VJEŽBA 4: VREMENSKI BROJAČI. D/A PRETVORNIK.

I. Cilj vježbe

Upoznati se s dostupnim vremenskim brojačima (engl. *timer*) na mikroupravljaču STM32F407VG. Podešavanje i upotreba vremenskih brojača opće namjene. Generiranje pulsno širinsko moduliranog signala pomoću vremenskog brojača. Upotreba vremenskog brojača i digitalno-analognog (D/A) pretvornika u svrhu generiranja zvuka.

II. Opis vježbe

Na laboratorijskim vježbama koristi se STM32F4 Discovery razvojna ploča kao hardverska komponenta dok se Atollic TrueStudio koristi kao pripadno integrirano razvojno okruženje. U vježbi se demonstrira podešavanje i upotreba vremenskih brojača mikroupravljača STM32F407VG: generiranje vremenske baze i generiranje pulsno-širinsko moduliranog (engl. *Pulse Width Modulation* - PWM) signala. Praktična upotreba PWM signala pokazuje se na primjeru upravljanja svjetlinom ugrađenih LED dok se generiranje vremenske baze u vježbi koristi za generiranje zvučnog signala koji se šalje na D/A pretvornik.

II.1. Brojači vremena u mikroupravljaču STM32F407

Vremenski brojači (engl. *timer*) je periferija mikroupravljača s kojom je moguće ostvariti različite operacije poput generiranja vremenske baze, brojanja impulsa, generiranja specijalnih signala poput PWM signala, mjerenje perioda vanjskih signala i sl. Generalno, vremenski brojači broje promjene takta koji se dovodi na njih, a koji može biti interni ili vanjski takt. Dodatno, ovaj takt može biti propušten kroz dijelitelj frekvencije (engl. *prescaler*). Na taj način je moguće promijeniti frekvenciju kojom vremenski brojač broji, a da se i dalje koristi isti ulazni takt. Osnovna karakteristika vremenskog brojača je broj bitova od kojih se sastoji brojač (16 ili 32 bita). Vrijednost do koje brojač broji je podesiva, nakon čega se brojač može resetirati na nulu i opet započinje brojati.

STM32F407VG sadrži relativno složenu periferiju za vremensko brojanje koja omogućava različite operacije. Ovaj mikroupravljač sadrži ukupno 14 vremenskih brojača koji se mogu podijeli na tri tipa:

1. Napredni upravljački vremenski brojači (engl. *advanced-control timers*),
2. Vremenski brojači opće namjene (engl. *general-purpose timers*), i
3. Osnovni vremenski brojači (engl. *basic timers*).

U tablici 4.1 dani su svi dostupni vremenski brojači u mikroupravljaču STM32F407VG te su izdvojene najvažnije karakteristike poput skraćenog naziva, rezolucije, načina brojanja, raspona dijelitelja ulaznog takta, podrška za izravan pristup memoriji (engl. *Direct Memory Access* - DMA), broj kanala koji podržavaju *capture/compare* operacije te izvor takta i njegovu maksimalnu vrijednost.

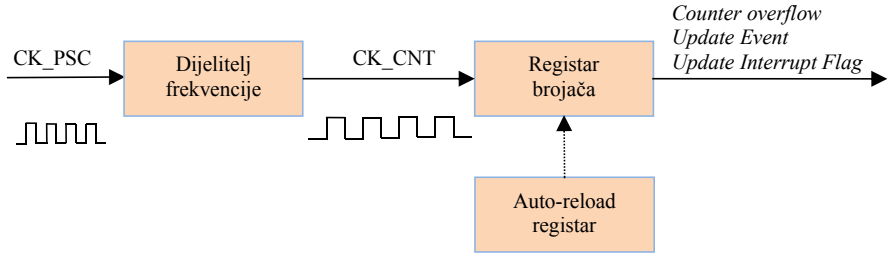
Tablica 4.1 Vremenski brojači u mikroupravljaču STM32F407VG

