Univerzitet u Beogradu
Elektrotehnički fakultet
Odsek za elektroniku

Predmet: 32bitni mikrokontroleri

Projekat: Implementacija FFT algoritma za obradu radarskog signala na ARM Cortex M4 procesoru

Autor: Predrag Bratić, 3007/16
Mentor: doc. dr Nenad Jovičić

Sadržaj

1 Uvod

Zadatak odabranog projekta u sklopu predmeta 32bitni mikrokontroleri je implementacija FFT (*Fast Fourier Transform)* algoritma nad radarskim signalom. Projekat se sastoji od sledećih dijelova:

- Akvizicije, odnosno analogno-digitalne konverzije primljenog signala sa radara
- Obrade primljenih podataka (FFT algoritam)
- Slanja obrađenih podataka na PC putem serijskog porta i prikaz dobijenog spektra u MATLAB programskom paketu

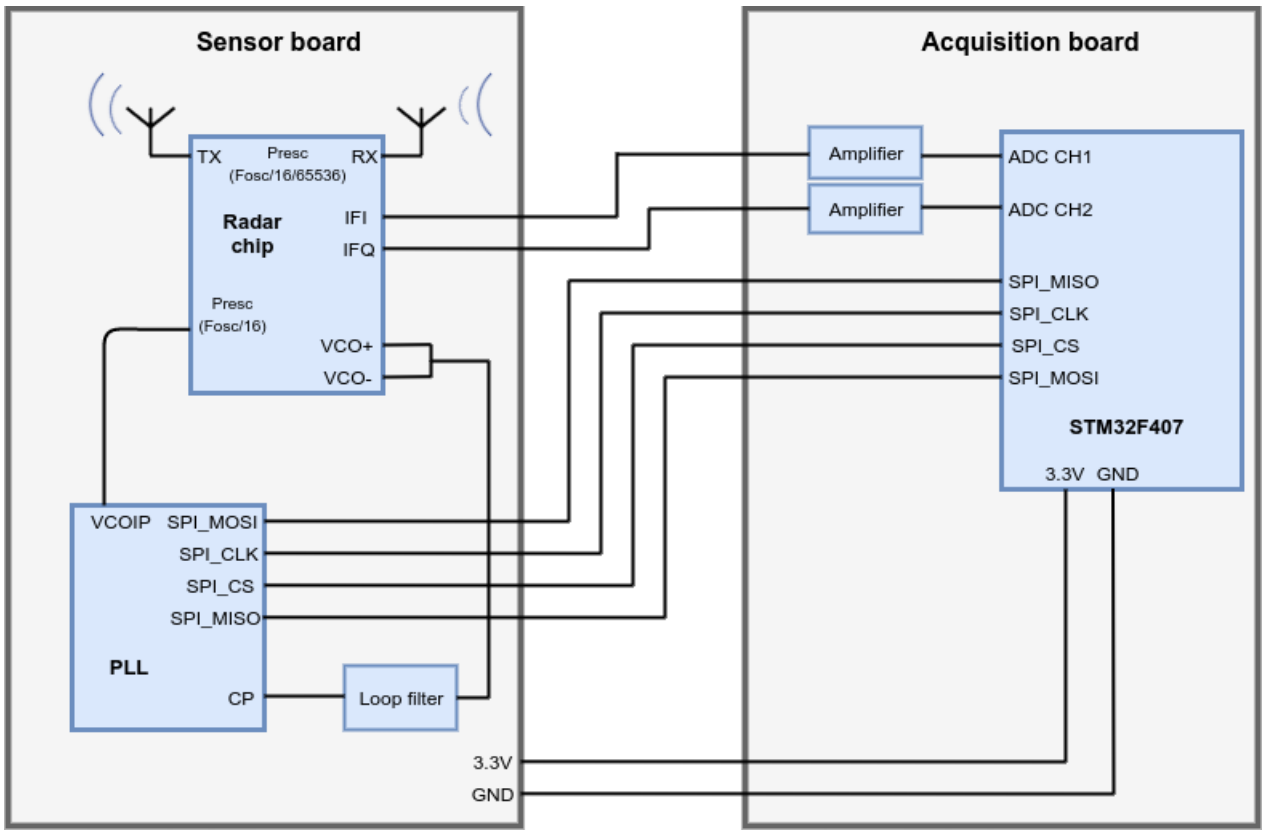
Pored A/D konvertora, od ostalih periferija korišćena su dva tajmera, SPI za konfiguraciju parametara PLL-a na senzor ploči, kao i UART za slanje rezultata na PC računar. Projekat je rađen u softverskom paketu *CooCox*, a na ploči za akviziciju je *ARM Cortex-M4* procesor, široko korišćen za zahtjevnije računske operacije vezane za digitalnu obradu signala.

