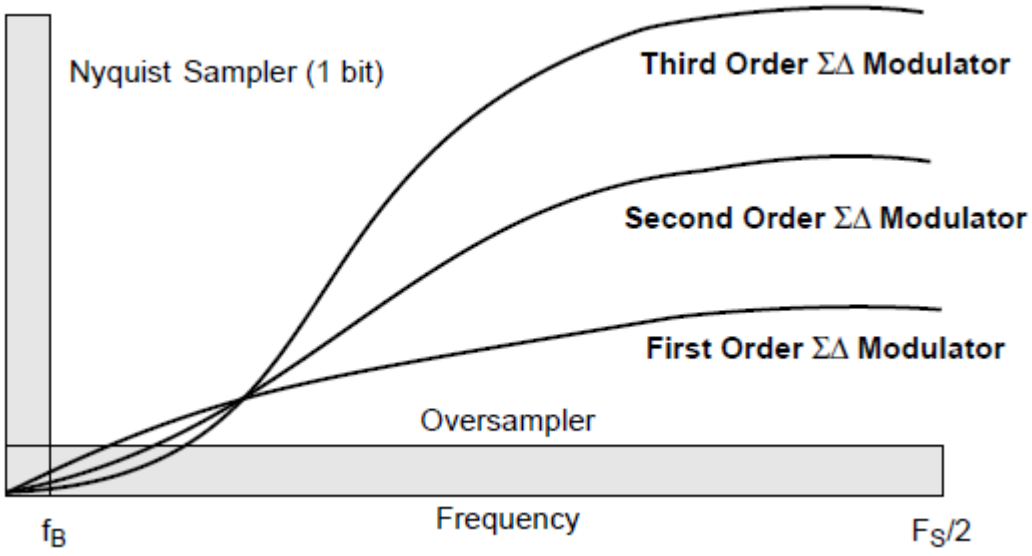
Dobijeni rezultat je još bolji nego u slučaju sigma delta modulatora prvog reda.

Analogno, može se konstruisati i delta sigma modulator trećeg reda dodavanjem još jednog integratora ispred ulaznog signala i uvođenjem povratne sprege sa izlazom. Tada se, na isti način kao što je dobijeno u slučaju modulatora drugog reda, dobija prenosna funkcija šuma kvantizacije sigma delta modulatora trećeg reda, koja je data izrazom:

$$Y(z) = (1 - z^{-1})^3 Q(z) \tag{5.3.4}$$

Na slici 5.3.2. je prikazana snaga šuma za sva tri stepena delta modulatora.

Sa slike se jasno vidi da svaki naredni stepen poboljšava karakteristike modulatora, odnosno, snaga šuma u opsegu od interesa je sve manja. Ipak, treba primetiti da su ova poboljšanja sve manja kako se povećava stepen modulatora, a kompleksnost kola raste. Osim toga, sa povećanjem stepena kolo može čak postati nestabilno. Zbog toga se ne koriste modulatori reda većeg od tri.



Slika 5.3.2. *Spektar snage šuma sigma delta modulatora prvog, drugog i trećeg stepena* (Sangil Park, 2008)

5.4. Digitalni filtri koji se koriste u sigma delta AD konvertorima

Iz prethodne analize se videlo da sigma delta modulator proizvodi povorku nula i jednica čija je srednja vrednost proporcionalna ulaznom naponu. Posmatrano u vremenskom domenu potrebno je na neki način pronaći prosečnu vrednost ovog signala da bi se dobila konačna digitalna vrednost. Odnosno, posmatrano u Z domenu, potrebno je filtrirati signal obzirom da je frekvencija odabiranja značajno veća od maksimalne frekvencije u spektru signala.

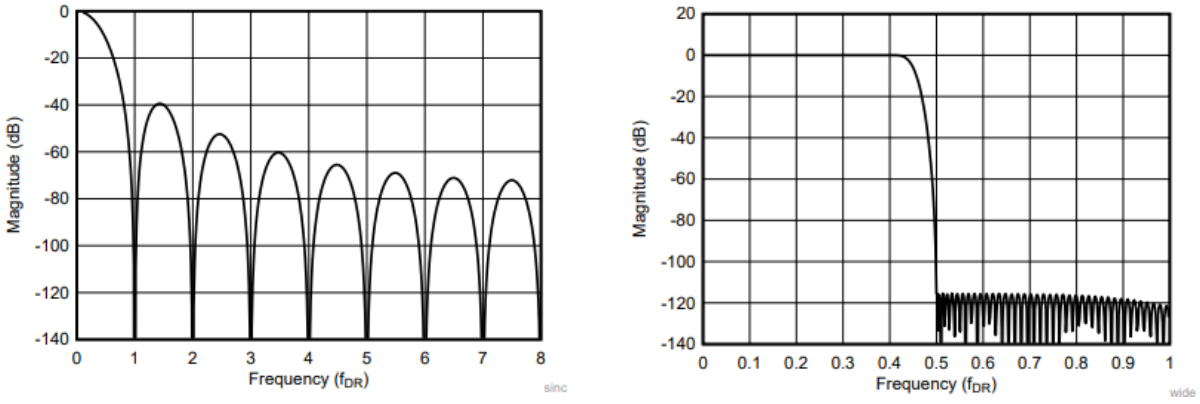
Filtrirani signal treba da ima manju frekvenciju i veći broj bita. Odnos frekvencije na kojoj radi sigma delta modulator i odnos frekvencije filtriranog signala se obeležava sa OSR (eng. *oversampling rate - OSR*).

$$OSR = \frac{f_m}{f_s} \tag{5.4.1}$$

Najčešće korišćeni digitalni filtar je sinc filtar. Ovaj fitar jasno implementira koncept na osnovu kog radi sigma-delta AD konvertor. *Sinc* filtar u vremenskom domenu izvršava usrednjavanje signala. Osim *sinc* filtra koristi se i *wideband* filtar. Na slici 5.4.1. su prikazane frekvencijske karakteristike i jednog i drugog filtra, gde je f_{DR} definisano kao:

$$f_{DR} = \frac{f_{MOD}}{OSR} \tag{5.4.2}$$

Granična učestanost je jednaka polovini f_{DR}.



Slika 5.4.1. *Spektar sinc3 i wideband filtaara* (Texas Instruments)

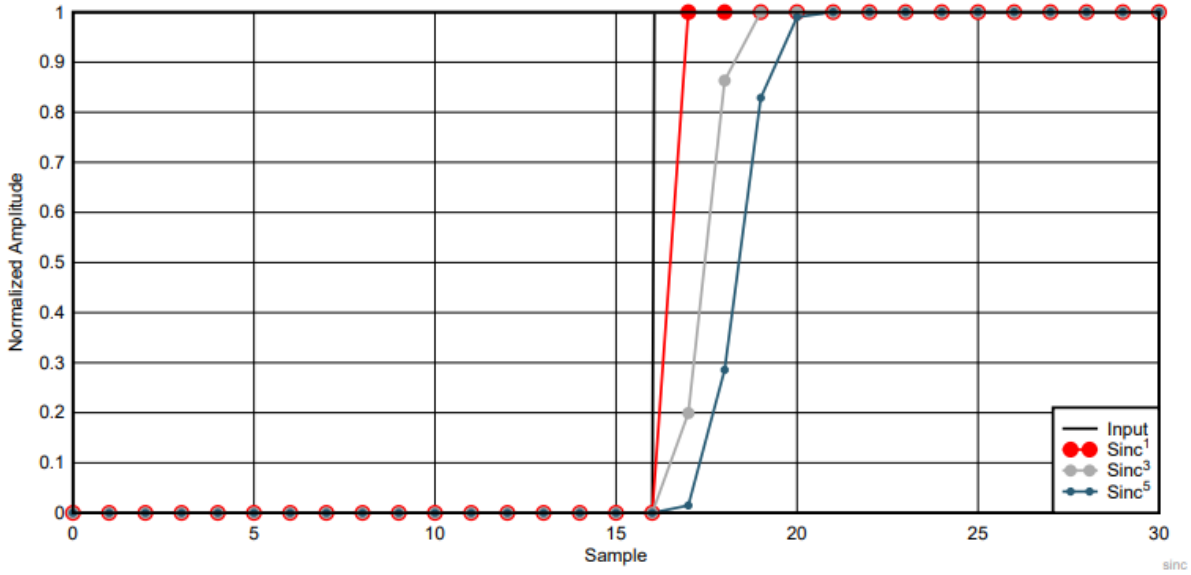
Oba filtra imaju konačan impulsni odziv (eng. *Finite Response Filters* - *FIR*). Oba filtra su stabilna i jednostavno se dizajniraju. Kao što se vidi sa slike, *sinc* filtar ima lošije karakteristike u frekvencijskom domenu. Signali koji su na učestanosti $0.262 f_{DR}$ su slabiji za $3dB$, a signali koji imaju učestanost veću od granične nisu dovoljno oslabljeni, pa se propušta dosta šuma i to do dovodi do preklapanja spektra. Sa druge strane, wideband filtar ima sjajne frekvencijske karakteristike – veliki nagib, deo signala u opsegu učestanosti od interesa nije oslabljen, a visoke učestanosti su značajno oslabljene što dovodi do toga da je i šum koji se propušta mnogo manji, pa samim tim i uticaj preklapanja na spektar signala.

Iako u frekvencijskom domenu *wideband* filtar deluje kao daleko bolje rešenje, ako posmatramo ove filtre u vremenskom domenu postaje jasno zašto je ipak *sinc* filtar daleko češće korišćen.

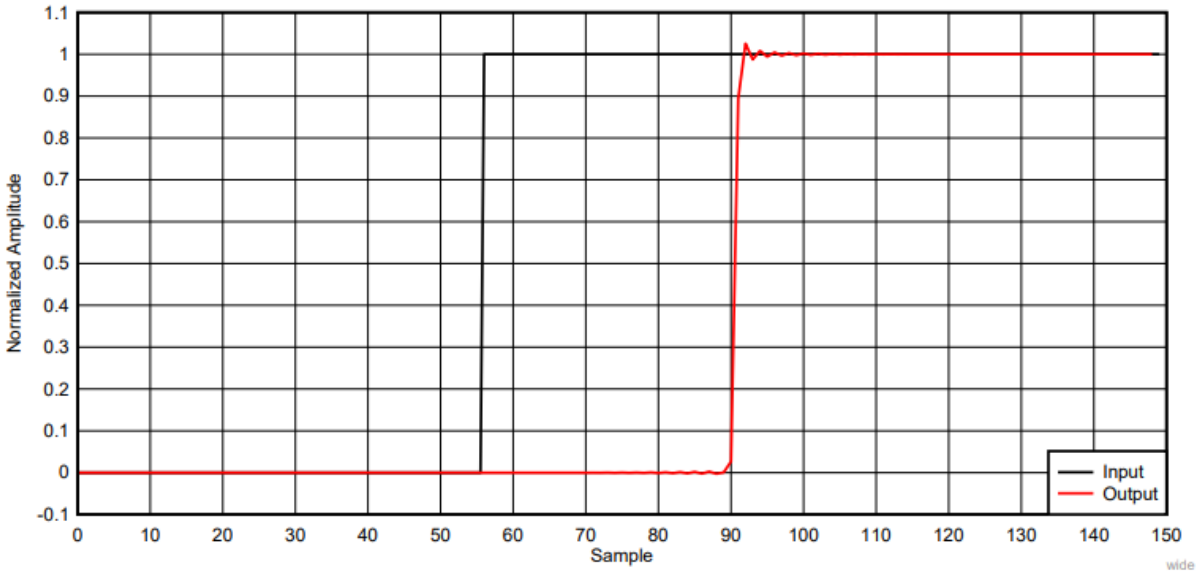
Na slici 5.4.2. je prikazan odziv sinc filtra na step pobudu, dok je na slici 5.4.3. prikazan odziv *wideband* filtra na istu pobudu.

Sa slika se jasno vidi da je odziv *wideband* filtra veoma spor. Ukoliko postoji više ulaznih kanala koje ciklično treba konvertovati, *sinc* filter je daleko bolja opcija. U zavisnosti od toga sa kakvim se ulaznim

signalima radi, *sinc* filtar može biti sasvim zadovoljavajućih karakteristika. Na primer, ako se konvertuju signali iz temperaturnih senzora koji imaju niske učestanosti *sinc* filtri daju dobre rezultate.



Slika 5.4.2. *Odziv sinc filtra na step pobudu* (Texas Instruments)



Slika 5.4.3. *Odziv wideband filtra na step pobudu* (Texas Instruments)

Ukoliko se radi o signalima koji imaju visoke učestanosti (audio, vibracije i slično) *wideband* filter je mnogo bolja opcija. Kod ovakvih signala *sinc* filtar će oslabiti deo signala na učestanostima od interesa, dok *wideband* filtar ima gotovo ravnu karakteristiku do Nikvistove učestanosti, a potom veoma strmo pada

za više učestanosti. Osim toga, kod naizmeničnih signala efekat preklapanja spektra može da bude problematičan, a u slučaju *sinc* filtra komponente na učestanostima većim od Nikvistove nisu dovoljno oslabljene, što nije slučaj sa *wideband* fitrom.

5.5. Sinc filtar

Sinc filtar je najčešće korišćeni filtar u sigma delta AD konvertorima i filtar koji se koristi u AD konvertoru mikrokontrolera msp430i2041.

Sinc filtar je dobio ime po svom fekvencijskom odzivu koji je oblika $\frac{\sin(x)}{x}$, mada se koristi i *comb sinc* naziv. Kao što je već pomenuto, *sinc* filtar usrednjava signal za određeni broj odbiraka M. Kako je potrebno dobiti signal sa frekvencijom odabiranja f_s, a ulazni signal ima frekvenciju f_m broj odbiraka koji je potrebno uzeti je, na osnovu formule 5.4.2. upravo OSR. Ovaj broj se zove i faktor decimacije pošto opisuje za koliko se smanjuje učestanost signala, to jest koliko odbiraka se odbacuje. Pošto je potrebno prebrojiti ukupan broj jedinica, iako se radi usrednjavanje, zapravo nema deljenja sa brojem odbiraka. Broj bitova potrebnih za predstavljanje signala je objašnjen u narednom odeljku, a ovde će biti zanemaren.

Na osnovu ovoga može da se izvede formula za usrednjavanje signala:

$$y[n] = \sum_{i=0}^{M-1} x[n-i] \tag{5.5.1}$$

Uzimajući u obzir da je

$$Z\{x[n-k]\} = z^{-k}Z\{x\} \tag{5.5.2}$$

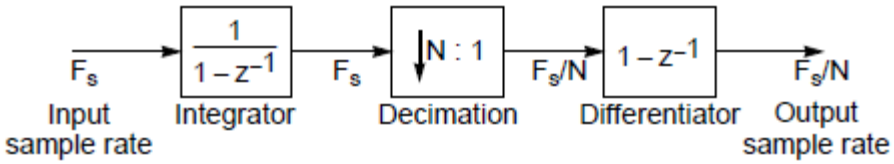
dobija se Z transformacija usrednjavanja signala:

$$Y(z) = X(z)\sum_{i=0}^{M-1} z^{-i} \tag{5.5.3}$$

Korišćenjem formule za geometrijsku sumu dobijamo prenosnu funkciju u zatvorenoj formi:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1-z^{-M}}{1-z^{-1}} \tag{5.5.4}$$

Ova prenosna funkcija se može posmatrati kao kaskadna veza dve prenosne funkcije – integratora sa učestanošću f_m i diferencijatora sa učestanošću $OSR\ f_m = f_s$. Na osnovu ovoga se ovaj filtar može prikazati blok dijagramom datim na slici 5.5.1.