Tip	Naziv	Rezolucija	Brojanje	Dijelitelj	DMA	Capture / Compare kanali	Max. takt [MHz]	Izvor takta
Napredni upravljački	TIM1 TIM8	16-bita	Gore, Dolje, Gore/Dolje	1-65536	Da	4	168	APB2
Opće namjene	TIM2 TIM5	32-bita	Gore, Dolje, Gore/Dolje	1-65536	Da	4	84	APB1
	TIM3 TIM4	16-bita	Gore, Dolje, Gore/Dolje	1-65536	Da	4	84	APB1
	TIM9	16-bita	Gore	1-65536	Ne	2	168	APB2
	TIM10 TIM11	16-bita	Gore	1-65536	Ne	1	168	APB2
	TIM12	16-bita	Gore	1-65536	Ne	2	84	APB1
	TM13 TIM14	16-bita	Gore	1-65536	Ne	1	84	APB1
Osnovni	TM6, TIM7	16-bita	Gore	1-65536	Da	0	84	APB1

Kako bi se mogao pravilno podesiti vremenski brojač, potrebno je poznavati vrijednost ulaznog takta. Izvor takta kod vremenskih brojača je napredna sabirnica za periferiju (engl. *Advanced Peripheral Bus* - APB) koja može biti niske brzine (APB1) ili visoke brzine (APB2). Takt APB1 i APB2 izvodi se iz sistemskog takta i koji se propušta kroz

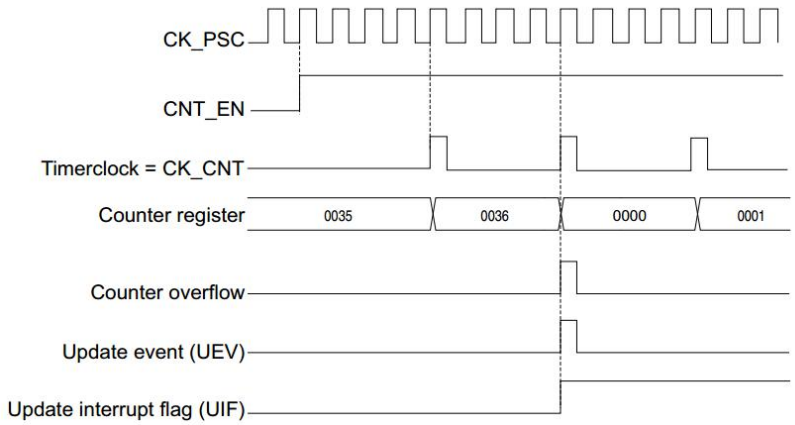
odgovarajuće dijelitelje za APB1 i APB2. Ovaj takt se dodatno množi s dva ako su navedeni dijelitelji različiti od 1 te se dovodi na vremenski brojač. Na primjer, uz maksimalni iznos sistemskog takta od 168 MHz za STM32F407VG mikroupravljač, te dijelitelje 4 i 2 za sabirnice APB1 i APB2, maksimalni takt za vremenske brojače koji su spojeni na APB1 iznosi (168/4) * 2 MHz , odnosno (168/2) * 2 MHz za brojače koji su spojeni na APB2. Za detalje pogledajte RM0090 str. 152 i 216.

Na slici 4.1. je blokovski prikazan princip rada vremenskog brojača. Interni takt koji se dovodi na vremenski brojač označen je s CK_PSC. Navedeni takt propušta se kroz dijelitelj frekvencije te nastaje takt CK_CNT koji je iste ili niže frekvencije. Sa svakom promjenom takta CK_CNT vrijednost registra brojača se inkrementira. Registar brojača se inkrementira do vrijednosti u *auto-reload* registru nakon čega se generira *overflow event* (OV) i *update event* (UEV). Zatim se brojač automatski resetira na 0 i započinje ponovo brojati.



Sl. 4.1 Blok prikaz rada vremenskog brojača.

Primjer rada vremenskog brojača u vremenu dan je na slici 4.2 pri čemu se koristi dijelitelj frekvencije iznosa 4, a vrijednost *auto-reload* registra je postavljena na 36. Primijetite da se vrijednost brojača inkrementira samo kada je vremenski brojač omogućen (signal CNT_EN). Registar brojača, *auto-reload* registar i registar dijelitelja frekvencije mogu se softverski čitati i mijenjati tijekom rada.



Sl. 4.2 Primjer rada vremenskog brojača vremenskim dijagramom.