2 Hardver sistema

Radarski sistem sa kojim je rađena akvizicija radi u FMCW (*Frequency Modulated Continuous Wave*) modu u kom konstantno na svom izlazu generiše frekvencijski modulisan signal. Sistem se sastoji od dvije ploče:

- Senzorske (RF) ploče
- STM32F407 ploče za akviziciju sa *ARM Cortex-M4* procesorom

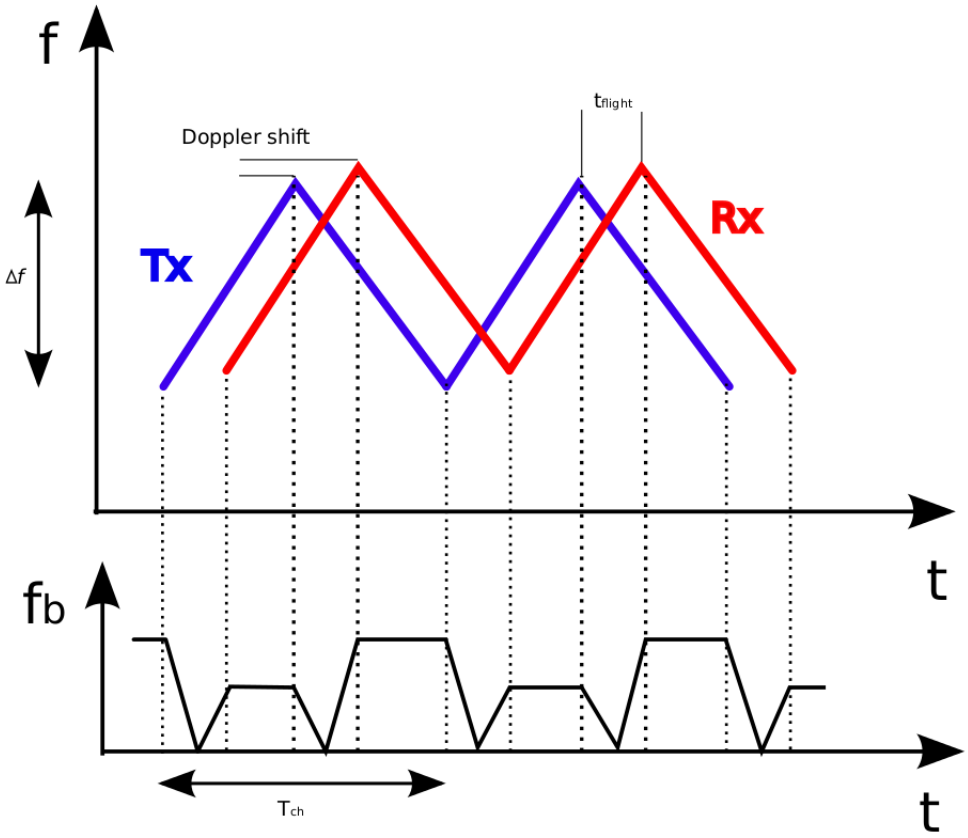
Glavni dijelovi senzorske ploče su radarski cip i PLL (*Phase Locked Loop*). Zadatak PLL-a je da generiše kontrolni napon za naponski kontrolisani oscilator (VCO - *Voltage Controlled Oscilator*) koji generiše na svom izlazu frekvencijski modulisani signal. Na slici ispod je data blok šema sistema. PLL konfiguracija se obavlja preko SPI interfejsa. Konfiguracija podrazumjeva način na koji će se obavljati frekvencijska modulacija izlaznog signala. U narednom poglavlju o principu rada FMCW radara je detaljnije objašnjen rad i cilj sistema.



Slika 1: Blok šema sistema

3 Princip rada FMCW radara

Na svom izlazu, sistem predaje frekvencijski modulisan signal pojačan dodatno pomoću pojačavača snage. Predajni signal se odbija od mete i primljeni signal predstavlja zakašnjen predajni signal. Miksovanjem ova dva signala se dobija informacija o *beat* frekvenciji odnosno frekvenciji na osnovu koje je moguće odrediti rastojanje mete od radara. Frekvencijska obrada rađena na kontroleru upravo iz primljenog signala obrađenog u analognom domenu, izvlači informaciju o ovoj frekvenciji. Na slici ispod dat je princip rada FMCW radara.