Slika 5.5.1. *Blok šema sinc filtra prvog reda* (Sangil Park, 2008)

Prenosnu funkciju u frekvencijskom domenu dobijamo zamenom $z = e^{j\omega}$.

$$H(j\omega) = \frac{1 - e^{-j\omega M}}{1 - e^{-j\omega}} \tag{5.5.5}$$

gde je $\omega = 2\pi \frac{f}{f_M}$.

Gornji izraz može da se transformiše na sledeći način:

$$H(j\omega) = \frac{e^{-\frac{j\omega M}{2}}}{e^{-\frac{j\omega}{2}}} \frac{\frac{e^{-\frac{j\omega M}{2}} - e^{-\frac{j\omega M}{2}}}{2}}{\frac{e^{-\frac{j\omega}{2}} - e^{-\frac{j\omega}{2}}}{2}} = \frac{e^{-\frac{j\omega M}{2}}}{e^{-\frac{j\omega}{2}}} \frac{\sinh\left(-\frac{j\omega M}{2}\right)}{\sinh\left(-\frac{j\omega}{2}\right)} \tag{5.5.6}$$

Korišćenjem osobine parnosti hiberboličke sinusne funkcije i fomule $\sinh(ix) = i\sin(x)$ dobijamo:

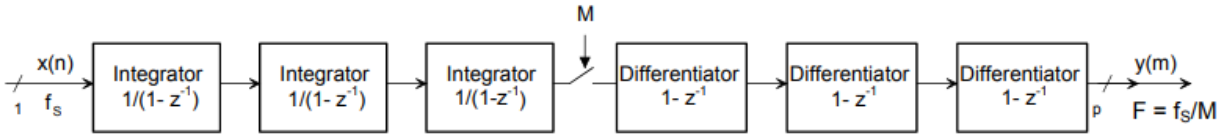
$$H(j\omega) = \frac{e^{-\frac{j\omega M}{2}}}{e^{-\frac{j\omega}{2}}} \frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} \tag{5.5.7}$$

Iz gore datog oblika funkcije lako se nalazi amplitudska karakteristika prenosne funkcije korišćenjem činjenica da je moduo proizvoda dva komplesna broja jednak modulu proizvoda i količnik modula kompleksnih brojeva jednak modulu količnika.

$$|H(j\omega)| = \left|\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right| \tag{5.5.7}$$

Na slici 5.5.3. se može videti ova amplitudska karakteristika obeležena sivom bojom.

Karakteristike mogu da se unaprede ukoliko se koristi sinc filtar višeg reda. Filter višeg reda se dobija kada se dodaju integratori i deferencijatori kao što je to prikazano na slici 5.4.2.



Slika 5.4.2. *Sinc filtar trećeg reda*

U opštem slučaju, prenosna funkcija *sinc* filtra k-tog reda je data izrazom:

$$H(z) = \left(\frac{1 - z^{-M}}{1 - z^{-1}}\right)^{k} \tag{5.5.8}$$

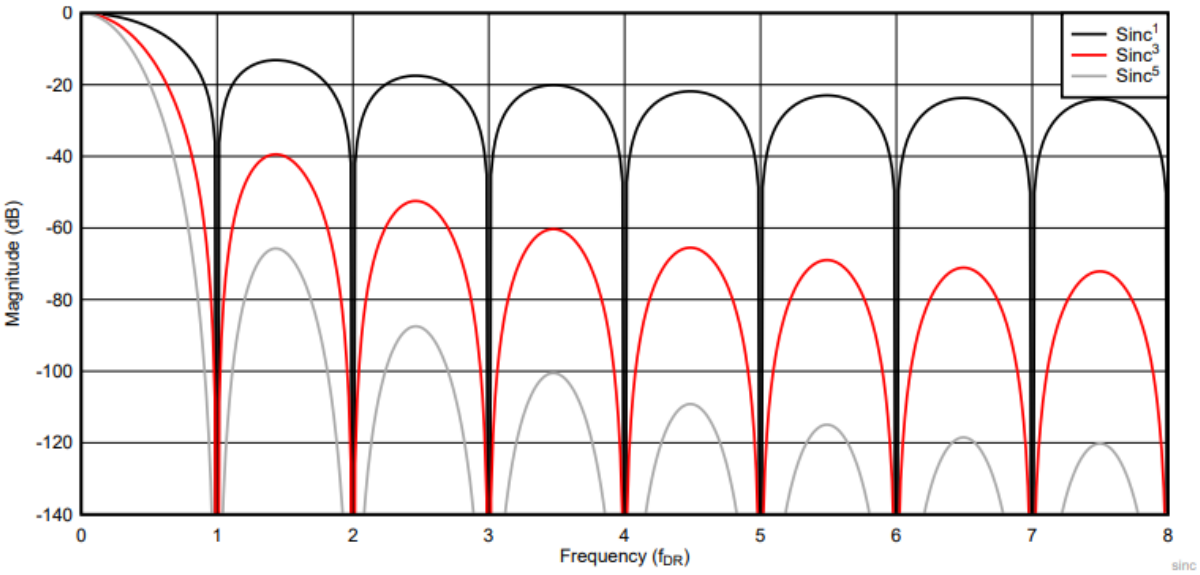
Odgovarajuća amplitudska karakteristika je:

$$|H(j\omega)| = \left|\frac{\sin\left(\frac{\omega M}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right|^{k} \tag{5.5.9}$$

Na slici 5.5.3. su prikazane amplitudske karakteristike sinc filtra prvog, trećeg i petog reda. Sa slike se jasno vidi da je i prilikom biranja reda sinc filtera potrebno praviti komporomise. Za filtar nižeg reda komponente na željenim učestanostima manje slabe, dok kod sinc filtra višeg reda dolazi i do slabljenja od oko 20 dB na Nikvistovoj učestanosti. Prednost filtra višeg reda je u tome što su komponente na učestanostima većoj od Nikvistove takođe više oslabljene pa je efekat preklapanja spektra manji.

Preporuka je da red sinc filtra bude za bar jedan veći od reda sigma-delta modulatora da bi se izbegla preklapanja u spektru.

Potrebno je pomenuti da i broj odbiraka koji se uzima takođe utiče na karakterisitiku filtra. Može se primetiti da nule amplitudske karakteristike predstavljaju umnoške frekvencije f_m/M. Pa je biranjem vrednosti M moguće pomerati ove nule.



Slika 5.5.3. *Amplitudska karakteristike sinc filtra prvog, trećeg i petog reda* (Texas Instruments)

5.6. Implementacija sinc filtra

Za implementaciju sinc filtra k-tog reda neophodno je definisati širine reči sa kojima rade komponente. Iako je podela filtra na integratore i diferencijatore zgodna za implementaciju, nije najjednostavnija za nalaženje širine izlaza filtra. Ovde je bolje prići problemu iz druge perspektive. Posmatrajmo *sinc* filtar prvog reda i pogledajmo definiciju u vremenskom domenu datom formulom 5.5.1. Ovaj put, ćemo smatrati da izlazni signal treba da bude u istom opsegu kao ulazni, i da su sve vrednosti dozvoljene. Tada je potrebno naći pravu aritmetičku sredinu signala, i izraz 5.5.1. će postati:

$$y[n] = \frac{1}{M}\sum_{i=0}^{M-1} x[n-i] \tag{5.6.1}$$

Odavde je prenosna funkcija u Z domenu data kao:

$$H(z) = \frac{1}{M}\frac{1-z^{-M}}{1-z^{-1}} \tag{5.6.2}$$

Obzirom da je širina signala sada svuda ista lako se nalazi prenosna funkcija za sinc filtar k-tog reda kao:

$$H(z) = \left(\frac{1}{M}\frac{1-z^{-M}}{1-z^{-1}}\right)^{k} \tag{5.6.3}$$

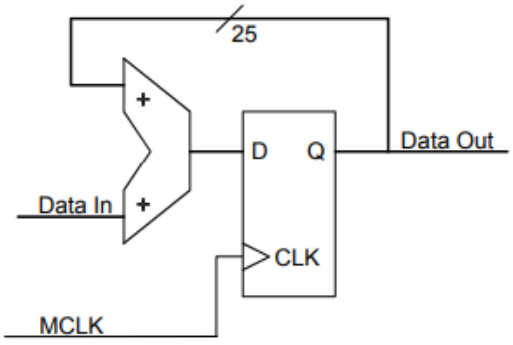
Sada se jasno vidi da je signal na izlazu smanjen M^k puta da bi bio smešten u opseg od 0 do 1. Prema tome, maksimalna vrednost koju može da ima je upravo M^k. Prema tome, broj bita potreban za predstavljanje izlaznog signala je:

$$B_{MAX} = [k \log_2 M] + 1 \tag{5.6.4}$$

Integratori i diferencijatori se implementiraju uz pomoć sabirača. Brojni sistem u kome se predstavlja rezultat je ili prirodni binarni kôd, ili komplement dvojke, pa važe osobine modularne aritmetike (3.3.2).

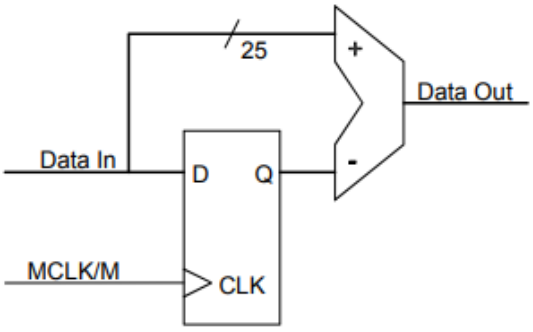
Prema tome, isti broj bitova potreban za izlazni signal je dovoljan i za sve međurezultate, odnosno, tolike širine reči treba da imaju svi integratori i diferencijatori koji čine ovaj filtar. Za konačni rezultat se vrši ekstenzija znaka za smeštanje u gore datu širinu.

Integrator i diferencijator se lako implementiraju u VLSI tehnologiji kao što je to prikazano na slikama 5.6.1. i 5.6.2.

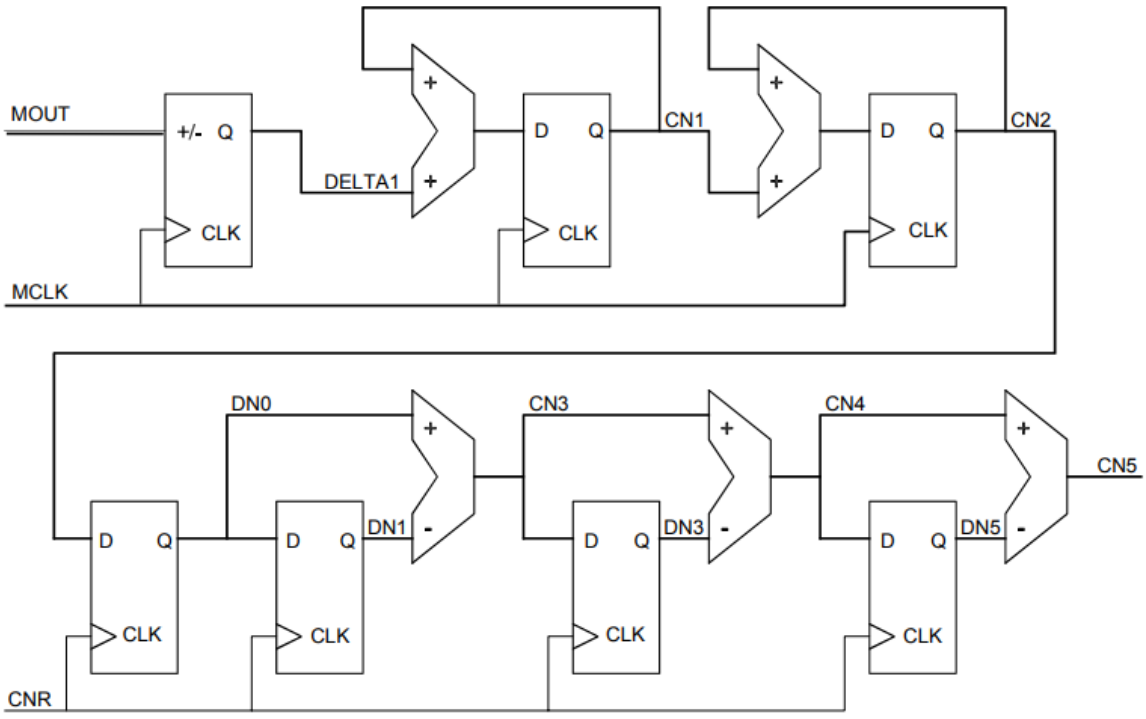


Slika 5.6.1. *Integrator sinc filtra (k=3, M=256)* (Texas Instruments, 2003)

Ceo filtar se potom veoma jednostavno implementira na osnovu blok šeme gde je decimacija signala urađena dovođenjem odgovarajućeg taktnog signala u diferencijatore. Implementacija celog sinc filtra trećeg reda je data na slici 5.6.3.



Slika 5.6.2. *Diferencijator sinc filtra (k = 3, M=256)* (Texas Instruments, 2003)



Slika 5.6.3*. Sinc filtar trećeg reda* (Texas Instruments, 2003)

5.6.1. Zaokruživanje i odbacivanje bitova

Iz formule 5.6.4. se naslućuje problem koji postoji sa direktnom implementacijom filtra datom na slici 5.6.3. Data formula ne garantuje da će širina reči biti deljiva sa 8, ili parna uopšte. Pošto su podaci u memoriji

predstavljeni bajtovima, potrebno je odbaciti određene bitove ili dobijeni broj zaokružiti, da bi se rezultat konverzije smestio u memoriju.

Neka je B_{MAX} broj bitova potreban za predstavljanje signala bez ikakvog zaokruživanja ili odbacivanja bitova kao što je dato u formuli 5.6.4. Pretpostavimo da se na ulazu svakog integratora i diferencijatora gube bitovi. Neka je sa j označen ulaz svakog elementa u kaskadnoj vezi, dakle ulaz prvog integratora sa 1, drugog sa 2, ..., i ulaz poslednjeg diferencijatora sa 6. Neka je i na izlazu poslednjeg diferencijatora još jedan element koji dodatno odbacuje bitove, pre upisa u izlazni registar. Ukupno ima $2k + 1$ stepena na kojima se odbacuju bitovi, odnosno, uvodi neka greška.