II.1.1 Vremenska baza

Osnovna primjena vremenskog brojača je generiranje vremenske baze. U tom se slučaju s određenom (željenom) frekvencijom pojavljuju UEV. Očito da period između dva UEV ovisi o *auto-reload* vrijednosti i taktu brojača. Stoga se ove vrijednosti moraju podesiti ako se želi generirati UEV s određenom frekvencijom odnosno periodom. Pri tome mora se uzeti u obzir da je maksimalna vrijednost brojača jednaka 2^{32} kod TIM2 i TIM5 odnosno 2^{16} kod ostalih vremenskih brojača, a vrijednost dijelitelja frekvencije je od 1 do (2^{16}+1).

Na primjer želi se generirati UEV svakih 500 ms pomoću TIM9. Frekvencija internog takta ovog vremenskog brojača iznosi f_{CK_PSC}=168MHz. To znači da se registar brojača inkrementira svakih 1/(168MHz) = 5.95 ns. Bez korištenja dijelitelja frekvencije PSC (tj. ako je njegova vrijednost jednaka 1) nije moguće odrediti 16-bitnu vrijednost *auto-reload* registra tako se UEV događa svakih 500 ms. Međutim, ako se postavi vrijednost dijelitelja frekvencije na 3360, tada se inkrementiranje registra brojača odvija svakih 3360/(168MHz) = 20 us. Tada je vrijednost *auto-reload* registra koja će uzrokovati UEV svakih 500ms jednaka 500ms/20us = 25000. Ovu vrijednost moguće je zapisati u 16-bitni *auto-reload* registar brojača TIM9.

Pokretanjem vremenskog brojača moguće je realizirati različite vremenske operacije u programu bilo prozivanjem vremenskog brojača (engl. *polling*) ili omogućavanjem zahtjeva za prekidom kod pojave UEV. U tablici 4.2 dan je primjer konfiguracije TIM9 tako da se generira zahtjev za prekidom svakih 500 ms. Kao i u radu s ostalom periferijom, najprije je potrebno aktivirati takt za TIM9, zatim podesiti vrijednost dijelitelja frekvencije i *auto-reload* registra.

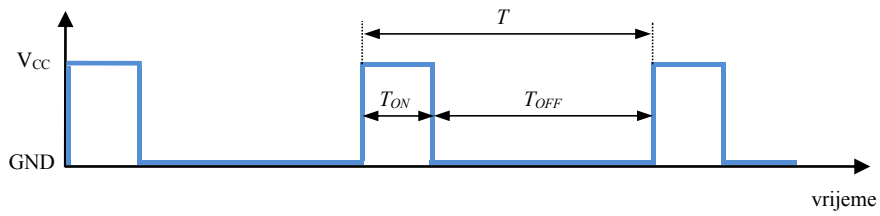
Nadalje, omogućuje se generiranje prekida u slučaju osvježavanja registra brojača (UEV), definira se prioritet prekida i omogućava u NVIC. Primijetite da je vrijednost dijelitelja frekvencije umanjena za 1 (jer se na tu vrijednost interno dodaje 1, vidi RM0900 str. 682). Također, ako se želi generirati zahtjev za prekidom s pojavom 25000-og brida takta brojača, tada vrijednost *auto-reload* registra treba biti za jedan manja (vidi sliku 4.2).

Tablica 4.2 Podešavanje TIM9 i odgovarajući rukovatelj prekida.

```
#include "stm32f4xx.h"

//omogući takt za TIM9
RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM9, ENABLE);

//podešavanje TIM9 - UEV svakih 500 ms
TIM_TimeBaseInitTypeDef TIM9InitStructure;
TIM9InitStructure.TIM_Prescaler = 3359;                 //vrijednost dijelitelja frekvencije
TIM9InitStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM9InitStructure.TIM_Period = 24999;                   //vrijednost auto-reload registra
TIM9InitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInit(TIM9, &TIM9InitStructure);

TIM_ITConfig(TIM9, TIM_IT_Update, ENABLE);
TIM_Cmd(TIM9, ENABLE);

//postavi prioritet TIM9 prekida
NVIC_SetPriority(TIM1_BRK_TIM9_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),1,0));

//omogući TIM9 prekid
NVIC_EnableIRQ(TIM1_BRK_TIM9_IRQn);

void TIM1_BRK_TIM9_IRQHandler(void)
{
     /* kod prekidne rutine
     ...
     ...
     */
     TIM_ClearITPendingBit(TIM9, TIM_IT_Update);
}
```