Slika 2: Princip rada FMCW radara

Na slici iznad su označeni određeni parametri. Δf predstavlja opseg frekvencija predajnog signala dok T_{ch} predstavlja period ponavljanja ovog opsega frekvencija. Ovaj prikaz frekvencije u odnosu na vrijeme predstavlja *chirp* signal koji je u ovom slučaju trougaonog oblika, a može imati i druge primjenljive oblike u zavisnosti od aplikacije. Na slici je prikazan generalniji slučaj kada se meta prema kojoj se šalje signal kreće pa se dobijaju dvije *beat* frekvencije iz kojih se može izvući informacija o udaljenosti i brzini kretanja mete. Ipak, vidi se da primljeni *chirp* signal nije pomjeren po visini, da bi te *beat* frekvencije bile iste. T_{flight} je vrijeme za koje se signal pošalje, a zatim primi od mete. Kako se elektromagnetni talasi kreću približno brzinom svjetlosti to je ovo vrijeme izuzetno kratko i iznosi: $T_{flight} = 2 \cdot R/c$ što za rastojanje od 15 m iznosi 100 ns.

Jendostavnom matematikom sa slike iznad se može izvesti zavisnost udaljenosti mete od *beat* frekvencije (sličnost trouglova):

$$R = \frac{c \cdot T_{ch} \cdot f_b}{4 \cdot \Delta f} \tag{1}$$

Krajnji rezultat ovog projekta je frekvencijski spektar primljenog signala, obrađenog u analognom domenu. Kako se frekvencijskom analizom tog signala dobija *beat* frekvencija, na osnovu ostalih poznatih parametara se moze izračunati rastojanje mete od radara po formuli iznad. Rezolucija daljine mete isključivo zavisi od propusnog opsega frekvencija Δf:

$$\Delta R = \frac{c}{2 \cdot \Delta f} \tag{2}$$

4 Softver sistema

Softver za sistem je pisan u *CooCox IDE* sa GCC kompajlerom. Programiranje i debagovanje je omogućeno preko ST-LINK v2 programatora. Za uspješno povezivanje sa PC računarom preko USB-a, odgovarajući ST-LINK v2 drajveri moraju biti instalirani. Softver se oslanja na standardnu STM32F4 CMSIS platformu i STM32F4xx_StdPeriph_Driver biblioteke iz STM32F4-Discovery_FW_V1.1.0 paketa. Link paketa je sledeći: `https://github.com/rowol/stm32_discovery_arm_gcc/tree/master/STM32F4-Discovery_FW_V1.1.0`. Za potrebe implementacije FFT algoritma, koristi se CMSIS DSP biblioteka.

Komunikacija između mikrokontrolera i PC računara je zamišljena protokolom komandi koje se šalju preko UART-a. Naime, u fajlovima *Commands.h* i *Commands.c* nalazi se spisak mogućih komandi za slanje od PC-a do mikrokontrolera. Te komande uključuju, reset komandu, setovanje frekvencije odabiranja, pokretanje FFT obrade, setovanje parametara PLL-a, itd. Komande su organizovane u jednostavnom formatu, tako da je dodavanje nove komande jednostavan proces. Naime, svaka komanda se šalje karakter po karakter preko serijskog porta, a programski kod na mikrokontroleru učitava karaktere u niz sve dok ne dobije znak za kraj komande. Nakon, uspješno očitanog znaka za kraj komande (CR karakter), provjerava se u tabeli komandi, po nazivu, da li je poslati string ustvari neka od već definisanih komandi. Ako jeste prelazi se na izvršavanje komande. Ako neke komande zahtjevaju dodatni argument uz njih, potrebno je poslati u formatu: naziv komande, razmak, argument.

Glavna funkcija je definisana u fajlu *main.c*. U ovom fajlu se vrši FFT obrada i slanje rezultujućih podataka na seriski port sto je detaljnije objašnjeno u predstojećem tekstu.

U fajlovima *my_functions.h* i *my_functions.c* su definisane dodatne funkcionalnosti, kao što su: inicijalizacija SPI interfejsa za komunikaciju sa PLL-om, kopiranje niza bajtova potrebnih za obradu, generisanje prozorske funkcije itd.