Neka je sa B_j označen broj bitova koji su odbačeni u j-tom stepenu (pri tome je to razlika ukupnog broja bitova i bitova koji se koriste na tom stepenu, dakle uključujući i bitove koji su odbačeni u prethodnim stepenima). Može se smatrati da su sve vrednosti pojednako verovatne i da samim tim i greška ima uniformnu raspodelu. Prema tome, širina ove uniformne raspodele je:

$$E_j = 2^{B_j} \tag{5.6.1.1}$$

Srednja vrednost greške zavisi od toga da li se bitovi odbacuju ili se radi zaokruživanje i data je izrazom:

$$\mu_j = \begin{cases} \frac{1}{2} E_j & odbacivanje \\ 0 & zaokruživanje \end{cases} \tag{5.6.1.2}$$

Varijansa je u oba slučaja ista i iznosi:

$$\sigma_j^2 = \frac{E_j^2}{12} \tag{5.6.1.3}$$

Svaka greška koja se uvede na j-tom nivu propagira do kraja sistema. Smatra se da su greške svakog nivoa nezavisne promenljive. Neka je sa $h[n]$ označen n-ti koeficijent impulsnog odziva j-tog stepena (odziv počev od j-tog stepena, uključujući njega samog, do kraja), tada se n-ti koeficijent odziva j-tog stepena može predstaviti izrazom:

$$y_j[n] = x\, h_j[n] \tag{5.6.1.4}$$

gde je x slučajna promenljiva koja predstavlja intenzitet greške opisana gore pomenutom uniformnom raspodelom. Kako je ovo filtar sa konačnim impulsnim odzivom, smatrajući da je greška po koeficijentima nezavisna, matematičko očekivanje ukupne greške je suma srednjih vrednosti grešaka po svim koeficijentima, pa ukupnu srednju vrednost greške stvorene na j-tom stepenu možemo da zapišemo kao:

$$\mu_{Tj} = \mu_j D_j \tag{5.6.1.5}$$

gde je D_j definisano kao suma svih koeficijenata impulsnog odziva j-tog stepena:

$$D_j = \begin{cases} \sum_n h_j[n], & j = 1, \dots, 2k \\ 1, & j = 2k+1 \end{cases} \tag{5.6.1.6}$$

Ukupna varijanasa se takođe nalazi suma varijansi po svim koeficijentima impulsnog odziva:

$$\sigma_{Tj} = \sigma_j^2 F_j^2 \tag{5.6.1.7}$$

gde je F_j dato kao:

$$F_j^2 = \begin{cases} \sum_n h_j^2[n], & j = 1, \dots, 2k \\ 1, & j = 2k+1 \end{cases} \tag{5.6.1.8}$$

Koeficijenti impulsnog odziva su dati sledećom jednačinom čije se izvođenje može naći u (Hogenauer, 1981):

$$h_j[n] = \begin{cases} \sum_{i=0}^{\left[\frac{n}{M}\right]} (-1)^i \binom{k}{i} \binom{Nk - j + n - Mi}{n - Mi}, & j = 1, \dots, k \\ (-1)^n \binom{2k + 1 - j}{n}, & j = k+1, \dots, 2k \end{cases} \tag{5.6.1.9}$$

Moguće je pokazati da je srednja vrednost greške svakog stepena osim prvog jednaka nuli, odnosno da važi:

$$D_j = \begin{cases} M^k, & j = 1 \\ 0, & j = 2, \dots, 2k \\ 1, & j = 2k+1 \end{cases} \tag{5.6.1.10}$$

Odavde proizilazi zanimljiv zaključak, a to je da srednja vrednost ukupne greške, osim za prvi i poslednji izvor, ne zavisi od toga da li se bitovi odbaciju ili zaokružuju.

Konačno, ukupnu srednju vrednost greške i varijansu možemo naći kao:

$$\mu_T = \sum_{j=1}^{2k+1} \mu_{Tj} = \mu_1 + \mu_{2k+1} \tag{5.6.1.11}$$

$$\sigma_T = \sum_{j=1}^{2k+1} \sigma_{Tj} \tag{5.6.1.12}$$

Ideja je da se greška koliko god je moguće ravnomerno rasporedi na svaki izvor greške, odnosno da varijansa greške svakog stepena ne bude veća od varijanse greške poslednjeg 2k+1 stepena. Ovaj uslov se može izraziti sledećom jednačinom:

$$\sigma_{Tj}^2 \leq \frac{1}{2k}\sigma_{T,2k+1}^2 , j = 1, \dots, 2k \tag{5.6.1.13}$$

Zamenom 5.6.1.1. , 5.6.1.2 i 5.6.1.7. u gornju jednačinu dobijamo:

$$\frac{1}{12}2^{2B_j}F_j^2 \leq \frac{1}{2k}\sigma_{T,2k+1}^2 , j = 1, \dots, 2k \tag{5.6.1.14}$$

Sređivanjem ove nejednačine i primenom logaritma dobija se rezultat:

$$B_j \leq -\log_2 F_j + \log_2 \sigma_{T,2k+1} + \frac{1}{2}\log_2 \frac{6}{k} \tag{5.6.1.15}$$

Ova formula govori koliko bitova treba odbaciti na svakom stepenu, pri tome se uzima najmanji ceo broj B koji zadovoljava datu nejednačinu.

Varijansu poslednjeg stepena nalazimo primenom formule za varijansu obzirom da je broj odbačenih bitova poslednjeg stepena razlika početnog broja bitova i veličine izlaznog registra.

6. Sigma-delta AD konvertor u MSP430i2041 mikrokontroleru

Mikrokontroler MSP430i2041 ima četiri nezavisna 24-bitna AD konvertora, odnosno kanala. Svi konvertori iz ove porodice mikrokontrolera su bazirani na sigma-delta modulatoru drugog reda i sinc, odnosno comb digitalnim filtrima. Ovi filtri mogu da imaju OSR (eng. oversampling ratio) od 32 do 256. Taktni signal na kome radi modulator je takođe isti za sve mikrokontrolere i iznosi 1024 MHz. Na osnovu formule 5.4.1. dobijamo frekvenciju odabiranja koja se kreće između 4 MHz, za OSR=256 i 32 MHz za OSR=32. Prema tome, gornje teoretske granice učestanosti signala koje se mogu meriti su 2-16 MHz[5]. Svi kanali mogu da koriste unutrašnji izvor referentnog napona od 1.2V, a moguće je i koristiti spoljašnji izvor referentnog napona. Konačno, svi kanali mogu da pristupe merenjima ugrađenog temperaturnog senzora.

6.1. Blok šema AD konvertora

Na slici 7.1.2. je prikazana blok šema AD konvertora, a na slici 7.1.1. blok šema koja prikazuje izvor referentnog napona.

Setovanjem signala SD24REFS kroz bafer se propušta unutrašnji referentni napon od 1.2V. U tom slučaju, poželjno je koristiti spoljašnji kondenzator od 100 nF koji je povezan između pina VREF i AVSS da bi se smanjio šum. Tada se ne sme priključiti bilo kakav izvor napona na VREF pin. Osim toga, po uključivanju referentnog napona, odnosno, promeni signala SD24REFS iz nule u jedinicu, neophodno je da prođe određeno vreme da se referentni napon stabilizuje. Ovo vreme je tipično 200 μs. Kada je signal SD24REFS resetovan da bi se koristio AD konvertor potrebno je obezbediti spoljašnji izvor napona.

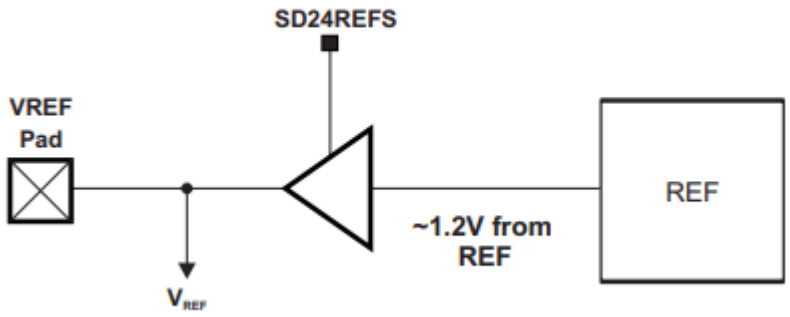
Signal SD24INCHx se koristi da se izabere izvor analognog napona. To mogu biti pinovi koji odgovaraju tom kanalu (kodovi od 000 do 101) ili signal senora za temperaturu (110). Kada se koriste pinovi napon koji se konvertuje je diferencijalni napon $V_{A+} - V_{A-}$ i ovaj napon se može kretati u opsegu $[-V_{FSR}, V_{FSR}]$, gde je

$$V_{FSR} = \frac{V_{REF}}{GAIN} \qquad (7.1.1.)$$

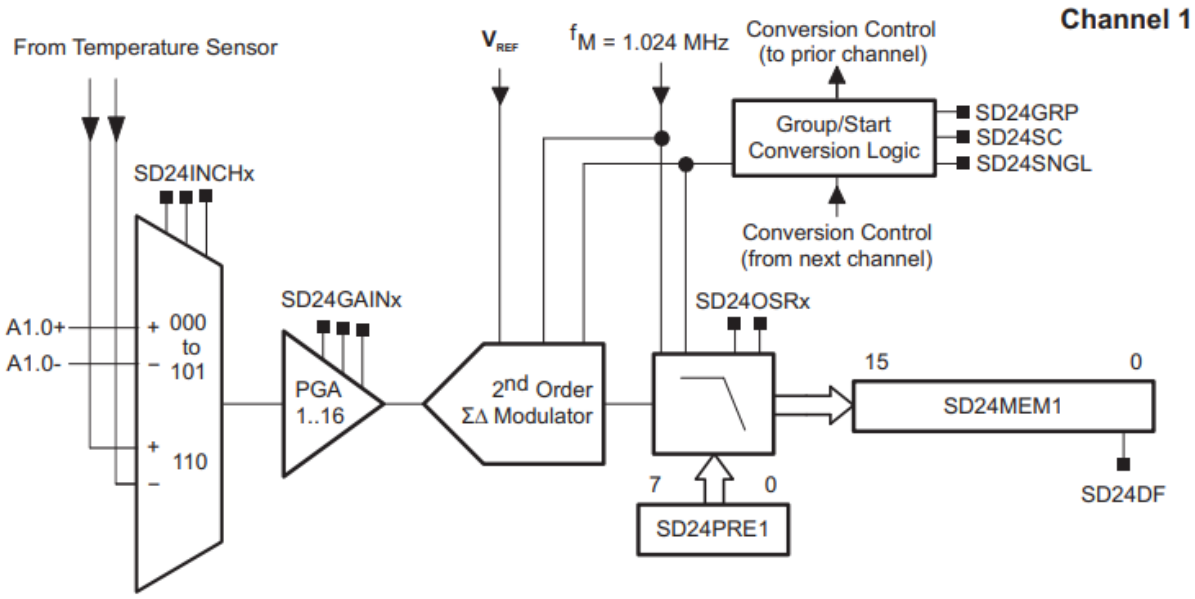
Signal SD2GAINx se koristi da se izabere pojačanje (GAIN) koje može biti 1,2,4,8 ili 16. Ipak, opseg diferencijalnog napona je realno manji od naznačenog, jer proivođač garantuje performanse samo kada je

[5] Na osnovu teoreme odabiranja. Kada se uzmu u obzir faktori kao što je slabljenje koje unosi sinc filtar, granice su strože.

apsolutna vrednost napona na pinu do 80% vrednosti napona V_{FSR}. Tako je, na primer, u slučaju kada se koristi referentni izvor napona i nema dodatnog pojačanja signala, realan opseg napona u kome konvertor radi onako kako je opisano, 928 mV.



Slika 6.1.1. *Biranje referentnog napona AD konvertora* (Texas Instruments, 2014)



Slika 6.1.2*. Blok šema AD konvertora* (Texas Instruments, 2014)

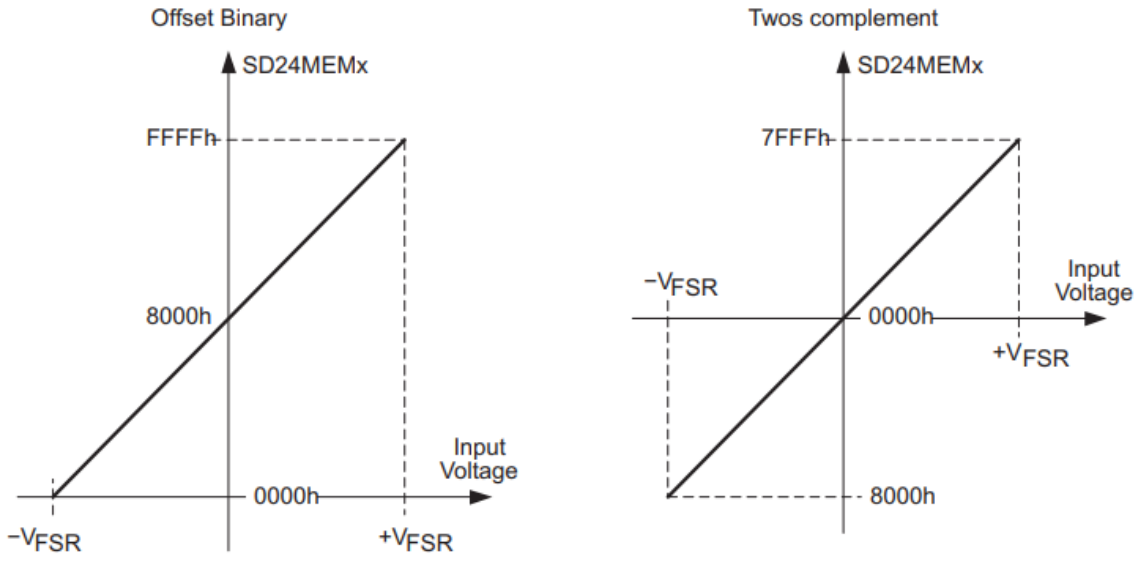
Analogni signal na izlazu pojačavača se prosleđuje sigma-delta modulatoru drugog reda. Ovaj modulator koristi izabrani referentni napon i radi na učestanosti 1.024 MHz. Izlaz ovog modulatora je povorka impulsa čija je srednja vrednost proporcionalna ulaznom analognom naponu. Ova povorka se prosleđuje digitalnom filtru koji propušta signal do učestanosti f_s koja zavisi od izabrane vrednosti za OSR. Signal SD24OSRx se koristi da se podesi faktor decimacije i on može da bude 256, 128, 64 ili 32. U zavisnosti od faktora

decimacije, granična učestanost digitalnog NF filtra se kreće od 4 kHz do 32 kHz. Ipak, smanjenje faktora decimacije dovodi do smanjenja broja bitova koji se koriste za predstavljanje signala. Tako, na osnovu formule 5.6.4. dobijamo da je broj bita potreban za predstavljanje punog rezultata 25 kada je OSR=256, dok je u slučaju OSR=32 dovoljno 16 bitova za predstavljanje podataka.

Formula 5.6.4. daje broj bitova potreban za predstavljanje podataka, ali u ovom konkretnom slučaju se ne koriste svi bitovi, već se najniži bit najverovatnije samo zaokružuje na parnu vrednost. Stoga su rezultati AD konvertora uvek za jedan bit manji od onog koji je dat formulom. Iako proizvođač to nije naveo, činjenica da je u slučaju kada je OSR=32 rezultat konverzije 15-bitna vrednost iako u registru ima mesta sa 16 bitova pokazuje da se ovo odbacivanje zaista vrši i to pre upisa u registar nezavisno od vrednosti koju ima signal SD24OSRx.

Signal SD24PREx ima smisla koristiti ako je u upotrebi više kanala za konverziju. Ovim signalom se obezbeđuje da konverzija odgovarajućeg kanala kasni određeni broj ciklusa. Naime, kada više kanala radi odjednom, bez obzira na to kada je startovana konverzija, rezultati su istovremeno upisani u njima odgovarajuće registre. Ovim signalom se obezbeđuje da se konverzije na kanalima ne dešavaju istovremeno.

Signal SD24DF služi da se podesi format rezultata konverzije – prirodni binarni sa ofsetom (SD24DF = 0) ili komplement dvojke (SD24DF = 1). Na slici 6.1.4. je prikazana veza između ulaznog signala i izlaza AD konvertora u zavisnosti od toga koji format je izabran.