II.1.2 Generiranje pulsno-širinsko moduliranog signala

Pulsno-širinski modulirani (PWM) signal često se koristi u ugradbenim računalnim sustavima u različitim primjenama (regulatori napona, pogonski uređaji motora, audio tehnika, u telekomunikacijama i sl.). Na slici 4.3. prikazan je PWM signal. PWM signal je periodički (digitalni) signal pri čemu se jedna perioda signala (T) sastoji od vremena u kojem je signal u visokoj logičkoj (T_{ON}) razini plus vrijeme u kojem je signal u niskoj logičkoj razini (T_{OFF}). Odnos vremena u kojem je signal u visokoj logičkoj razini i perioda PWM signala naziva se *duty cycle.* Na slici je dan primjer signala koji ima duty cycle 25%.



Sl. 4.3 Primjer PWM signala s vrijednošću *duty-cyclea* 25%.

Za ilustraciju upotrebe PWM signala iskoristit će se ugrađena LED na razvojnoj ploči. Promjenu intenziteta osvjetljenja ugradbene LED moguće je postići slanjem PWM signala na pin na koji je spojena LED dioda. Ako je frekvencija PWM signala dovoljno visoka (nekoliko stotina herca), dobija se dojam kao da je na priključcima LED srednja vrijednost napona PWM signala koja ovisi o *duty-cycle*u PWM signala. Na taj način se mijenjanjem *duty-cycle*-a može postići različita razina osvjetljenja ugradbene LED. Na istom principu se zasniva i upravljanje brzinom vrtnje istosmjernog motora i općenito aplikacije u kojima je potrebno upravljati količinom snage kojom se opskrbljuje uređaj od interesa.

Za primjer generiranja PWM signala uzet će se vremenski brojač TIM4. Prema tablici 4.1, TIM4 ima 4 kanala na kojima je moguća opcija generiranja PWM signala. Ovi kanali spojeni su na GPIO pinove prema tablici 4.3. Frekvencija PWM signala ovisi o taktu brojača i *auto-reload* vrijednosti registra, na isti način kao i kod generiranja vremenske baze. *Duty-cycle* PWM signala definiraje vrijednost u registru TIM4_CCRx (pri čemu je x oznaka kanala).

Tablica 4.3 TIM4 kanali i pripadni GPIO pinovi			
CH1	CH2	CH3	CH4
PD12, PB6	PD13, PB7	PD14, PB8	PD15, PB9

U tablici 4.3 dan je primjer podešavanja vremenskog brojača TIM4 u svrhu generiranja PWM signala na pinu PD14 na koji je spojena ugrađena LED. Najprije se pin PD14 konfigurira tako da ima alternativnu funkciju te se povezuje s TIM4. Zatim je potrebno aktivirati takt za TIM4 i definirati vremensku bazu koja će predstavljati period PWM signala. U ovom primjeru PWM signal ima frekvenciju 2 kHz odnosno period iznosa 500 us. Budući da je TIM4 spojen na APB1 te je ulazni takt 84 MHz, može se koristiti dijelitelj frekvencije iznosa 1 te je tada vrijednost *auto-reload* registra jednaka 500us*84MHz/(1) = 42000. Nadalje, potrebno je podesiti karakteristike *Output Compare* načina rada. Odabire se PWM način rada te se za početnu vrijednost duty-cyclea postavlja vrijednost 0. Vrijednost duty-cyclea može se mijenjati podešavanjem vrijednosti u registru TIM4_CCR3.

Tablica 4.3 Generiranje PWM signala s vremenskim brojačem TIM4 na pinu PD14.

```
#include "stm32f4xx.h"

//uključivanje takta za port D
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

//koristi se alternativna funkcija pina PD14
GPIO_InitTypeDef GPIO_InitStructPWM;
GPIO_InitStructPWM.GPIO_Pin = GPIO_Pin_14;
GPIO_InitStructPWM.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructPWM.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructPWM.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructPWM.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_Init(GPIOD, &GPIO_InitStructPWM);

// alternativna funkcija pina PD14 je PWM signal s TIM4
GPIO_PinAFConfig(GPIOD, GPIO_PinSource14, GPIO_AF_TIM4);

//omogući takt za TIM4
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

//podešavanje TIM4 vremenske baze - frekvencija PWM signala je 2 kHz
TIM_TimeBaseInitTypeDef TIM4_InitStruct;
TIM4_InitStruct.TIM_Prescaler = 0;
TIM4_InitStruct.TIM_CounterMode = TIM_CounterMode_Up;
TIM4_InitStruct.TIM_Period = 41999;
TIM4_InitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInit(TIM4, &TIM4_InitStruct);
TIM_Cmd(TIM4, ENABLE);

//podešavanje karakteristika PWM signala
TIM_OCInitTypeDef TIM4_OCInitStruct;
TIM4_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
TIM4_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
TIM4_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
TIM4_OCInitStruct.TIM_Pulse = 0;                        //početna vrijednost duty-cyclea je 0%

TIM_OC3Init(TIM4, &TIM4_OCInitStruct);                  //PWM signal na CH3
TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);

TIM4->CCR3 = (int)(42000*0.1) - 1;              //duty-cycle iznosi 10%
TIM4->CCR3 = (int)(42000*0.5) - 1;              //duty-cycle iznosi 50%
```

II.2 D/A pretvornik

Mikroupravljač STM32F407VG sadrži dva 12-bitna digitalno-analogna D/A pretvornika koji pretvaraju digitalni signal u analogni naponski signal koji je dostupan na odgovarajućim pinovima mikroupravljača (PA4 i PA5). D/A pretvornik može raditi u 8-bitnom i 12-bitnom režimu rada, podržava različite izvore kojima se može pokrenuti D/A pretvorba i DMA (engl. *Direct Memory Access*).

Digitalna riječ se linearno pretvara u analogni naponski signal između 0 i V_{REF+}. Izlazni naponski signal D/A pretvornika jednak je: DAC_OUTx = V_{REF+} x DORx/4096 pri čemu je DOR izlazni podatkovni registar D/A pretvornika. Oba D/A pretvornika sadrže izlazni *buffer* kojeg je moguće zasebno aktivirati kako bi se smanjila izlazna impedancija (u suprotnom je izlaz direktno spojen na R-2R otporničku mrežu D/A pretvornika). Najjednostavniji način započinjanja D/A pretvorbe je zapisivanjem odgovarajuće vrijednosti u 16-bitni DAC_DHR registar pri čemu je

moguće koristiti tri različita formata zapisa: 8-bitni desno poravnati podatak, 12-bitni lijevo poravnati podatak i 12-bitni desno poravnati podatak. Iz ovog registra se vrijednost prebacuje izlazni podatkovni registar nakon tri APB1 takta.

U tablici 4.4 dan je primjer podešavanja i zapisivanja vrijednosti u drugi D/A pretvornik pomoću SPL. Kako bi dobivena analogna vrijednost bila dostupna na pinu mikroupravljača, najprije je potrebno podesiti odgovarajući pin mikroupravljača tako da ima alternativnu funkciju. Drugi D/A pretvornik spojen je na PA5 pin mikroupravljača. Nadalje, potrebno je aktivirati takt za drugi D/A pretvornik te izvršiti željenu konfiguraciju pomoću `DAC_Init` funkcije te ga aktivirati pomoću `DAC_Cmd` funkcije. D/A pretvorbu moguće je pokrenuti pomoću funkcije `DAC_SetChannel2Data`. Ova funkcija kao prvi argument prima način poravnanja podataka, dok je drugi argument digitalna vrijednost koja se želi pretvoriti u odgovarajuću analognu vrijednost.

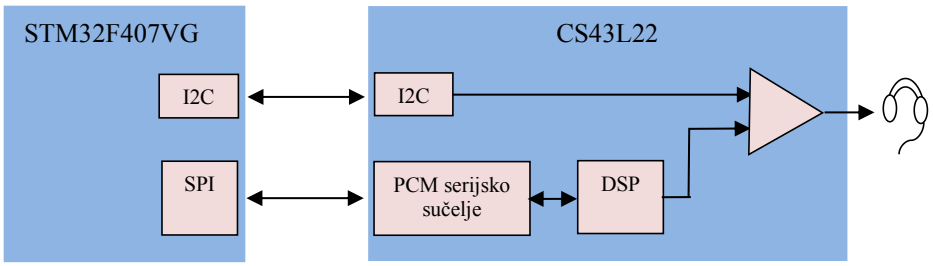
Tablica 4.4 Podešavanje drugog D/A pretvornika

```
//omogući takt za port A
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

//koristi se alternativna funkcija pina PA5
GPIO_InitTypeDef GPIO_DAC_InitStruct;
GPIO_DAC_InitStruct.GPIO_Pin = GPIO_Pin_5;
GPIO_DAC_InitStruct.GPIO_Mode = GPIO_Mode_AN;
GPIO_DAC_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOA, &GPIO_DAC_InitStruct);

//omogući takt za D/A pretvornik
RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

//podesi drugi D/A pretvornik
DAC_InitTypeDef DAC_InitStruct;
DAC_InitStruct.DAC_Trigger = DAC_Trigger_None;
DAC_InitStruct.DAC_WaveGeneration = DAC_WaveGeneration_None;
DAC_InitStruct.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
DAC_Init(DAC_Channel_2, &DAC_InitStruct);

//omogući drugi D/A pretvornik
DAC_Cmd(DAC_Channel_2, ENABLE);

//pokreni D/A pretvorbu na drugom D/A pretvorniku
DAC_SetChannel2Data(DAC_Align_12b_R, 2305);
```

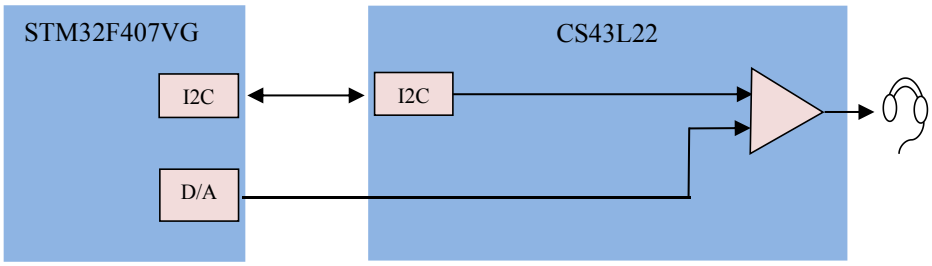
II.3 Generiranje zvučnog signala

STM32F407 Discovery razvojna ploča sadrži vanjski audio D/A pretvornik CS43L22 namijenjen za PDA, igraće konzole i sl. CS43L22 je visoko integrirani stereo D/A pretvornik s integriranim pojačalom klase D i stereo pojačalom za slušalice. Na taj način je moguće generirati zvuk koji je dostupan na 3.5mm konektoru koji se nalazi na Discovery razvojnoj ploči.

Podešavanje i upravljanje CS43L22 provodi se preko I2C komunikacijskog sučelja. Dva osnovna načina rada CS43L22 su dana na slici 4.4 a) i b) po pitanju razmjene signala s mikroupravljačem. Na slici 4.4 a) PCM signal se D/A pretvornik šalje pomoću serijskog sučelja pri čemu se koristi I2S (Inter-IC Sound) audio protokol. Na slici 4.4 b) koristi se interni D/A pretvornik mikroupravljača za dobivanje analognog signala, a CS43L22 se podešava tako da propušta ulazni analogni signal na pojačalo za slušalice s mogućnošću pojačavanja/stišavanja signala (opcija "Analog Passthrough"). Shemu spoja između STM32F407VG i CS43L22 moguće je pronaći u "UM1472 User manual Discovery kit with STM32F407VG MCU".



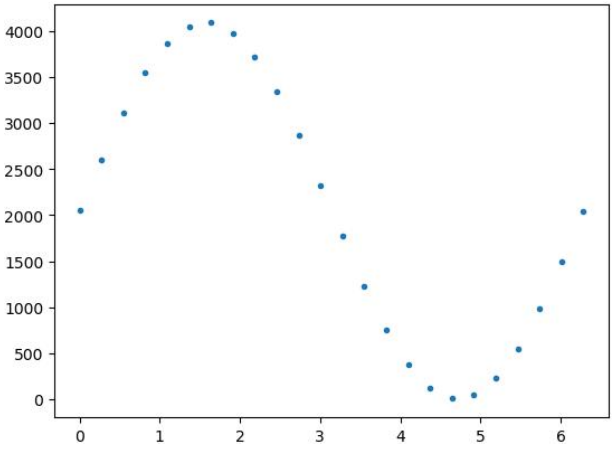
a)



b)

Sl. 4.4 Načini razmjene podataka između STM32F407VG i CS43L22.

Najčešći način kodiranja zvučnog signala je impulsno kodna modulacija (engl. *Pulse-code modulation* - PCM) pri čemu se zvučni signal uzorkuje u pravilnim vremenskim intervalima, a svaki uzorak se kvantizira na najbližu kvantizacijsku razinu. Primjer kvantizacije sinusnog valnog oblika u slučaju korištenja 12 bitnog D/A pretvornika dan je na slici 4.5 pri čemu se za jednu periodu signala koristi 24 mjerna uzorka. Ovakav signal se može slati pomoću ugrađenog D/A pretvornika na na CS43L22 kao što je dano na slici 4.4 b) pri čemu frekvencija dobivenog zvučnog signala ovisi o veličini perioda između dva uzastopna uzorka signala (odnosno o vremenskom periodu između prvog i zadnjeg uzorka signala).




```
volatile const uint16_t sine_wave[24] =
{2048, 2593, 3098, 3526, 3843, 4028, 4065,
3953, 3700, 3324, 2853, 2323, 1772, 1242,
771, 395, 142, 30, 67, 252, 569, 997, 1502,
2047};
```

Sl. 4.5 Primjer diskretiziranog i kvantiziranog sinusnog signala.

III. Priprema za vježbu

1. Razmislite kako vrijednost dijelitelja utječe na vremensku rezoluciju brojanja, a kako na maksimalno vrijeme koje je moguće izmjeriti vremenskim brojačem.
2. O čemu sve može ovisiti odabir frekvencije PWM signala, npr. kod upravljanja brzinom vrtnje istosmjernog motora?
3. Koliko iznosi vrijeme pretvorbe D/A pretvornika koji se nalazi u mikroupravljaču STM32F407VG?

IV. Rad na vježbi

1. Napišite program koji će periodički uključivati i isključivati dvije ugradbene LED korištenjem vremenskih brojača. Period uključivanja/isključivanja prve LED treba biti 1000 ms, a period druge treba biti 250 ms. Za uključivanje/isključivanje prve LED iskoristite TIM9, a za uključivanje/isključivanje druge LED iskoristite TIM2.

2. Napišite program koji će postepeno povećavati jačinu svjetlosti LED koja je na razvojnoj ploči. Kada dostigne maksimalnu svjetlinu LED, program treba postepeno smanjivati jačinu svjetlosti LED. Postupak se treba ponavljati pri čemu je jedan ciklus pojačavanja/smanjivanja osvjetljenja treba trajati ukupno 4 sekunde. Za definiranje razine svjetlosti ugrađene LED generirajte pulsno-širinski moduliran signal pomoću brojača vremena.

3. Napišite program koji će generirati zvučni signal frekvencije 1 kHz na temelju projekta naziva "LV4_sound_template" koji se nalazi na adresi https://gitlab.com/rgrbic/UGRS_LV. Ovaj projekt sadrži "analog passthrough" konfiguraciju CS43L22 D/A pretvornika. Na taj način se analogni signal koji se generira pomoću prvog D/A pretvornika mikroupravljača STM32F407VG (pin PA4) šalje izravno na pojačalo za slušalice kroz

CS43L22. Za generiranje odgovarajuće vremenske baze koristite TIM4. Pomoću slušalica provjerite ispravnost rješenja.

4. Na temelju rješenja prethodnog zadatka generirajte DTMF signal za telefonsku brojku 1. Ovaj signal je mješavina dva sinusna signala frekvencije 697 Hz i 1209 Hz. Pomoću slušalica provjerite ispravnost rješenja. Usporedite dobiveni signal sa signalom kojeg dobivate pritiskom tipke 1 na vašem mobilnom telefonu.

V. Za one koji žele znati više

CS43L22 ima mogućnost pojačavanja i stišavanja zvučnog signala namještanjem pojačanja preko I2C komunikacijskog sučelja. Na temelju datasheeta CS43L22 te projekta "LV4_sound_template" napišite program koji će omogućiti pojačavanje/stišavanje zvučnog signala u slušalicama na temelju zakreta potenciometra koji je spojen na STM32F407VG mikroupravljač.