U fajlovima *ADC.h* i *ADC.c* su date definicije vezane za analogno-digitalnu konverziju. Tu su funkcije za inicijalizaciju A/D konvertora, startovanje i stopiranje konverzije itd.

Konfiguracija serijskog prenosa je odrađena u fajlovima *UARTconf.h* i *UARTconf.c*. U fajlovima *Timer_set.h* i *Timer_set.c* su inicijalizovani potrebni tajmeri.

Sekvenca komandi koju je potrebno poslati za uspješno startovanje FFT obrade je sledeća:

- **rst** - komanda za reset sistema
- **dofft** - komanda za setovanje flega za FFT obradu primljenog signala
- **fmcw** - komanda za konfiguraciju parametara PLL-a (konfiguracija *chirp* signala)
- **fs 80000** - komanda za setovanje frekvencije odabiranja i početak A/D konverzije (podrazumjevano podešavanje frekvencije odabiranja je 80 kHz, ali postoje i druge mogućnosti)

4.1 Procesiranje primljenog signala

Poslednja gore navedena komanda, čiji je format **fs arg**, gdje je **arg** željena frekvencija odabiranja, pokreće A/D konverziju. A/D konverzija se pokreće u prekidnoj rutini jednog od korišćenih tajmera (TIM4) koja se izvršava na svakih **1/arg** sekundi. Zbog toga po jednom *chirp* signalu imamo:

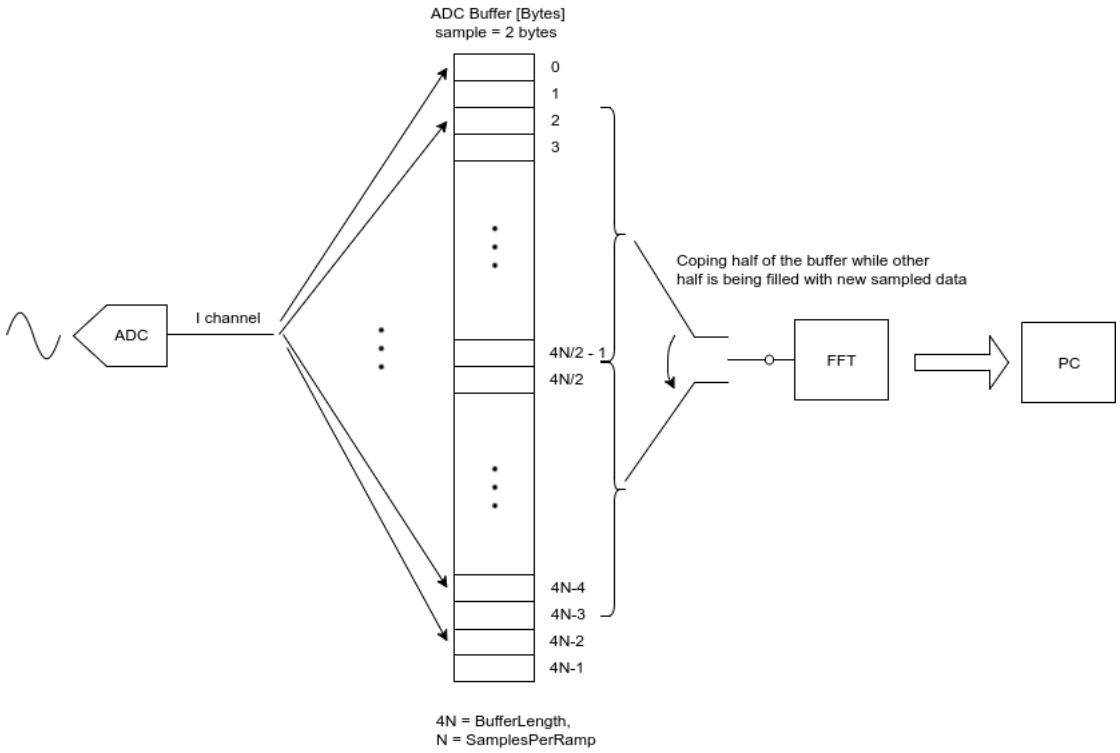
$$N = T_{ch} \cdot F_s \tag{3}$$

odbiraka. 80 kHz je podrazumjevana frekvencija odabiranja, međutim mogu da se koriste i druge podržane frekvencije date u tabeli ispod.