Slika 6.1.4. *Ulazni napon i izlaz AD konvertora (levo: binarni sa ofsetom, desno: komplement dvojke)*

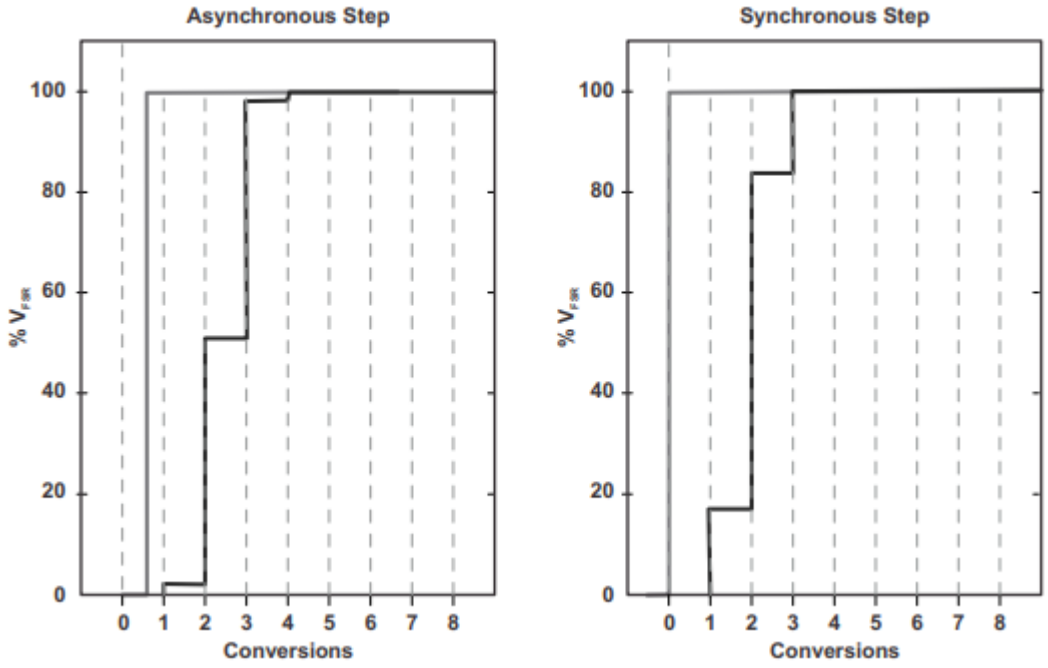
(Texas Instruments, 2014)

Signali SD24GRP i SD24SNGL se koriste da se izabere režim rada konvertora. Postoje četiri režima – jedan kanal i jedna konverzija, jedan kanal i kontinualna konverzija, više kanala koji se jednom konvertuju i više kanala koji se kontinualno konvertuju. Signal SD24SC se koristi da se započne konverzija.

6.2. Implementacija AD konvertora

Sam sigma-delta AD konvertor u ovom mikrokontroleru ima prilično jednostavnu implementaciju, iako ona može biti i mnogo kompleksnija od one objašnjene u prethodnom poglavlju. U dokumentaciji se navodi da je u osnovi AD konvertora jednobitni sigma-delta modulator drugog reda sa sinc digitalnim filtrom trećeg reda. Sigma-delta modulator, kao i sinc filtar trećeg reda su objašnjeni u prethodnom poglavlju i na osnovu onoga što je naveo proizvođač nema razloga da se pretpostavlja da se implementacija istih u ovom mikrokontroleru značajno razlikuje.

Nakon podešavanja parametara konverzije potrebno je da prođe određeno vreme pre nego što se rezultati konverzije mogu smatrati validnim. Kada se signal SD24INTDLY resetovan prvi prekid koji označava završenu konverziju se dešava tek nakon četvrte konverzije koja je u najgorem slučaju prva konverzija koja se može smatrati validnom. Na slici 6.2.1. je prikazan odziv konvertora na step pobudu.



Slika 6.2.1. *Odziv digitalnog filtra na step pobudu* (Texas Instruments, 2014)

U tabeli 6.2.1. je dat pregled širine podataka i učestanosti odabiranja za sve moguće vrednosti OSR faktora. Kao što je pomenuto u prethodnom odeljku, LSB bit konverzije se ne koristi.

OSR	fs=fm/OSR	Veličina rezultata
256	4 kHz	24 bit
128	8 kHz	21 bit
64	16 kHz	18 bit
32	32 kHz	15 bit

Tabela 6.2.1. *Učestanosti odabiranja i veličina rezultata u zavisnosti od vrednosti OSR faktora*

Konačno, MSP430 mikrokontroleri su mikrokontroleri smanjene potrošnje (eng. *low power*) pa u skladu sa tim imamo to da je taktni signal koji koristi modulator neaktivan kada nema konverzije.

6.3. Režimi rada AD konvertora

U odeljku 6.1. je rečeno da postoje četiri režima rada koji se podešavaju setovanjem, odnosno resetovanjem signala SD24SNGL i SD24GRP. U tabeli 6.3.1. je dat pregled ovih režima i vrednosti kontrolnih signala koje im odgovaraju.

SD24SNGL	SD24GRP	Režim
0	0	Jedan kanal, jedna konverzija
1	0	Jedan kanal, kontinualna konverzija
0	1	Grupa kanala, jedna konverzija
1	1	Grupa kanala, kontinualna konverzija

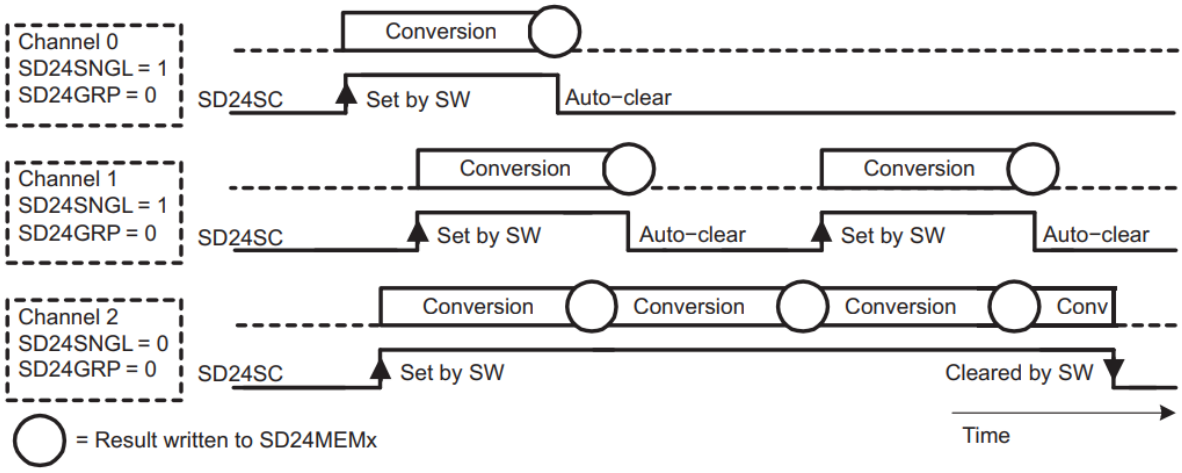
Tabela 6.3.1. *Režimi rada AD konvertora i vrednosti kontrolnih signala koje im odgovaraju*

Signalom SD24SC se startuje konverzija. Ukoliko je signal SD24SNGL bio aktivan, a SD24GRP resetovan, radi se samo jedna konverzija i bit SD24SC se automatski resetuje po završenoj konverziji. Ukoliko se ovaj signal resetuje pre završetka konverzije kanal se gasi zajedno sa digitalnim filtrom i vrednost u izlaznom registru je nedefinisana.

Ukoliko se promeni vrednost signala SD24SNGL u 0 radi se kontinualna konverzija na jednom kanalu. Konverzija počinje kada se aktivira SD24SC i traje dok se isti signal ne resetuje. Kao i u prethodnom slučaju, ne garantuje se da je u izlaznom registru validna vrednost ukoliko se ovaj bit resetuje pre završetka konverzije. Odnosno, treba smatrati validnim samo one izlzne podatke koji su se nalazili u ovom registru pre resetovanja ovog bita.

Na slici 6.3.1. su prikazani načini na koje se može vršiti konverzija na jednom kanalu. Na gornjem dijagramu se vidi kada se na jednom kanalu radi jedna konverzija. Na dijagramu u sredini je prikazana situacija u kojoj je takođe podešeno da se radi jedna konverzija, ali se koristi za prikupljanje više odbiraka signala. Tada se za svaki odbirak softverom startuje konverzija. Na donjem dijagramu je prikazana konverzija u slučaju u kome je izabrana kontinualna konverzija.. Ukoliko je potreban veći bro odbiraka i softverom se aktivira svaka sledeća konverzija ne mora da znači da će odabiranje signala da se radi uniformno i tu treba na odgovarajući način napraviti softver.

Setovanjem signala SD24GRP se aktivira grupna konverzija. Kanali se grupišu tako što se u grupu ubacuje kanal koji se nalazi posle onog kanala čiji je signal SD24GRP setovan. Tako, na primer, ako se ovaj signal postavi na jedinicu na nultom kanalu, a ima vrednost 0 na kanalu 1, grupu čine kanali 0 i 1, pri čemu je kanal 1 master. Signal SD24GRP poslednjeg kanala nema funkciju i uvek je resetovan. Master kanal je kanal čiji SD24SC signal startuje konverziju na svim kanalima.

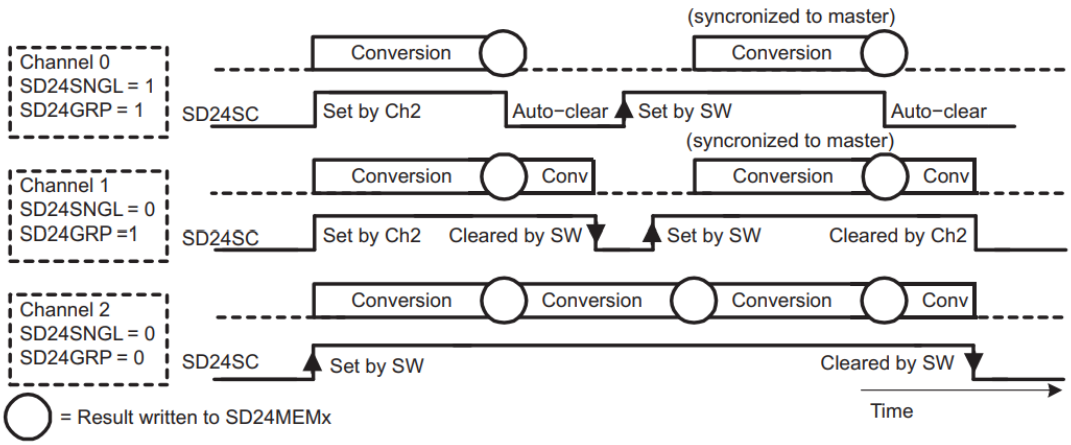


Slika 6.3.1. *Konverziije jednog kanala*

Kada je signal SD24SNGL jednak jedinici na kanalu u grupi radi se jedna konverzija. Konverzija automatski počinje onda kada se SD24SC signal na master kanalu aktivira. Konverzija se ipak može završavati nezavisno od master kanala softverom tako što se SD24SC signal deaktivira za određeni kanal u grupi.

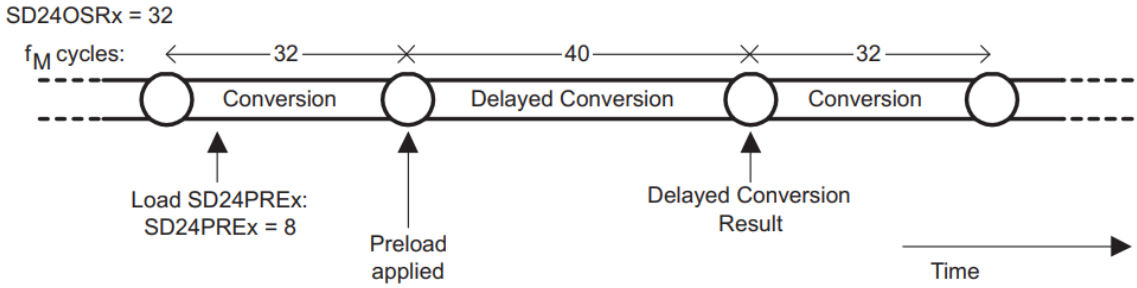
Ukoliko je signal SD24SNGL jednak nuli, taj kanal radi kontinualnu konverziju. Ova konverzija je sinhronizovana sa konverzijom master kanala. Ukoliko se signal SD24SC određenog kanala softverom aktivira on će se i tada sinhronizovati sa SD24SC signalom master kanala..

Na slici 6.3.2. su prikazani vremenski dijagrami grupne konverzije.U grupnoj konverziji na slici učestvuju prva tri kanala, i kanal 2 je master kanal. Kanal 0 radi jednu konverziju, dok je kod kanala 2 izabrana kontinualna konverzija signala. Signal SD24SC master kanala se aktivira na početku i briše na kraju. U međuvremenu se SD24SC0 briše nakon završene konverzije i ponovo setuje softverom. Kao što se vidi sa slike, konverzija ne počinje odmah po setovanju ovog kanala već onda kada počne sledeća konverzija na master kanalu. Ovaj signal se potom briše po završetku te jedne konverzije. Sa druge strane, kanal 1 vrši kontinualnu konverziju i u toku druge konverzije je signal SD24SC1 ugašen od strane softvera čime je konverzija odmah zaustavljena. Nakon toga je softverski ponovo dat signal za početak konverzije i kao u prethodnom slučaju konverzija se sinhronizuje sa konverzijom na master kanalu. Kada je signal SD24SC2 deaktiviran ovo zaustavlja i konverziju na kanalu 1.



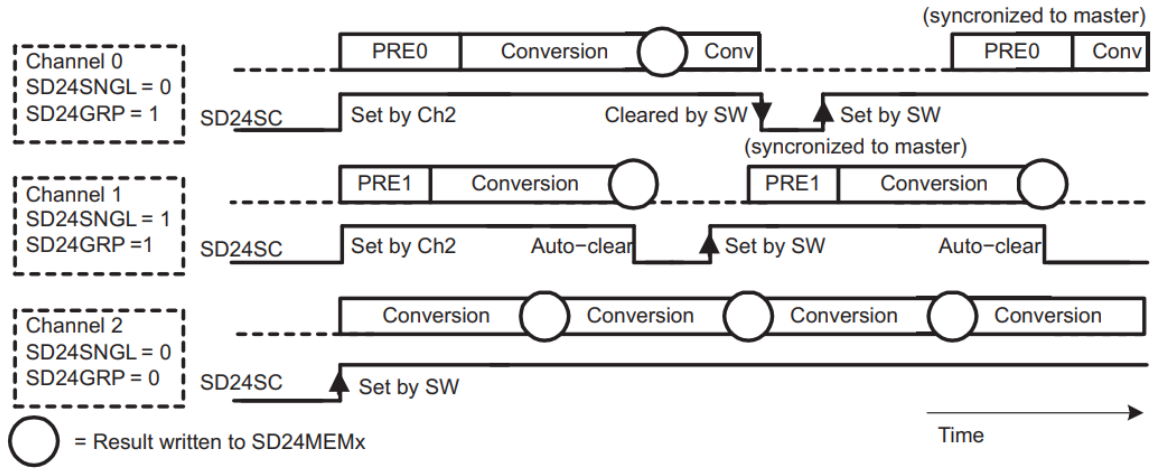
Slika 6.3.2. *Grupna konverzija*

Signal SD24PREx je već pomenut u odeljku 6.1. kao način da se konverzija zakasni za nekoliko ciklusa modulatora. Upisom vrednosti od 0-255 u registar SD24PREx se za isti broj ciklusa može zakasniti konverzija na datom kanalu. Kašnjenje se dešava u prvom ciklusu konverzije nakon upisa u ovaj registar kao što je prikazano na slici 6.3.3.



Slika 6.3.3. *Pomeranje konverzije upisom u SD24PREx registar*

Ovo kašnjenje se dešava svaki put na početku konverzije (aktiviranjem SD24SC signala). Ukoliko se radi o grupnoj konverziji preporuka je da master kanal nema kašnjenje da bi se lakše pratili ostali kanali obzirom da se konverzije sinhronizuju prema glavnom kanalu. Ova situacija je prikazana na slici 6.3.4. Predstavljena je grupna konverzija u kojoj učestvuju nulti, prvi i drugi kanal, pri čemu je glavni kanal kanal broj 2. Na nultom i drugom kanalu se vrši kontinualna, a na prvom jedna konverzija. Kanal 0 ima kašnjenje PRE0, dok je kašnjenje kod kanala 1 PRE1. Setovanjem signala SD24SC2 počinje konverzija i kao što se vidi sa dijagrama ona se na nultom i prvom kanalu dešava nakon upisanog zakašnjenja. Kao i u prethodnom slučaju, ako se na kanalima 0 i 1 zaustavi konverzija, a potom ponovo započne ona je ponovo sinhronizovana sa prvom narednom konverzijom master kanala. U ovom slučaju, razlika je u tome što ukoliko nije došlo do promene SD24PREx signala, ponovo konverzija biva zakašnjena za istu vrednost.



Slika 6.3.4. *Grupna konverzija sa kašnjenjem* (Texas Instruments, 2014)

6.4. Generisanje prekida i dohvatanje rezultata konverzije

Svaki kanal, odnosno AD konvertor ima dva izvora prekida SD24IFG i SD24OVIFG. Oba signala se nalaze u SD24CCTLx regitru odgovarajućeg kanala. Signal SD24IFG se aktivira kada se u izlazni registar AD konvertora upiše rezultat konverzije. Ukoliko je dozvoljen prekid, završetak konverzije će ga aktivirati. Ovaj signal se automatski resetuje kada se pročita vrednost iz izlaznog registra, mada se može i ručno obrisati softverom. Signal SD24OVIFG označava da je došlo do upisivanja novog rezultata konverzije u izlazni registar pre nego što je stara vrednost bila pročitana. Ovaj signal se ne resetuje automatski već je neophodno to uradti softverom.

Svi kanali su ukombinovani u jedan vektor u tabeli prekida. U registru SD24IV se tada nalazi vrednost prekida najvišeg prioriteta. Pri tome najveći prioritet ima prekid usled propuštene vrednosti (SD24OVIFG).

Kada se javi ovaj prekid neophodno je proveriti SD24CCTLx registre aktivnih kanala da bi se odredilo gde je sve došlo do ovog problema. Ukoliko je postojalo više prekida AD konvertora, nakon što se opsluži prekid najvišeg prioriteta, instrukcijom RETI će se obrisati vrednost u SD24IV registru i sada će se ponovo aktivirati prekid, ali će u SD24IV registru bit kod onog prekida koji je došao na red.

Vrednost koja se nalazi u SD24IV registru se može iskoristiti kao pomeraj kao što je to prikazano na slici 6.4.1. Na ovaj način se određivanje toga odakle je došao prekid i odlazak na odgovarajuću obradu značajno skraćuje[6].

```
; Interrupt handler for SD24.

INT_SD24                ; Enter Interrupt Service Routine   6
   ADD   &SD24IV,PC     ; Add offset to PC                  3
   RETI                 ; Vector 0: No interrupt            5
   JMP   ADOV           ; Vector 2: ADC overflow            2
   JMP   ADM0           ; Vector 4: CH_0 SD24IFG            2
   JMP   ADM1           ; Vector 6: CH_1 SD24IFG            2
;
; Handler for CH_2 SD24IFG starts here. No JMP required.
;
ADM2     MOV   &SD24MEM2,xxx   ; Move result, flag is reset
         ...                   ; Other instruction needed?
         JMP   INT_SD24        ; Check other int pending     2
;
; Remaining Handlers
;
ADM1     MOV   &SD24MEM1,xxx  ; Move result, flag is reset
         ...                  ; Other instruction needed?
         RETI                 ; Return                      5
;
ADM0     MOV   &SD24MEM0,xxx  ; Move result, flag is reset
         RETI                 ; Return                      5
```

Slika 6.4.1. *Metod za proveru i opsluživanje odgovarajućeg prekida koji predlaže proizvođač* (Texas Instruments, 2014)

Jako je bitno napomenuti da se SD24IFG signal automatski resetuje tek prilikom čitanja izlaznog registra. Ukoliko to nije urađeno, ako se reset ne obavi softverski, ovaj prekid će biti aktivan ponovo po izlasku iz prekidne rutine.

U vezi sa prekidom je i signal SD24INTDLY koji je podrazumevano nula i u takvom stanju prve četiri konverzije neće izazvati prekid pošto je to broj ciklusa potreban da se stabilizuje odziv digitalnog filtra (slika 6.2.1).

Izlazni registar kanala je označen sa SD24MEMx i širine je 24 bita. Ovaj registar se samo čita i pristupa se celoj reči. U slučajevima kada su rezultati konverzije veći od 16 bita koriste se signali SD24LSBACC i SD24LSBTOG da se izabere odgovarajućih 16 bitova kao što je to prikazano na slici 6.4.2. Signal

[6] Ovakva obrada prekida je moguća kod svake komponente.

SD24LSBACC služi da se izabere pristup višim ili nižim delovima rezultata. Signal SD24LSBTOG se koristi da se vrednost signala SD24LSBACC automatski menja prilikom svakog čitanja izlaznog registra.

Slika 6.4.2. *Širine rezultata konverzije za različite vrednosti OSR faktora i njihovo čitanje* (Texas Instruments, 2014)

6.5. Pregled registara AD konvertora

Svaki kanal ima svoj kontrolni registar SD24CCTLx, izlazni registar SD24MEMx, registar za kontrolu ulaza SD24INCTLx, i registar u koji se upisuje evetualno kašnjenje konverzije SD24INCTLx. Pored ovih registara koji su karakteristični za pojedinačne kanale postoje i tri registra za kontrolu AD konvertora uopšte. To su kontrolni registar SD24CTL, registar sa brojem prekida SD24IV i trim registar SD24TRIM.

U tabelama ispod je dat pregled ovih registara i signala u njima.

Naziv signala	Opis signala
SD24REFS	Biranje referentnog napona
SD24OVIE	Dozvola *overflow* prekida

Tabela 6.5.1. *SD24CTL registar (16 bit, read/write)*

Naziv signala	Opis signala
SD24IVx	Izvor prekida

Tabela 6.5.2. *SD24IV registar (16 bit, read)*

Naziv signala	Opis signala
SD24REFS	Biranje referentnog napona
SD24OVIE	Dozvola *overflow* prekida

Tabela 6.5.1. *SD24CTL registar (16 bit, read/write)*

Naziv signala	Opis signala
SD24INTDLY	Ignorisanje prvih 4 konverzija pri generisanju prekida
SD24GAINx	Pojačanje diferencijalnog pojačavača
SD24INCHx	Dovođenje izlaza temperaturnog senzora na ulaz kanala

Tabela 6.5.4. SD24INCTLx *registar (16 bit, read/write)*

Naziv signala	Opis signala
ConversionResults	Rezultat konverzije

Tabela 6.5.5. *SD24MEMx registar (16 bit, read)*

Naziv signala	Opis signala
SD24SNGL	Jedna ili kontinualna konverzija
SD24OSRx	Biranje frekvencije decimacije
SD24LSBTOG	Biranje automatske promene LSBACC bita nakon čitanja izaznog registra
SD24LSBACC	Biranje najnižih 16 bitova rezultata
SD24OVIFG	Overflow prekid fleg
SD24DF	Format podataka (binarni sa ofsetom, komplement dvojke)
SD24IE	Dozvola prekida
SD24IFG	Fleg koji označava da se desio prekid
SD24SC	Signal kojim se započinje konverzija
SD24GRP	Signal kojim se ovaj kanal grupiše sa narednim kanalom

Tabela 6.5.6. SD24CCTLx *registar (16 bit, read/write)*

Naziv signala	Opis signala
PreloadValue	Kašnjenje konverzije

Tabela 6.5.5. SD24PREx *registar (16 bit, read/write)*

6.6. Karakteristike AD konvertora

U tabeli 6.6.1. je dat pregled karakteristika koje daje proizvođač.

Parametar	Uslovi	Ulazni napon	Tipične vrednosti (dB)
SINAD	SD24REFS = 1, SD24OSRx = 256, SD24GAINx = 1, VCC = 3V	$928mV\sin(2\pi 50t)$	89
THD			100
SFDR			100
INL			-0.003, 0.003

Tabela 6.6.1. *Dinamičke karakteristike AD konvertora koje daje Texas Instruments*

7. Testiranje sigma-delta AD konvertora

Testiranje sigma-delta AD konvertora koji se nalazi u mikrokontroleru MSP430i2041 je se svodi na ispitivanje rada konvertora u situacijama kada se na njegov ulaz dovedu različiti oblici periodičnih signala koji su generisani uz pomoć generatora signala.

Projekat se sastoji iz dve celine – programa za mikrokontroler i programa koji vrši obradu podataka na računaru. Osnovna ideja je da se na ulaz AD konvertora dovede izlaz generatora signala. Rezultati konverzije se upisuju u kružni bafer i potom preko UART-a šalju računaru. Podaci koje računar prikuplja se u realnom vremenu prikazuju u vremenskom i frekvencijskom domenu. Osim toga, moguće je snimanje podataka koji stižu fajl i na osnovu upisanih podataka računanje parametara AD konvertora.

Za projektovanje softvera mikrokontrolera je korišćen *Code Composer Studio,* i program je napisan u programskom jeziku C. Za obradu podataka na računaru je korišćen *Matlab*, pri čemu je deo programa napisan u C++-u i on se poziva u okviru *Matlab* programa.

7.1. Program mikrokontrolera

Program mikrokontrolera radi u *low-power* režimu rada i sve aktivnosti se dešavaju unutar prekidnih rutina. Na početku programa su inicijalizovani AD konvertor i UART.

Inicijalizacija UART serijske komunikacije je urađena po algoritmu opisanom u poglavlju 2.1.2. tako da baud rate bude 9600, a kao izvor takta je iskorišćen SMCLK. Na početku nije dozvoljen UART prekid.

Inicijalizacija AD konvertora podrazumeva biranje unutrašnjeg izvora referentnog napona, dozvole prekida na kanalu 0, podešavanje OSR faktora na 256 čime se bira frekvencija odabiranja od 4 kHz i dobija 24-bitni rezultat. Rezultat se predstavlja u komplementu dvojke. Nakon postavljenih podešavanja obezbeđen je i određen broj praznih ciklusa da bi se obezbedilo da se unutrašnji izvor napona stabilizuje.

Nakon inicijalizacije obe periferije se startuje AD konvertor i mikrokontroler ulazi u režim smanjene potrošnje i sve dalje aktivnosti se događaju unutar prekidnih rutina.

Postoji jedan bafer koji može da sačuva 512 24-bitnih rezultata konverzije (u RAM memoriji nema dovoljno mesta za čuvanje većeg broja odbiraka). Bafer je predstavljen kao matrica sa 3 kolone gde svaka kolona sadrži jedan bajt rezultata, pri čemu je u nultoj koloni najniži bajt. Obe periferije imaju svoj pokazivač koji govori o tome dokle su stigle sa upisom, odnosno sa čitanjem, dok UART poseduje i pokazivač na trenutnu kolonu, obzirom da se šalju poruke dužine 8 bitova.

Kada AD konvertor napuni bafer dozvoljava UART prekide i maskira svoje. Isto tako, kada UART pošalje svih 512 rezultatata konverzije, on maskira svoj prekid i ponovo aktivira AD konvertor. Na ovaj način se dobija 512 uzastopnih odbiraka konverzije koje računar može grupno da obradi.

Razmatrane su i drugačije realizacije koje bi eventualno obezbedile veći broj odbiraka u jednoj grupi, ali zbog toga što je serijska komunikacija značajno sporija od AD konvertora, ovako nešto nije moglo da se izvede bez dodatnog odabiranja signala koje bi graničnu učestanost, to jest dinamički opseg, smanjilo bar 8 puta.

7.2. Program za obradu podataka na računaru

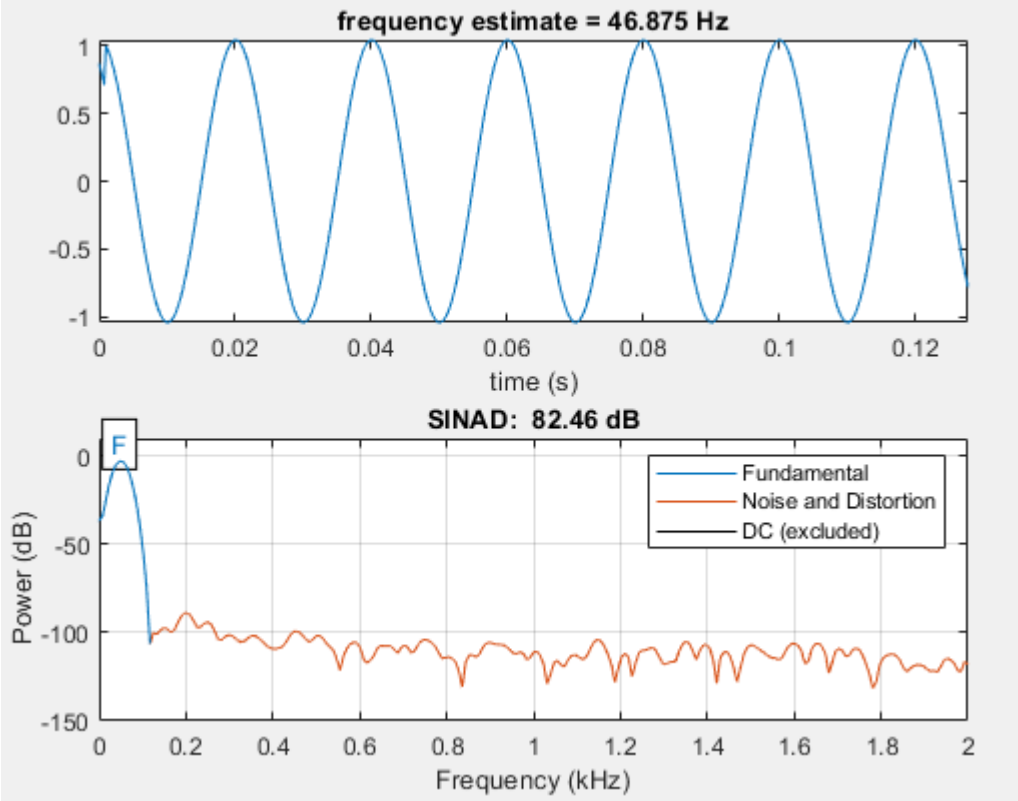
Obrada podataka na računaru se može podeliti na dve celine. Jedan deo obrade se izvršava u funkciji koja prima podatke preko UART-a i prikazuje ih u realnom vremenu, s tim da je omogućeno upisivanje podataka u fajl. Drugi deo obrade se svodi na obradu podataka koji su upisani u fajl.

Funkcija koja koja se poziva na početku dobija ime fajla kao argument. Ovo je naziv fajla u koji treba upisati podatke. Ova funkcija uspostavlja serijsku komunikaciju sa mikrokontrolerom i kreira figuru na kojoj će biti prikazani podaci. Kao što je pomenuto u prethodnom poglavlju, na raspolaganju je uvek 512 odbiraka, pa se sa svakom novom grupom ponovo iscrtavaju rezultati. Ova operacija se izvršava brže nego što je AD konvertoru potrebno da prikupi novih 512 podataka, pa nema gubljenja odbiraka (prema tome, ovakva implementacija je osetljiva na to koliko brzo radi računar). Figura se sastoji iz dva grafika – prikaz odbiraka u vremenskom domenu i prikaza u frekvencijkom domenu (prikazuje se snaga signala u decibelima). Ujedno se automatski detektuje maksimalna učestanost u spektru i prikazuje procenjena frekvencija sinusoide kao i SINAD vrednost. Obe vrednosti se računaju na osnovu poslednjih 512 podataka. Na slici 7.2.1. se može videti izgled ovog programa.

Kada je kursor na figuri, klikom na *space* se mogu sačuvati trenutni podaci. Ovi podaci se čuvaju u posebnom nizu i moguće je sačuvati više ovakvih “slika”. Program se završava kada se zatvori figura i tada se zatvara i serijski port, a sačuvani podaci upisuju u fajl. Ideja je da se maksimalno olakša testiranje i omogući čuvanje upravo onih podataka koji su bitni i reprezentativni, a da program pritom nesmetano radi.

Kao što je već rečeno, UART šalje bajtove poruke redom, počev od najnižeg bajta. Program čeka da stignu tri bajta i kombinuje ih u jedan 24-bitni podatak. Iako je ovakav način slanja podataka jako efikasan, problem je u tome što se računar i mikrokontroler moraju nekako sinhronizovati. Ovo je obezbeđeno time što računar šalje mikrokontroleru podatak preko UART-a, čime označava da je spreman za prijem narednih 512 odbiraka. Sa druge strane, kada AD konvertor završi sa konverzijom, on aktivira UART TX prekid ukoliko je prethodno računar javio da je spreman. U suprotnom, AD konvertor označava da je završio i

blokira svoje prekide, i tek unutar RX prekidne rutine kada računar javi da je spreman, može da se aktivira TX prekid.



Slika 7.2.1. *Program koji prikazuje podatke u realnom vremenu*

Oblikovanje pristiglih bajtova u jedan 32-bitni podatak je urađeno u funkciji koja je implementirana u C++ programskom jeziku obzirom da je ovakva manipulacija bitovima mnogo lakša u jeziku niskog nivoa nego u *Matlab*-u.

Drugi deo programa se koristi za obradu podataka koji su upisani u fajl. Obrada podataka podrazumeva određivanje dinamičkih karakteristika AD konvertora.

Dinamičke karakteristike se određuju na osnovu rezultata koji se dobiju dovođenjem sinusoidalnog signala na ulaz konvertora. Ove karakteristike se računaju za različite vrednosti frekvencije ulaznog signala. *Matlab* pruža odličnu podršku za računanje parametara kao što su SNR, THD, SINAD i SFDR. Ove funkcije, osim što računaju date parametre (sa pretpostavkom da je ulazni signal sinusoida) mogu i da prikažu grafik na kome je jasno obeleženo šta je uzeto u obzir prilikom njihovog izračunavanja.

Zanimljivo je bilo i proveriti šta se dešava u vremenskom domenu. Za ovu svrhu je na osnovu datih odbiraka određena *best fit* sinusoida i posmatrana je razlika između merenog i "originalnog" signala. Na osnovu ove razlike se može pronaći raspodela greške kvantizacije i njen spektar.

7.3. Rezultati testiranja

U ovom odeljku je dat pregled rezultata testiranja. U prvom odeljku su prikazani rezultati testiranja dinamičkih karakteristika. U drugom odeljku je data analiza greške kvantizacije.

7.3.1. Analiza dinamičkih karakteristika

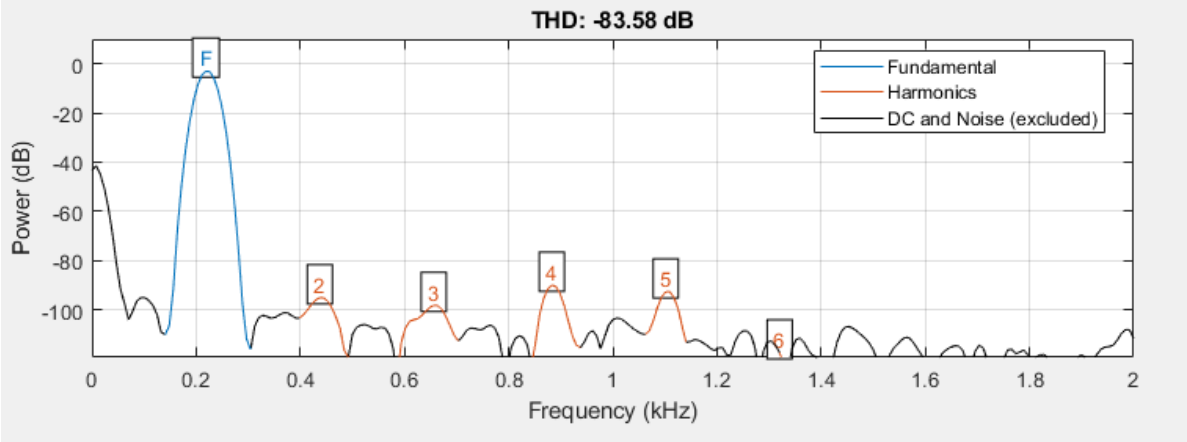
U tabeli 7.3.1.1. je dat pregled određenih parametara za različite učestanosti ulaznog signala kao i ENOB koji je određen na osnovu formule:

$$ENOB = \frac{SINAD - 1.76\,\text{dB} + 20\,\log\left(\frac{V_{FSR}}{V_{IN}}\right)}{6.02\,\text{dB}} \tag{7.3.1}$$

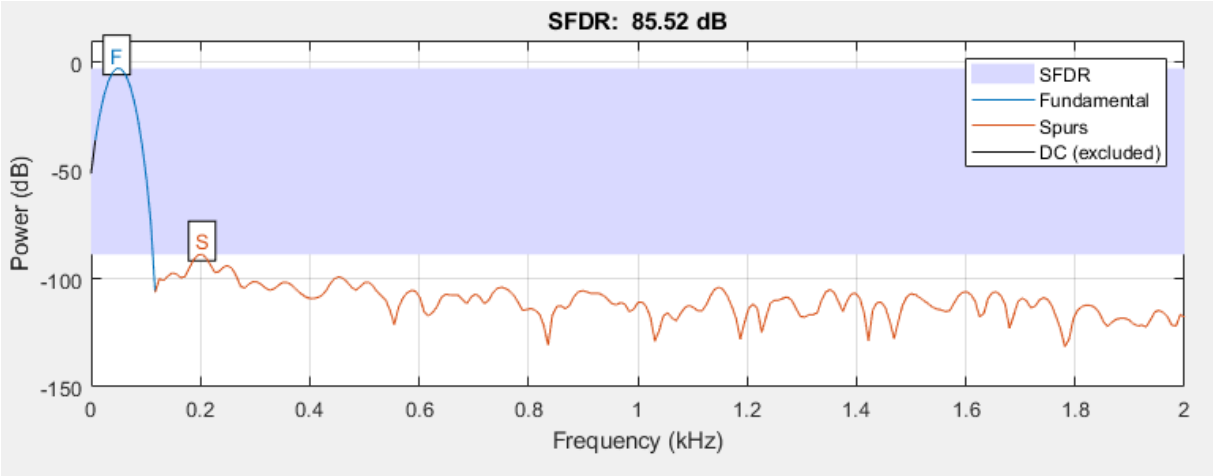
frekvencija	SINAD [dB]	THD [dB]	SFDR [dB]	ENOB
25 Hz	82.76	-3.79	51.88	13.72
50 Hz	81.89	-20.72	58.93	13.57
221 Hz	83.12	-36.71	20.92	13.79
555 Hz	83.06	-39.35	28.27	13.77
870 Hz	80.41	-49.28	39.50	13.33
1 kHz	80.36	-inf	31.07	13.32
1.5 kHz	79.29	-inf	37.37	13.14
1.8 kHz	74.46	-inf	41.08	12.34

Tabela 7.3.1.1. *Dinamčki parametri*

Na slikama 7.3.1.1. i 7.3.1.2. se mogu videti primeri određivanja parametara THD i SFDR gde je prikazano koje se komponente signala uzimaju prilikom njihovog izračunavanja. Kao što se i vidi sa slike, za računanje totalnog harmonijskog izobličenja se koristi prvih šest harmonika.



Slika 7.3.1.1. *Primer određivanja THD parametra (f =221 Hz)*



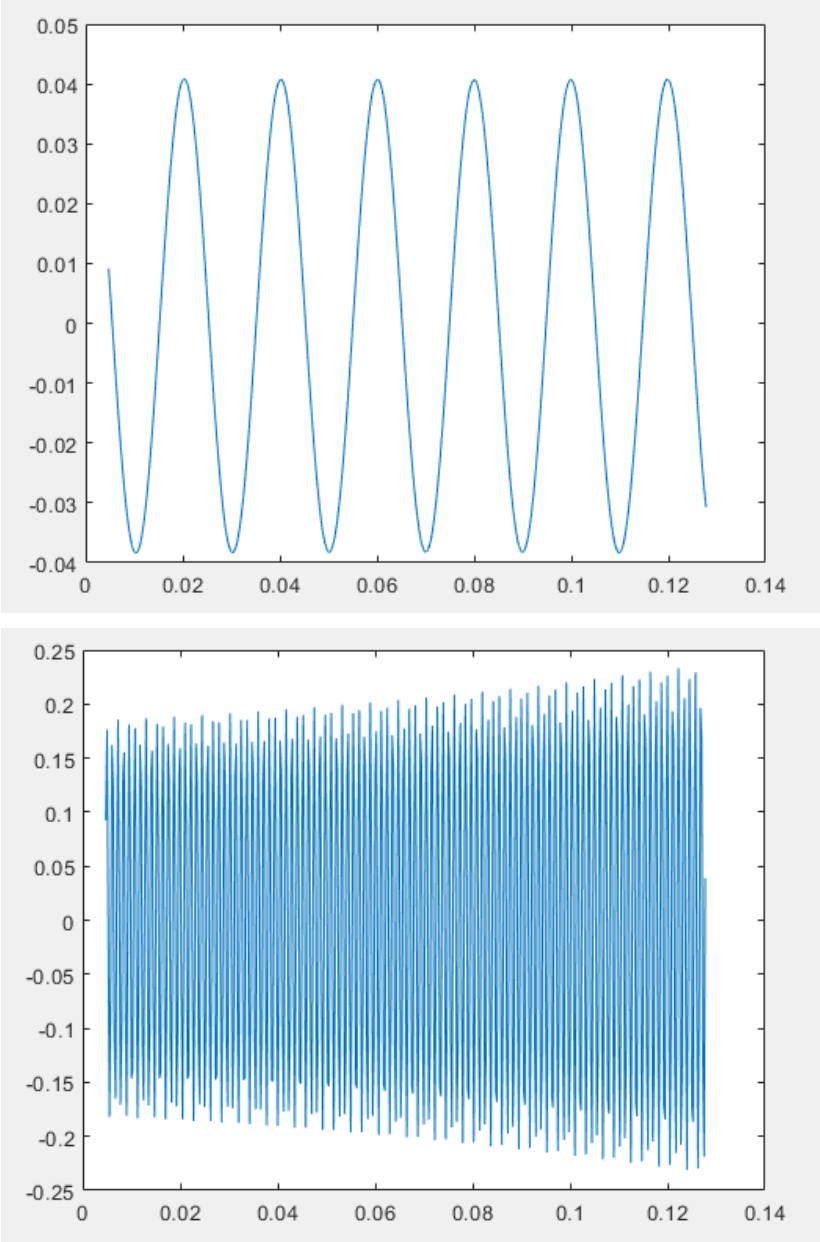
Slika 7.3.1.2. *Primer određivanja SFDR parametra (f=50 Hz)*

7.3.2. Analiza greške kvantizacije

Greška kvantizacije se detaljnije može analizirati tako što se mereni signal poredi sa originalnim. Kao originalni signal je korišćena sinusoida koja ima istu frekvenciju odabiranja kao i merena, sa frekvencijom i amplitudom koje su zadate generatoru. Za određivanje faze je korišćen *Matlab* paket za fitovanje podataka *Curve Fitting Tool*.

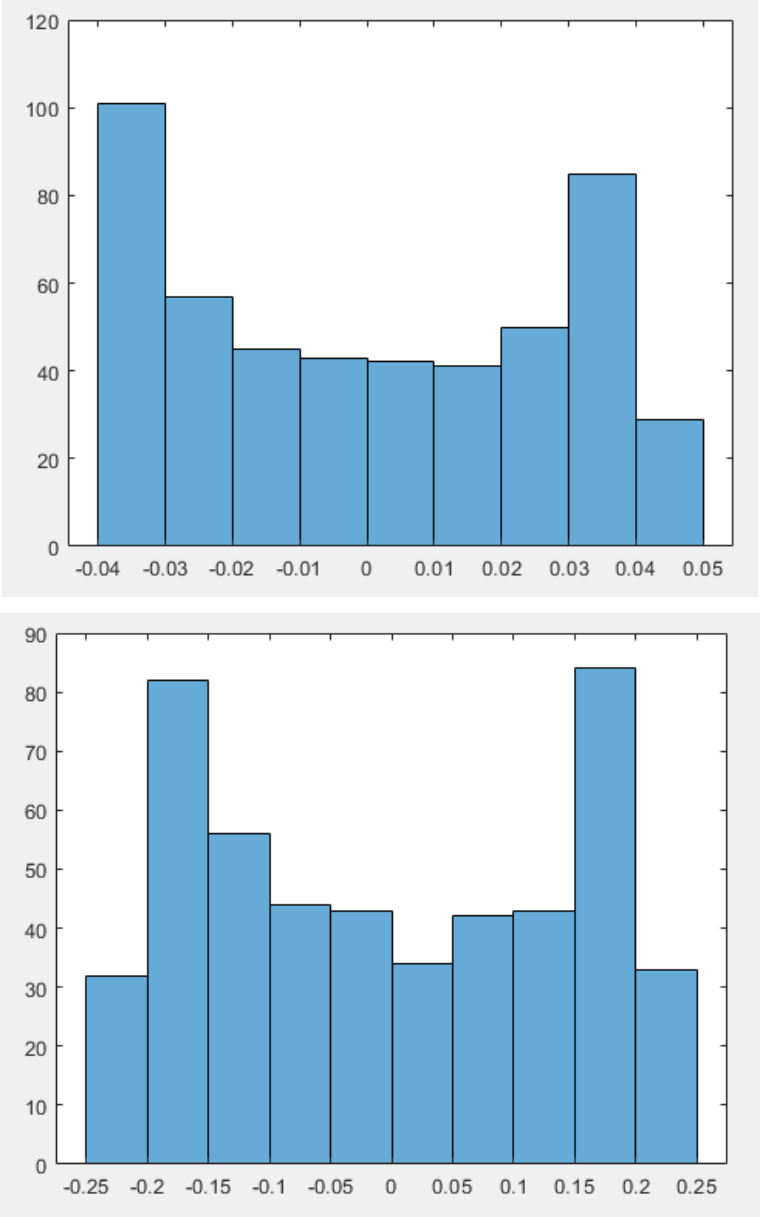
Određena je razlika između ova dva signala i ona predstavlja grešku kvantizacije. Prikazana je njena zavisnost u vremenu koja pokazuje eventualnu korelaciju između greške kvantizacije i ulaznog signala, određena je njena raspodela i prikazana je njena snaga u frekvencijskom domenu. Dalje su prikazani rezultati za dve frekvencije ulaznog signala – 50 Hz i 870 Hz.

Na slici 7.3.2.1. su prikazane greške koje odgovaraju ovim signalima. Niža učestanost je parni umnožak učestanosti odabiranja signala, i na slici se vidi da očigledno postoji korelacija između ulaznog signala i greške kvantizacije. Ovakav rezultat je predviđen teorijom opisanom u poglavlju 3.6. Ovde se jasno vidi da se ovo dešava zato što se signal uzima uvek u istim tačkama, pa se greška kvantizacije periodično ponavlja. Sa druge strane, na slici možemo da vidimo da, kada je frekvencija sinuosoide 870 Hz, za koju signal odabiramo u drugim tačkama, greška nije korelisana sa signalom.



Slika 7.3.2.1. *Zavisnost greške kvantizacije od vremena (f= 50 Hz gore, f=870 Hz dole)*

Za date razlike je generisan i histogram na osnovu kog se može još bolje proceniti šta se dešava sa greškom kvantizacije. U oba slučaja greška kvantizacije ima raspodelu koja je približna uniformnoj centriranoj u nuli. U slučaju signala od 50 Hz ovo je neka vrsta srećne situacije – iako je greška bila korelisana sa signalom, greška ima oblik sinusoide čija je srednja vrednost nula. Prema tome, korelacija sa ulaznim signalom ne znači da greška ne može da ima nultu srednju vrednost.

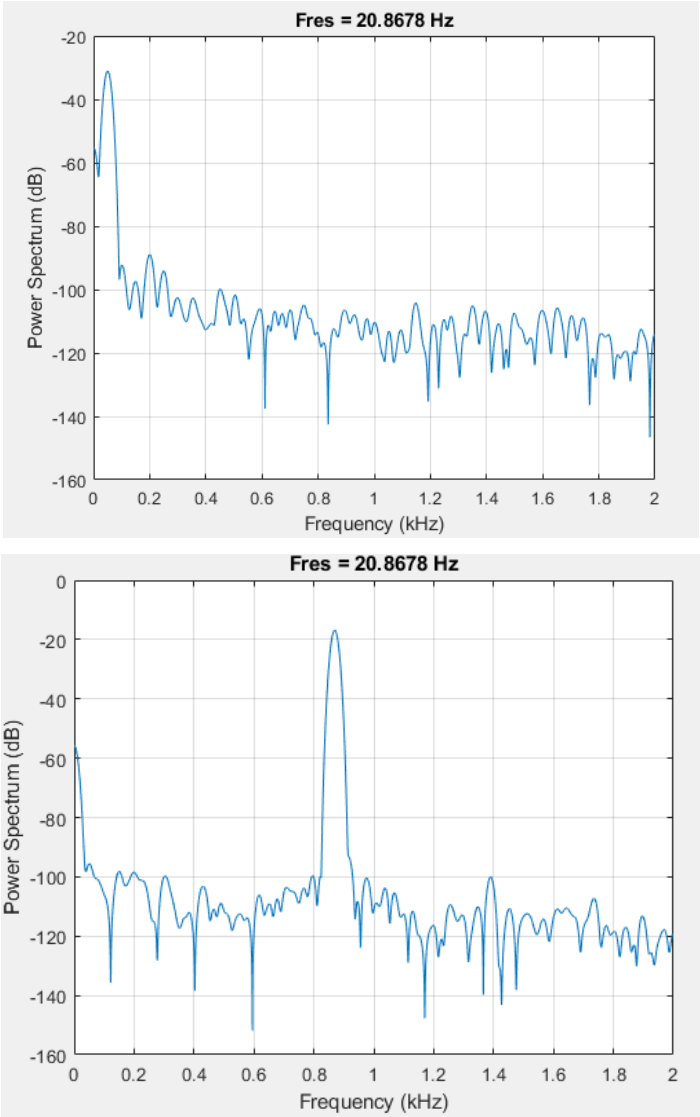


Slika 7.3.2.2. *Raspodela greške kvantizacije (f=50 Hz gore, f=870 Hz dole)*

Primetimo koliko je zapravo velika varijansa greške kvantizacije kada signal ima visoku učestanost. Ovo je posledica slabljenja signala na visokim učestanostima koje proizvodi sinc filtar koji se nalazi u sigma-

delta AD konvertoru. Primetimo još i da ova greška ne može da se detektuje posmatranjem samo parametra SINAD. Slabljenje signala na ovoj učestanosti iznosi 1.93 dB.

Konačno, na slici 7.3.2.3. možemo da vidimo snagu greške kvantizacije za oba ulazna signala. Sa slike se vidi da su komponente van glavne učestanosti koje predstavljaju šum ravnomerno raspoređene u spektru, i reda su veličine 100 dB. Iako ne možemo direktno da posmatramo početak efekata oblikovanja šuma - činjenica da komponente na višim učestanostima više slabe (zbog *sinc* filtra) kombinovano sa dobijenim spektrom govori o tome da je šum zaista bio jači na višim učestanostima.



Slika 7.3.2.3. *Snaga greške kvantizacije (f=50 Hz gore, f=870 Hz dole).*

8. Zaključak

U ovom radu je dat pregled sigma-delta AD konvertora u mikrokontrolerima serije MSP430. Testiran je sigma-delta AD konveror koji se nalazi u mikrokontroleru MSP430i2041. Dat je pregled fundamentalne teorije diskretizacije signala i objašnjeni su parametri koji opisuju AD konvertore. Sigma-delta AD konvertor je detaljno obrađen – na teorijskom nivou i njegova implementacija u pomenutom mikrokontroleru. Testiranje AD konvertora je bilo interesantno sa programerske tačke gledišta jer je zahtevalo programiranje na niskom i visokom nivou apstrakcije u programskim jezicima C, C++ i Matlab.

Rezultati testiranja kada je na ulazu bila sinusoida frekvencije 50 Hz su bili u skladu sa onim koji su dati u dokumentaciji proizvođača. Efektivan broj bitova AD konvertora je u ovom slučaju 13.57. Treba imati u vidu da je korišćen jako mali broj odbiraka, samo 512, za nalaženje svih rezultata. Ova informacija je značajna zbog toga što je poznato da metrike koje opisuju odnos signala i šuma kvadratno rastu sa brojem odbiraka. Nažalost, zbog toga što se morala koristiti serijska komunkacija i zbog ograničenosti memorije AD konvertora, nije bilo moguće koristiti veći broj odbiraka.

Moglo se videti da dinamičke karakteristike postaju lošije sa povećanjem učestanosti ulaznog signala (mada ne toliko značajno). Pokazano je i da ove metrike nisu sasvim dovoljne za objektivnu procenu efektivnog broja bitova kada signal ima visoku učestanost zbog slabljenja koje unosi *sinc3* filtar na višim učestanostima. Ovo ostavlja prostora da se ispita efektivan broj bitova u situacijama u kojima se koristi 16-bitni rezultat, ali je frekvencija odabiranja značajno veća – 32 kHz, pa samim tim i učestanost na kojoj *sinc* filtar počinje značajnije da degradira signal.

Pokazana je i zanimljiva osobina raspodele greške kvantizacije – njena korelacija sa ulaznim signalom ne mora nužno da znači da ona značajno odstupa od unifromne raspodele čija je srednja vrednost nula. Ovo je dodatni argument za opravdanost modelovanja greške kvantizacije belim šumom.

Iako je gornja granica učestanosti signala za koju AD konvertor radi u okviru specifikacija koje daje proizvođač dosta manja od polovine učestanosti odabiranja, ovo je moglo da se predvidi obzirom da je poznato kako radi *sinc* filtar. Ono što možda nije bilo odmah očigledno je koliko malu rezoluciju frekvencije možemo da postignemo. Ovo ograničenje proizilazi iz nemogućnosti da se sačuva više od 512 odbiraka. Poznato je da sa povećanjem broja odbiraka raste i rezolucija na kojoj se radi FFT. Sa korišćenim brojem odbiraka, ova rezolucija je oko 20 Hz. Sa ovako malom rezolucijom prirodno se pojavljuje i donja granica učestanosti pošto je tada za niske učestanosti realtivna greška između stvarne i procenjene frekvencije dosta velika. Ukoliko bi trebalo da se radi sa signalima nižih učestanosti, tada bi bilo bolje da se dati program modifikuje. Pošto je broj odbiraka ograničen, bolja rezolucija bi mogla da se postigne kada bi se u samom

mikrokontroleru radilo usrednjavanje signala i potom ta vrednost upisivala u bafer. Na ovaj način bi se smanjila učestanost odabiranja, a usrednjavanje signala bi funkcionisalo kao NF filtar koji bi sprečio preklapanje spektra.

9. Reference

Gisselquist Technology's ZipCPU . (n.d.). *Bit growth in FPGA arithmetic*. Retrieved from Gisselquist Technology's ZipCPU : https://zipcpu.com/dsp/2017/07/21/bit-growth.html

Analog Devices Inc., Engineeri. (2005). *Data Conversion Handbook.* Retrieved from https://www.analog.com/en/education/education-library/data-conversion-handbook.html

Analog Devices. (n.d.). *Sigma Delta ADCs and DACs.* Analog Devices. Retrieved from https://www.analog.com/media/en/technical-documentation/application-notes/29252429152571724505492368045817 1AN283.pdf

Asst. Prof. Dr. Rozita Teymourzadeh, C. (n.d.). *VLSI DESIGN OF ADVANCED DIGITAL FILTERS.* Retrieved from https://arxiv.org/ftp/arxiv/papers/1806/1806.03721.pdf

Bennett, W. R. (1948). Spectra of quantized signals. *The Bell System Technical Journal* .

D. Živković, M. P. (1997). *Impulsna i digitalna elektronika.* Beograd: Nauka.

Donadio, M. P. (2000). CIC Filter Introduction. Iowegian. Retrieved from http://home.mit.bme.hu/~kollar/papers/cic.pdf

Đurić, R. (n.d.). Materijali sa predmeta "Osnovi elektronike". Katedra za elektroniku Elektrotehničkog fakulteta Univerziteta u Beogradu.

Embedded. (2020). *DSP Tricks: Reducing A/D Converter Quantization Noise*. Retrieved from Embedded: https://www.embedded.com/dsp-tricks-reducing-a-d-converter-quantization-noise/

Embedded. (2020, February 17). *Understanding analog to digital converter specifications*. Retrieved from Embedded: https://www.embedded.com/understanding-analog-to-digital-converter-specifications/

Embedded Staff. (n.d.). *Understanding Analog to Digital Converter Specifications*. Retrieved from Embedded: https://www.embedded.com/understanding-analog-to-digital-converter-specifications/

Embedded Staff. (n.d.). *Understanding Cascaded Integrtor Comb Filters*. Retrieved from Embedded: https://www.embedded.com/understanding-cascaded-integrator-comb-filters/

Gisselquist Technology's ZipCPU. (2017). *Rounding Numbers without Adding a Bias*. Retrieved from Gisselquist Technology's ZipCPU: https://zipcpu.com/dsp/2017/07/22/rounding.html

Gruber, R. (n.d.). Efficient VHDL Implementation of Decimation Filters for a Next Generation Wireless Receiver. Institute for Electronics Graz University of Technology. Retrieved from https://diglib.tugraz.at/download.php?id=576a718eed9fc&location=browse

Harris, F. J. (2004). Cascade Integrator Comb Filters. In *Multirate Signal Processing for Communication Systems.* Pearson.

Hogenauer, E. (1981). An economical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics Speech and Signal Processing*.

Jovičić, N. (2020). Beleške sa predavanja iz predmeta "Namenski računarski sistemi". Katedra za elektroniku Elektrotehničkog fakulteta Univerziteta u Beogradu.

Kester, W. (2020). *Taking the Mystery out of the Infamous Formula SNR = 6.02N + 1.76dB and Why You Should Care* . Retrieved from Analog Devices: https://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf

Li, B. (2003). Design of Multi-bit Sigma-Delta Modulators for Digital Wireless Communications. Stockholm. Retrieved from https://www.diva-portal.org/smash/get/diva2:9496/FULLTEXT01.pdf

Manuel Jiménez, R. P. (2014). *Introduction to Embedded Systems Using Microcontrollers and the MSP430.* Springer.

Measurement Computing . (2004). *Data Acquisition Handbook.* Measurement Computing Corporation.

Microchip. (2020, 3 31). *ADC Differential Nonlinearity*. Retrieved from Microchip Developer Help: https://microchipdeveloper.com/adc:adc-differential-nonlinearity

Microchip. (2020, February 17). *ADC Gain Error*. Retrieved from Microchip Developer: https://microchipdeveloper.com/adc:adc-gain-error

Microchip. (2020, 3 31). *ADC Integral Non-Linearity*. Retrieved from Microchip Developer Help: https://microchipdeveloper.com/adc:adc-inl

Microchip. (2020, February 17). *ADC Offset Error*. Retrieved from Microchip Developer : https://microchipdeveloper.com/adc:adc-offset-error

Microchip Developer. (2020). *ADC Quantization Error*. Retrieved from Microchip Developer Help : https://microchipdeveloper.com/adc:adc-quantization-error

Popović, M. (2003). *Digitalna obrada signala.* Beograd: Akademska misao.

Popović, M. (2006). *Signali i sistemi.* Beograd: Akademska misao.

Sangil Park, P. D. (2008). *Principles d Sigma-Delta Modulation for Analog-to-Digital Converters.* Motorola. Retrieved from https://web.archive.org/web/20060621221221/http://digitalsignallabs.com/SigmaDelta.pdf

Sanjay Pithadia, S. L. (2009). *LDO PSRR Measurement Simplified.* Texas Instruments. Retrieved from https://www.ti.com/lit/an/slaa414a/slaa414a.pdf?ts=1593719793111&ref_url=https%253A%252F%252Fwww.google.com%252F#:~:text=Power%20Supply%20Rejection%20Ratio%20or,frequencies%2C%20injected%20at%20its%20input.

Saranovac, L. (2020). Beleške sa predavanja iz predmeta "Digitalna Elektronika". Katedra za elektroniku Elektrotehničkog fakulteta Univerziteta u Beogradu.

Texas Instruments. (1995). *Understanding Data Converters.* Retrieved from http://www.ti.com/lit/an/slaa013/slaa013.pdf

Texas Instruments. (2003). *Combining the ADS1202 with an FPGA Digital Filter for.* Retrieved from http://www.ti.com/lit/an/sbaa094/sbaa094.pdf

Texas Instruments. (2014). *MSP430i2xx Family - Users Guide.*

Texas Instruments. (2020). *MSP430i204x, MSP430i203x, MSP430i202x Mixed-Signal Microcontrollers Datasheet.*

Texas Instruments. (n.d.). *Digital Filter Types in Delta-Sigma ADC.* Texas Instruments. Retrieved from http://www.ti.com/lit/an/sbaa230/sbaa230.pdf

U.Meyer-Baese. (n.d.). *Digital Signal Processing with Field Programmable Gate Arrazes.* Springer.

Željko Đurović, B. D. (2006). *Sistemi automatskog upravljanja.* Beograd: Akademska misao.

Prilog

low_level_init.c

```
/* --COPYRIGHT--,BSD_EX
 * Copyright (c) 2013, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ****************************************************************************
 *
 *                       MSP430 CODE EXAMPLE DISCLAIMER
 *
 * MSP430 code examples are self-contained low-level programs that typically
 * demonstrate a single peripheral function or device feature in a highly
 * concise manner. For this the code may rely on the device's power-on default
 * register values and settings such as the clock configuration and care must
 * be taken when combining code from several examples to avoid potential side
 * effects. Also see www.ti.com/grace for a GUI- and www.ti.com/msp430ware
 * for an API functional library-approach to peripheral configuration.
 *
 * --/COPYRIGHT--*/
//****************************************************************************
// MSP430i20xx Initialization Routines - low_level_init.c
//
// This function is called by the start-up code before "main" is called, and
// before data segment initialization is performed. The function affects the
// following modules:
//  - JTAG - JTAG is disabled
//  - TLV - A TLV checksum is performed
//  - PMM - The shared reference is calibrated to 1.16V
//  - Clock System - The DCO is calibrated to 16.384MHz
//  - SD24 - The SD24 reference voltage trimming is calibrated
//
```

```
// The return value of this function controls if data segment initialization
// should take place. If 0 is returned, it is bypassed.
//
// Any code example or application written for the MSP430i20xx which takes
// advantage of any of the affected modules is suggested to include this
// function to ensure full calibration of the application.
//
// Additionally, this initialization routine is suggested for small, short
// code examples to ensure the JTAG will be unlocked. The i20xx series of
// devices requires execution of 64 MCLK cycles before the JTAG can be unlocked.
// If a device does not execute 64 MCLK cycles, it cannot be accessed via JTAG.
//*****************************************************************************
#include "msp430.h"

#ifdef __TI_COMPILER_VERSION__
int _system_pre_init(void)
#elif __IAR_SYSTEMS_ICC__
int __low_level_init(void)
#elif __GNUC__
extern int system_pre_init(void) __attribute__((constructor));
int system_pre_init(void)
#else
#error Compiler not supported!
#endif
{
    unsigned long *jtagPwd = (unsigned long *)JTAG_DIS_PWD1;

    /* Feed the watchdog timer */
    WDTCTL = WDTPW | WDTCNTCL;

    /* Check JTAG password locations and disable JTAG if passwords don't match.
     * Else the JTAG will be enabled in the 64th cycle after reset.
     */
    if ((*jtagPwd != 0x00000000) && (*jtagPwd != 0xFFFFFFFF))
    {
        /* Disable JTAG */
        SYSJTAGDIS = JTAGDISKEY;
    }

    /* Calibration section
     * Check for the BORIFG flag in IFG1. Execute calibration if this was a BORIFG.
     * Else skip calibration
     */
    if (IFG1 & BORIFG)
    {
        /* Perform 2's complement checksum on 62 bytes of TLV data */
        unsigned int checksum = 0;
        unsigned char *TLV_address_for_parse = ((unsigned char *)TLV_START);
        unsigned int *TLV_address_for_checksum = ((unsigned int *)TLV_START + 1);

        do
        {
            checksum ^= *TLV_address_for_checksum++;
        } while (TLV_address_for_checksum <= (unsigned int *)TLV_END);

        checksum ^= 0xFFFF;
        checksum++;

        /* If check sum is not correct go to LPM4 */
        if (*((unsigned int *)TLV_START) != checksum)
        {
            /* Enter LPM4 if checksum failed */
            __bis_SR_register(LPM4_bits);
```

```
        }

        /* Check sum matched, now set calibration values */

        /* Calibrate REF */
        REFCAL1 = *(TLV_address_for_parse + TLV_CAL_REFCAL1);
        REFCAL0 = *(TLV_address_for_parse + TLV_CAL_REFCAL0);

        /* Calibrate DCO */
        CSIRFCAL = *(TLV_address_for_parse + TLV_CAL_CSIRFCAL);
        CSIRTCAL = *(TLV_address_for_parse + TLV_CAL_CSIRTCAL);
        CSERFCAL = *(TLV_address_for_parse + TLV_CAL_CSERFCAL);
        CSERTCAL = *(TLV_address_for_parse + TLV_CAL_CSERTCAL);

        /* Calibrate SD24 */
        SD24TRIM = *(TLV_address_for_parse + TLV_CAL_SD24TRIM);

        /* Clear BORIFG */
        IFG1 &= ~(BORIFG);
    }

    /* Feed the watchdog timer */
    WDTCTL = WDTPW | WDTCNTCL;

    /* Return value:
     *  1 - Perform data segment initialization.
     *  0 - Skip data segment initialization.
     */
    return 1;
}
```

main.c

```
/*
 * @file main.c
 * @brief Program that continually reads 512 samples of 24-bit AD conversion results
 * from channel 0 and sends them via UART.
 *
 * @detail ADC converts signal and writes the result until the buffer is full. When
 * there's no more space left, UART begins sending all collected samples, each one
 * as three 8-bit messages. When all messages have been sent, UART is stopped and ADC
 * takes over again. Micro-controller enters low power after initializing everything
 * so all actions happen inside interrupts.
 *
 */
#include "msp430.h"
#include <stdint.h>

#define bufferSize 512

// Array that stores conversion result. buffer[0][i] contains LSB for i-th
// conversion
static uint8_t buffer[3][bufferSize];

static uint16_t producerCounter = 0;
static uint16_t consumerCounter = 0;

static uint8_t producerCurrBuff = 0;
static uint8_t consumerCurrBuff = 0;

static uint8_t PCReady = 0;
static uint8_t ADCReady = 0;
```

```
inline void initADC(){

    SD24CTL = SD24REFS;                         // Select internal reference voltage.
    SD24CCTL0  |= SD24DF | SD24IE | SD24LSBTOG; // Enable interrupt, use two's
complement, 256 OSR and enable LSB toggle.

    __delay_cycles(3200);                       // Delay ~200us for 1.2V ref to settle

}

inline void startADC(){
    SD24CCTL0 |= SD24SC;                        // Set bit to start conversion
}

inline void initUART(){

    P1SEL0 |=   BIT2 | BIT3;                    // P1.2/3 eUSCI_A Function
    P1SEL1 &= ~(BIT2 | BIT3);

    UCA0CTL1 |= UCSWRST;                        // Hold eUSCI in reset
    UCA0CTL1 |= UCSSEL_2;                       // SMCLK
    UCA0BR0   = 0xAA;                           // 9600 baud
    UCA0BR1   = 0x06;
    UCA0MCTLW = 0xD600;                         // 16.384MHz/9600 = 1706.6667 (See UG)
    UCA0CTL1 &= ~UCSWRST;                       // Release from reset
    UCA0IE |= UCRXIE;                           // Enable RX interrupt.
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;                   // Stop WDT

    initUART();
    initADC();
    startADC();

    __bis_SR_register(LPM0_bits | GIE);         // Enter LPM0 w/ interrupts
    __no_operation();                           // For debugger
}

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=SD24_VECTOR
__interrupt void SD24_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(SD24_VECTOR))) SD24_ISR (void)
#else
#error Compiler not supported!
#endif
{
    static int16_t producerRes0;                // Lower 16 bits.
    static int16_t producerRes1;                // Higher 16 bits.

    switch (__even_in_range(SD24IV,SD24IV_SD24MEM3)) {
        case SD24IV_NONE: break;
        case SD24IV_SD24OVIFG: break;
        case SD24IV_SD24MEM0:

            producerRes1 = SD24MEM0;            // Read upper 16 bits.
            producerRes0 = SD24MEM0;            // Read lower 16 bits.

            buffer[producerCurrBuff++][producerCounter] = producerRes0;           //
Byte 0.
            buffer[producerCurrBuff++][producerCounter] = producerRes0 >> 8;      //
Byte 1.
```

```
            buffer[producerCurrBuff++][producerCounter++] = producerRes1 >> 8;    //
Byte 2.

            producerCurrBuff = 0;
            if (producerCounter >= bufferSize) { // Buffer full. Stop writing to
buffer and start UART transmission.
                producerCounter = 0;
                SD24CCTL0  &= ~SD24IE;
                if (PCReady != 0){
                    PCReady = 0;
                    ADCReady = 0;
                    UCA0IE |= UCTXIE;
                }
                else
                    ADCReady = 1;
            }

            break;

        case SD24IV_SD24MEM1: break;
        case SD24IV_SD24MEM2: break;
        case SD24IV_SD24MEM3: break;
        default: break;
    }
}


#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
    switch(__even_in_range(UCA0IV,USCI_UART_UCTXCPTIFG)) {
        case USCI_NONE: break;
        case USCI_UART_UCRXIFG:
            PCReady = UCA0RXBUF;
            if (ADCReady != 0){
                ADCReady = 0;
                PCReady = 0;
                UCA0IE |= UCTXIE;
            }
        case USCI_UART_UCTXIFG:

            UCA0TXBUF = buffer[consumerCurrBuff++][consumerCounter];    // Send LSB of
a sample first.

            if (consumerCurrBuff >= 3) {                                // Move to
next sample if all bytes have been sent.

                consumerCurrBuff = 0;
                consumerCounter++;

                if (consumerCounter >= bufferSize){                     // Everything
has been sent. Stop UART and start ADC again.
                    consumerCounter = 0;
                    UCA0IE &= ~UCTXIE;
                    SD24CCTL0  |= SD24IE;
                }
            }
```

```
            break;
        case USCI_UART_UCSTTIFG: break;
        case USCI_UART_UCTXCPTIFG: break;
        default: break;
    }
}
```

plot_real_time.m

```
% July, 2020
% plot_real_time.m
%
% This script is used to plot incoming stream of data read from serial
% port. It is intended to be used with MSP430i2041 which reads 24-bit values
% from ADC; sends them via UART to PC at 9600 baud rate. This script
% plots data in real time. UART is sending 8 bit data in order from LSB to
% MSB. In order to convert received bytes into int32 a C++ function
% pack_data is used. UART sends 512 consequtive samples. These are
% displayed along with SINAD parameter.
%
% Clicking space will save current 512 samples.
%
% Start this script before starting MSP to make sure things are
% synchronised.
%
% Jovana Savic, jovana.savic9494@gmail.com
%--------------------------------------------------------------------------%
function R = plot_real_time()
%--------------------------------------------------------------------------%
% Parameters used.
%--------------------------------------------------------------------------%
filename = "test_batch_1.txt";
baudRate = 9600;
max24bit = 8388608; % Maximum absolute value that can be displayed with signed 24
bits.
refVoltage = 1.2;
GAIN = 1;
numOfSamples = 512; % Ideally power of two for fft.
yLimits = [-refVoltage/GAIN * 1.2, refVoltage/GAIN * 1.2];
xLimits = [1, numOfSamples];
scaleFactor = 1 / max24bit * refVoltage / GAIN;
Fs = 4000; % Sampling rate is 4000 Hz.
f = Fs*(0:(numOfSamples/2))/numOfSamples;

savedData = zeros(numOfSamples*10, 1);
savedDataCount = 0;
saveData = 0;
sync = 1;

%--------------------------------------------------------------------------%
% Prepare for plot.
%--------------------------------------------------------------------------%

% Open a serial port
s = serial('COM4');
set(s,'BaudRate',baudRate);

try
    fopen(s);
catch
```

```
    fclose(instrfind); % Close all ports in case someone (me) didn't close this one.
    fopen(s);
end

%--------------------------------------------------------------------------%
% Prepare initial plot.
% Loop until someone closes the figure. After 512 new values,
% refresh the plot.
%--------------------------------------------------------------------------%
data = zeros(numOfSamples, 1);

% Create figure.
fig = figure(1);
set(fig,'WindowKeyPressFcn',@keyPressCallback);

while (ishandle(fig))
    write(s, 1, 'uint8');
    for i=1:numOfSamples
        tmp_data = uint8(fread(s, 3, 'uint8')); % From LSB (1) to MSB (3)
        while (isempty(tmp_data))
            tmp_data = uint8(fread(s, 3, 'uint8'));
        end
        data(i) =  double((pack_data(tmp_data(3), tmp_data(2),
tmp_data(1)))).*scaleFactor;
    end

    Y = fft(data);
    P2 = abs(Y/numOfSamples);
    P1 = P2(1:numOfSamples/2+1);
    P1(2:end-1) = 2*P1(2:end-1);

    [~,loc] = max(P1);
    FREQ_ESTIMATE = f(loc);

    subplot(2, 1, 1);
    plot(data), ylim(yLimits), xim(xLimits);
    title ( {['Frequency estimate = ', num2str(FREQ_ESTIMATE),' Hz']});
    subplot(2,1,2)
    sinad(data, Fs);

    drawnow

    if (saveData == 1)
        save_data();
    end
end

%--------------------------------------------------------------------------%
% Clean up serial port
%--------------------------------------------------------------------------%
fclose(s);
delete(s);
clear s;

fileID = fopen(filename,'w');
fprintf(fileID,'%d\n',savedData);
fclose(fileID);

%--------------------------------------------------------------------------%
% Key press callback function.
%--------------------------------------------------------------------------%
 function keyPressCallback(source,eventdata)
      % determine the key that was pressed
```

```
      keyPressed = eventdata.Key;
      if strcmpi(keyPressed,'space') % Save current data
          saveData = 1;
      end
 end

function save_data()
    savedData(savedDataCount*numOfSamples+1:(savedDataCount + 1)*numOfSamples) =
data(:, 1);
    savedDataCount = savedDataCount + 1;
    savedDataCount = mod(savedDataCount,10);
    saveData = 0;
end

end
```

pack_data.cpp

```
/*=========================================================

 To build in Matlab: mex -R2018a pack_data.cpp
 *=========================================================*/

#include <iostream>
#include <cstdint>
#include "mex.h"



/*********************************/

int32_t mexcpp(uint8_t byte2, uint8_t byte1, uint8_t byte0) {
    int32_t result = 0;
    if (byte2 & 0x80){
      // This is a negative number.
      result |= 0xFF000000; // add leading zeros
    }
    result |= byte0 | (byte1 << 8) | (byte2 << 16);
    return result;
}

/* The gateway function. */
void mexFunction(int nlhs, mxArray* plhs[],
                 int nrhs, const mxArray* prhs[]) {

    /* Check for proper number of arguments */
    if(nrhs != 3) {
        mexErrMsgIdAndTxt("MATLAB:mexcpp:nargin",
                          "MEXCPP requires three input arguments.");
    }
    if(nlhs != 1) {
        mexErrMsgIdAndTxt("MATLAB:mexcpp:nargout",
                          "MEXCPP requires one output argument.");
    }

    /* Check if the input is of proper type */
    if(!mxIsUint8(prhs[0]) ||                                    // not double
       mxIsComplex(prhs[0]) ||                                   // or complex
       !mxIsScalar(prhs[0])) {                                  // or not scalar
        mexErrMsgIdAndTxt("MATLAB:mexcpp:typeargin",
                          "First argument has to be uint8.");
```

```
  }
  if(!mxIsUint8(prhs[1]) ||                                      // not double
     mxIsComplex(prhs[1]) ||                                     // or complex
     !mxIsScalar(prhs[1])) {                                    // or not scalar
      mexErrMsgIdAndTxt("MATLAB:mexcpp:typeargin",
                        "Second argument has to be uint8.");
  }
  if(!mxIsUint8(prhs[2]) ||                                      // not double
     mxIsComplex(prhs[2]) ||                                     // or complex
     !mxIsScalar(prhs[2])) {                                    // or not scalar
      mexErrMsgIdAndTxt("MATLAB:mexcpp:typeargin",
                        "Third argument has to be uint8.");
  }


  /* Acquire pointers to the input data */
  uint8_t* vin1 = mxGetUint8s(prhs[0]);
  uint8_t* vin2 = mxGetUint8s(prhs[1]);
  uint8_t* vin3 = mxGetUint8s(prhs[2]);


  plhs[0] = mxCreateNumericMatrix(1, 1, mxINT32_CLASS, mxREAL);
  int32_t* data = (int32_t*) mxGetData(plhs[0]);
  *data = mexcpp(*vin1, *vin2, *vin3);
  return;

}
```