Frekvencija odabiranja
1 kHz
20 kHz
25 kHz
40 kHz
80 kHz
160 kHz

Table 1: Podržane frekvencije odabiranja

Nakon sto je jedan odbirak konvertovan u DR (*Data Register*) registar ADC-a, DMA kontroler ga prenosi odmah u mjesto predviđeno u memoriji. Mjesto u memoriji gdje se smještaju odbirci je ADC bafer i veličine je $4 \cdot N$. Razlog za ovu veličinu bafera je taj da su odbirci signala dvobajtni (iako je korisno samo 12 bita zbog rezolucije ADC-a). Ako imamo N odbiraka po jednom *chirp* signalu, onda imamo $2 \cdot N$ bajtova, a to znači da polovinu ADC bafera napune podaci koji potiču od jednog *chirp* signala. Nakon punjenja polovine bafera, setuje se fleg koji daje znak da se ovi podaci mogu iščitati i koristiti za obradu. Ovi podaci se kopiraju u novi niz u kojem se nakupljaju podaci iz ADC bafera da bi se na većem segmentu vršila FFT obrada. Za vrijeme dok se ovi podaci kopiraju, DMA kontroler podacima iz DR registra ADC-a puni drugi dio bafera. Tako se izbjegava prebrisavanje neiščitanih podataka iz ADC bafera. Ovaj postupak je prikazan na slici ispod:



Slika 3: Princip rada akvizicije i obrade signala

Nakon uspješnog nakupljanja dovoljnog broja podataka, vrši se obrada. Funkcije koje se pozivaju za FFT obradu su sledeće:

- ***void arm_mean_f32(float32_t * pSrc,***
  uint32_t blockSize,
  float32_t * pResult) - funkcija koja traži srednju vrijednost vektora realnih brojeva; traži se srednja vrijednost da se oduzme DC frekvencija

- ***void arm_offset_f32(float32_t * pSrc,***
  float32_t offset,
  float32_t * pDst,
  uint32_t blockSize) - funkcija kojom se od ulaznog signala oduzima njegova srednja vrijednost (DC vrijednost)

- ***void arm_rfft_fast_f32(arm_rfft_fast_instance_f32 * S,***
  float32_t * p,
  float32_t * pOut,
  uint8_t ifftFlag) - funkcija koja vrši FFT obradu, vraća realan spektar

- ***void arm_abs_f32(float32_t * pSrc,***
  float32_t * pDst,
  uint32_t blockSize) - funkcija koja izračunava amplitudsku karakteristiku frekvencijskog spektra

Nakon obrade spektar se šalje preko UART-a na PC računar brzinom od 921600 b/s.

Kada je potrebno da se tačno zna od koje tacke *chirp* signala kreću podaci, neophodno je izvršiti sinhronizaciju starta početka *chirp* signala i starta A/D konverzije. Za to je zadužen

tajmer TIM2, koji radi u *capture* modu i mjeri preskaliranu izlaznu frekvenciju VCO-a. Naime, kako je na startu *chirp* signala minimalna frekvencija, u ovoj prekidnoj rutini se svaki put kada se izvrši, zapamti trenutna frekvencija pa se poredi sa prethodnom i tim algoritmom se odredi minimum. Kada je minimum određen, setuje se fleg za početak konverzije tj. tajmer TIM4 zadužen za A/D konverziju može da izvrši rutinu do kraja. Idući start konverzije ne kreće dok se opet ne setuje navedeni fleg.

5 Zaključak

Na osnovu dobijenih rezultata i prikazivanja na PC računaru, vidi se da je *beat* frekvencija vidljiva za veća rastojanja, dok je sam spektar dosta zašumljen, te da je potrebno odraditi predprocesiranja na vremenskom signalu prije FFT obrade za čistiji spektar. Na osnovu *beat* frekvencije može da se odredi rastojanje mete što ovom sistemu daje veliku primjenu u realnim problemima. Naime, uzevši u obzir i Doplerov pomjeraj, moze da se odredi i brzina kretanja mete. Idući korak u algoritmu bilo bi predprocesiranje primljenog signala, zatim FFT obrada, a nakon toga implementacija algoritma za detekciju pikova u spektru, odakle bi se informacija o rastojanju mete mogla dobiti direktno na kontroleru. Ipak, za složeniju obradu na kontroleru, ograničavajući faktor postaje i RAM memorija sistema.

Literatura

[1] *Predavanja iz predmeta 32bitni mikrokontroleri*

[2] *STM32F4xx Reference Manual:* http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf