Elektrotehnički fakultet, Univerzitet u Beogradu

Predmet: Sistemi u realnom vremenu

- Projektni zadatak 28 -

Sistem za merenje vrednosti sa potenciometara realizovan na MSP430 mikrokontroleru

Profesori:

Ivan Popović

Haris Turkmanović

Student:

Aleksandar Milošević 658/17

Beograd 2022.

Sadržaj

Uvod

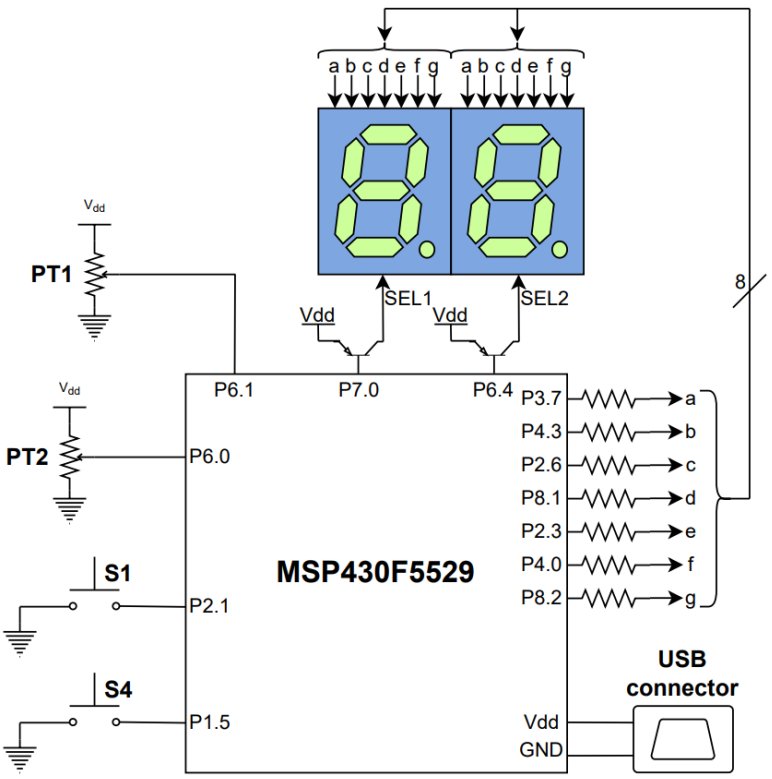
- Hardver razvojnog okruženja

Za realizaciju ovog sistema, na raspolaganju su bili:

- MSP430F5529LP razvojna platforma (slika 1)
- shield sa dodatnim periferijama (potenciometri, LED, 7-seg. displej, tasteri) (slika 4)

Slika 1 - MSP430F5529 mikrokontroler

MSP430F5529LP razvojna platforma sadrži MSP430F5529 mikrokontroler iz familije MSP430 mikrokontrolera (MCU - Microcontroller Unit) koje proizvodi kompanija Texas Instruments. On poseduje 16-bitni procesor (CPU - Central Processing Unit) sa redukovanim setom instrukcija (RISC - Reduced Instruction Set Computer), sa Von-Neumann-ovom arhitekturom sa dve magistrale MAB i MDB koje dele isti adresni prostor. Dakle on u sebi poseduje 27 osnovnih instrukcija i pruža mogućnost 7 vrsta adresiranja. Sam CPU sadrži 16 internih registara, od kojih su praktično 12 njih opšte (R4 do R15) namene, dok se dva koriste kao PC (Program Counter), SP (Stack Pointer), i još dva kao generatori konstanti, ali je takođe dozvoljeno čitanje i upis u njih. Takođe se ubraja u Low Power MCU koji imaju smanjenu potrošnju i omogućava pet Low Power režima. Memorijska mapa sadrži Flash/ROM (u koju se može smeštati i programski kod i podaci), RAM (u koju se smeštaju privremeni podaci) i adrese namenjene za pristup internim periferijama mikrokontrolera. Ukoliko je sam programski kod smešten u okviru RAM memorije sistem ima skoro duplo manju potrošnju.



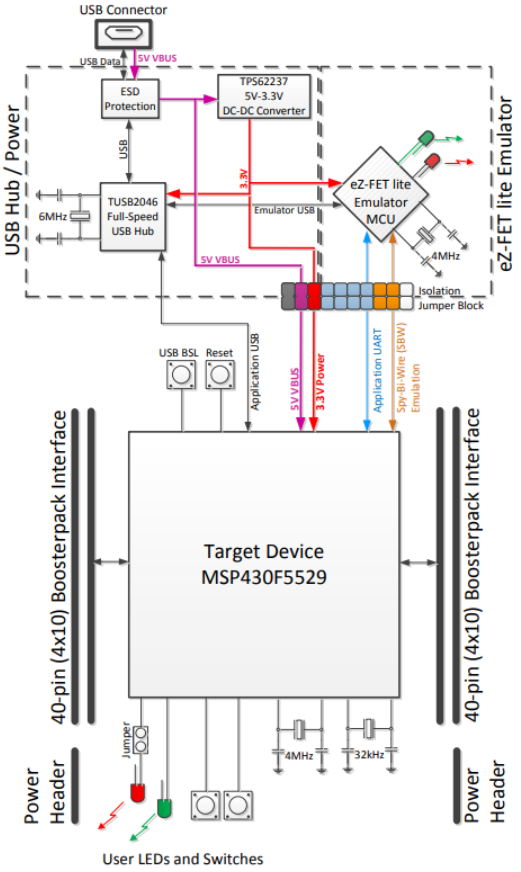
Slika 2 - Blok šema sistema

Od pomenutih eksternih periferija, koriste se tasteri S1 i S4, potenciometri PT1 i PT2, 7-segmentni displej (slike 1, 2 i 3).

Svim periferijama upravlja i očitava ih mikrokontroler. Sistem se napaja preko USB interfejsa sa naponom od $5V$. Potenciometri su povezani preko odgovarajućih pinova mikrokontrolera na AD konvertor internu periferiju MCU-a. ADC je tako povezan na potenciometre, da se rezultati sa PT1 snimaju na kanalu 1, a rezultati sa PT2 snimaju na kanalu 0 AD konvertora.

- Softver razvojnog okruženja

Sam proces razvoja ovakvog sistema podrazumeva cross-development razvoj softvera. Cross-development omogućava da se softver razvija (piše, build-uje, kompajlira) na host sistemu, a potom putem USB ili nekog drugog interfejsa se ceo programski kod prenosi (smešta u odgovarajuću memoriju) na target sistemu. Host sistem je sistem na kome se razvija kod (u ovom slučaju PC računar), a target sistem je sistem na kome se pokreće taj kod (u ovom slučaju MSP430F5529 mikrokontroler).



Slika 3 - Blok šema MSP430F5529 Launch Pad-a

Za realizaciju softvera sistema, korišćeno je Code Composer Studio 11.1 (CCS11.1) integrisano razvojno okruženje (IDE - Integrated Development Environment), kompanije Texas Instruments, bazirano na *eclipse* platformi. Ono omogućava cross-development razvoj (razvoj programskog koda na PC računaru i prenos putem odgovarajućeg interfejsa na sistem sa MCU-om). Softver je razvijan na bazi FreeRTOS operativnog sistema koji pruža mogućnosti za razvoj operativnog sistema u realnom vremenu. Free RTOS omogućava sve neophodne objekte za kreiranje operativnog sistema poput taskova, semafora (binarnih, brojačkih, mutex-a), redova sa porukama (queue-ovi), softverskih tajmera i grupe događaja. Takođe pruža i širok spektar API funkcija za rad sa tim objektima. Postoje i posebne API funkcije koje omogućavaju rad sa FreeRTOS objektima u okviru samih prekidnih rutina, koje nemaju blokirajući karakter, što nephodno da bi sistem ispravno funkcionisao. Sam Scheduler vrši raspoređivanje taskova po prioritetnom Round-Robin algoritmu. Takođe su u okviru samog projekta uključene i dodatne biblioteke, koje apstrahuju rad sa pojedinim hardverom i čine ga znatno jednostavnijim[3]. Sistemski tajmer RTOS-a, uz pomoć koga se realizuju i sistemski tikovi prema kojima scheduler raspoređuje taskove je realizovan pomoću tajmera A0 mikrokontrolera.

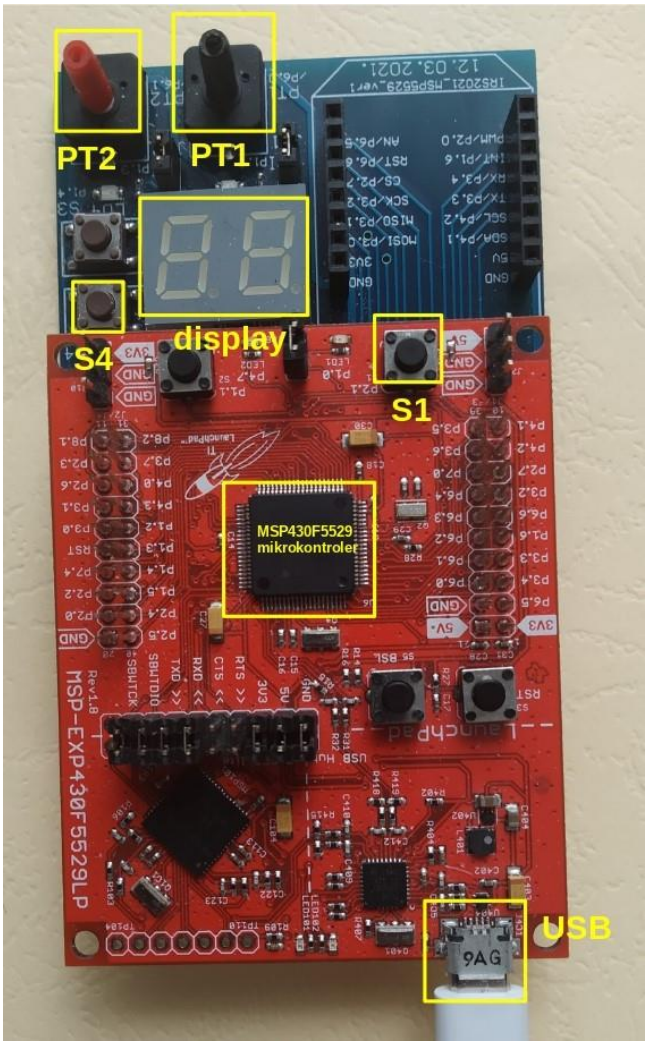
- Zadatak

Startuje se akvizicija sa kanala A0, A1 na svakih 1000ms pomoću taska Timer Task koji koristi vTaskDelayUntil. Potrebno je implementirati odloženu obradu prekida (deffered interrupt processing) AD konvertora tako što se rezultat konverzije u prekidnoj rutini upisuje u red sa porukama (Queue) i obaveštava se Task1 o prispeću nove poruke putem grupe događaja (EventGroup). Poruka treba da sadrži informaciju o kanalu koji je očitan i gornjih 9 bita rezultata AD konverzije. Task1 čuva poslednju očitanu vrednost za svaki kanal. Task2 ispituje tastere S1-S4 i na pritisak odgovarajućeg tastera obaveštava Task1 putem grupe događaja (EventsGroup) o kanalu čije očitane vrednosti rezultata konverzije treba da šalje tasku Task3. Svaki put kada stigne nova vrednost sa AD konvertora Task1 smešta odgovarajući podatak u red sa porukama na kojem čeka Task3. Task3 računa razliku između uzastopnih vrednosti očitanog kanala i prikazuje na multipleksiranom LED displeju.

- Jednostavno korisničko uputstvo

U početnom trenutku displej je ugašen. Pritiskom na jedan od prekidača (S1/S4) šalju se rezultati sa odgovarajućeg potenciometra (PT1/PT2) na obradu i ubrzo (za jedne sekunde ili brže) se prikazuju na displeju. Pritiskom na drugi taster ažurira se vrednost sa drugog potenciometra. Rezultati se prikazuju u heksa-decimlanom zapisu gde gde je leva cifra veće težine. Dakle da bi se vrednost sa nekog od potenciometara ažurirala, potrebno je da se pritisne odgovarajući taster, nakon čega će se periodično (svake sekunde) vrednost sa tog potenciometra ažurirati i tako ažurirana formirati razliku koja se ispisuje na 7-segmentnom displeju. Dok traje ažuriranje vrednosti sa jednog potenciometra, vrednost sa drugog potenciometra se ne ažurira, sve dok se ne pritisne taster zadužen za nju, od kada program ažurira nju, a vrednost sa prvog potenciometra ostaće zapamćena poslednja koja je bila očitana.

Sistem se dakle kao što je napomenuto napaja putem USB interfejsa sa 5V, a svu obradu i rad sa periferijama vrši mikrokontroler.



Slika 4 - Slika celokupnog sistema

Arhitektura softvera

- Uvod

Softver sistema se sastoji iz 4 taska (Task1, Task2, Task3, Timer Task), jedne prekidne rutine (ADC12ISR) i jednog softverskog tajmera (7SEG MUX Timer). Od mehanizama za sinhronizaciju između pomenutih objekata se korsiti grupa događaja (xEventsGroup) i jedan binarni semafor (xSem7segMux). Prenos podataka između taskova ili između prekidne rutine i taska se koriste queue-ovi (xADCvalQueueISR, xADCvalQueue).

Posebni tipovi (strukture) koji su korišćeni:

- queue_message_t - je struktura koja sadrži rezultat konverzije sa AD konvertora i takođe sadrži informaciju o tome sa kog kanala AD konvertora dolazi taj rezultat.

Spisak korišćenih objekata RTOS-a:

- taskovi
- binarni semafor
- queue-ovi
- softverski tajmer
- grupa događaja

Po pokretanju sistema, najpre se vrši početna inicijalizacija periferija mikrokontrolera.

- Početna inicijalizacija periferija mikrokontrolera

Proces inicijalizacije podrazumeva da se upisom u kontrolere periferija obavi njihovo postavljanje u željene režime. Tako se po pokretanju sistema (ili po njegovom resetu pritiskom na dugme RST na Launch Pad crvenoj platformi) se:

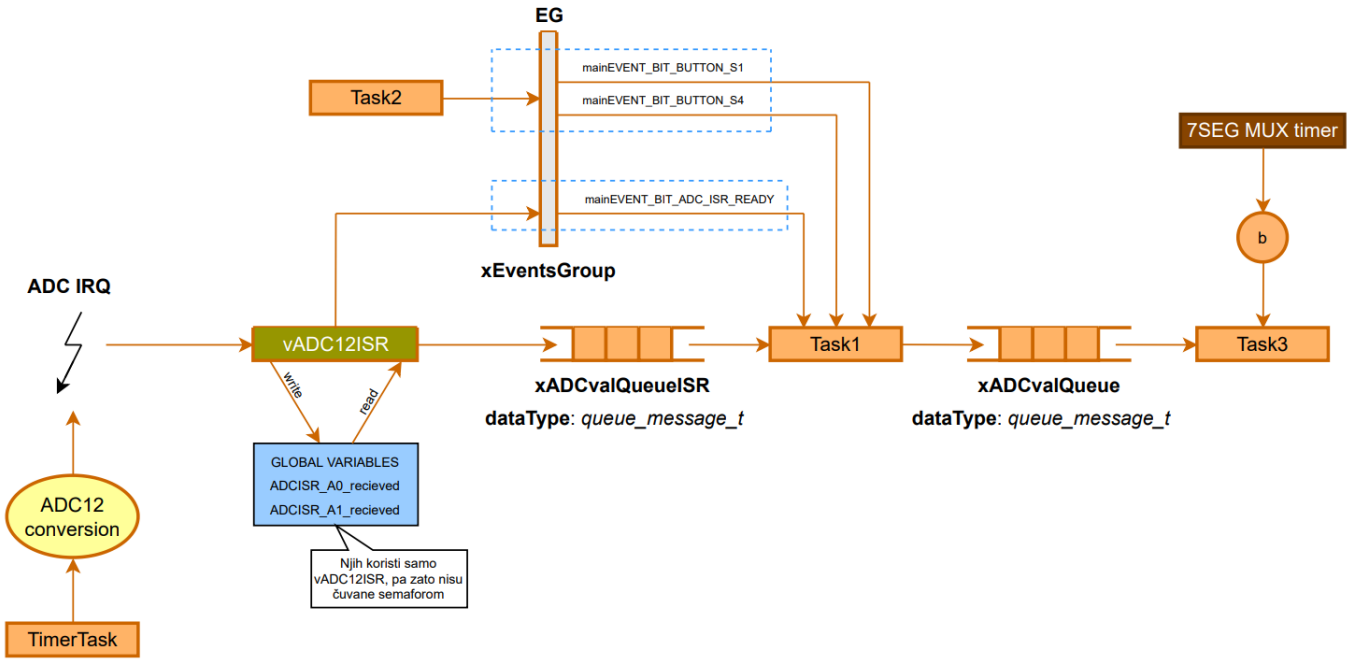
- zabranjuju maskirajući prekidi kako bi se neometano izvršio ovaj ceo proces
- zaustavlja Watch Dog tajmer (koji je po default-u uključen) jer se ne koristi u ovoj realizaciji
- podešavaju se pinovi P2.1 i 1.5, koji odgovaraju tasterima S1 i S4, da budu ulazni
- podešava se ADC12 konvertor da bude spreman da konvertuje vrednosti sa pinova P6.0 i P6.1 (na koje su povezani PT2 i PT1)
- podešavaju se pinovi na koje je povezan sedmo-segmentni displej
- ponovo se dozvoljavaju maskirajući prekidi po uspešnom inicijalizovanju internih periferija MCU-a

- Raspodeljenost poslova

Funkcije (poslove) koje sistem obavlja su podeljene u okviru taskova, tajmerskih callback f-ja i prekidnih rutina na sledeći način:

- Prekidna rutina AD konvertora - u okviru koje se rezultati AD konverzije šalju preko ogovarajućeg queue-a do taska 1 i on se obaveštava o tome putem grupe događaja.
- Task1 (prioriteta 4) - rezultate koje dobije iz queue-a pamti i ukoliko je pritisnut određeni taster (tu informaciju dobija od taska 2 putem grupe događaja) prosleđuje do taska 3 zajedno sa informacijom sa kog potenciometra dolaze rezultati.
- Task2 (prioriteta 1) - detektuje pritisak tastera i šalje signal tasku 1 putem gupe događaja o tome koji je od tastera pritisnut (S1/S4).
- Task 3 (prioriteta 2) - prihvata rezultate iz queue-a ako su stigli i ažurira rezultat. Potom prikazuje rezultat na displeju.
- Timer Task (prioriteta 3) - služi da pokreće periodično AD konverziju svake sekunde
- 7SEG MUX tajmerska callback f-ja - služi da oslobodi binarni semafor koji koristi Task 3 da bi vršio multipleksiranje displeja.

Na slici 5 su prikazani RTOS objekti i ostali elementi sistema koji ilustruju način komunikacije i razmene podataka između objekata u sistemu.



Slika 5 - Arhitektura softvera

- Redovi sa porukama (queue-ovi)

xADCvalQueueISR - je red sa porukama u koji se upisuje u okviru ADC prekidne rutine, a čita se iz njega u okviru taska 1. Poruke koje prihvata su tipa queue_message_t koji je već opisan u uvodnom delu. Sam red sa porukama može da prihvati maksimalno dve poruke. On prihvata poruku sa rezultatima potenciometra PT2, a potom potenciometra PT1.

xADCvalQueue - je red sa porukama u koji se upisuje u okviru taska 1, a čita se iz njega u okviru taska 3. Poruke koje prihvata su tipa queue_message_t koji je već opisan u uvodnom delu. Sam red sa porukama može da prihvati maksimalno jednu poruku. On prihvata poruku sa rezultatima (tasterom) odabranog potenciometra (PT2 ili PT1).

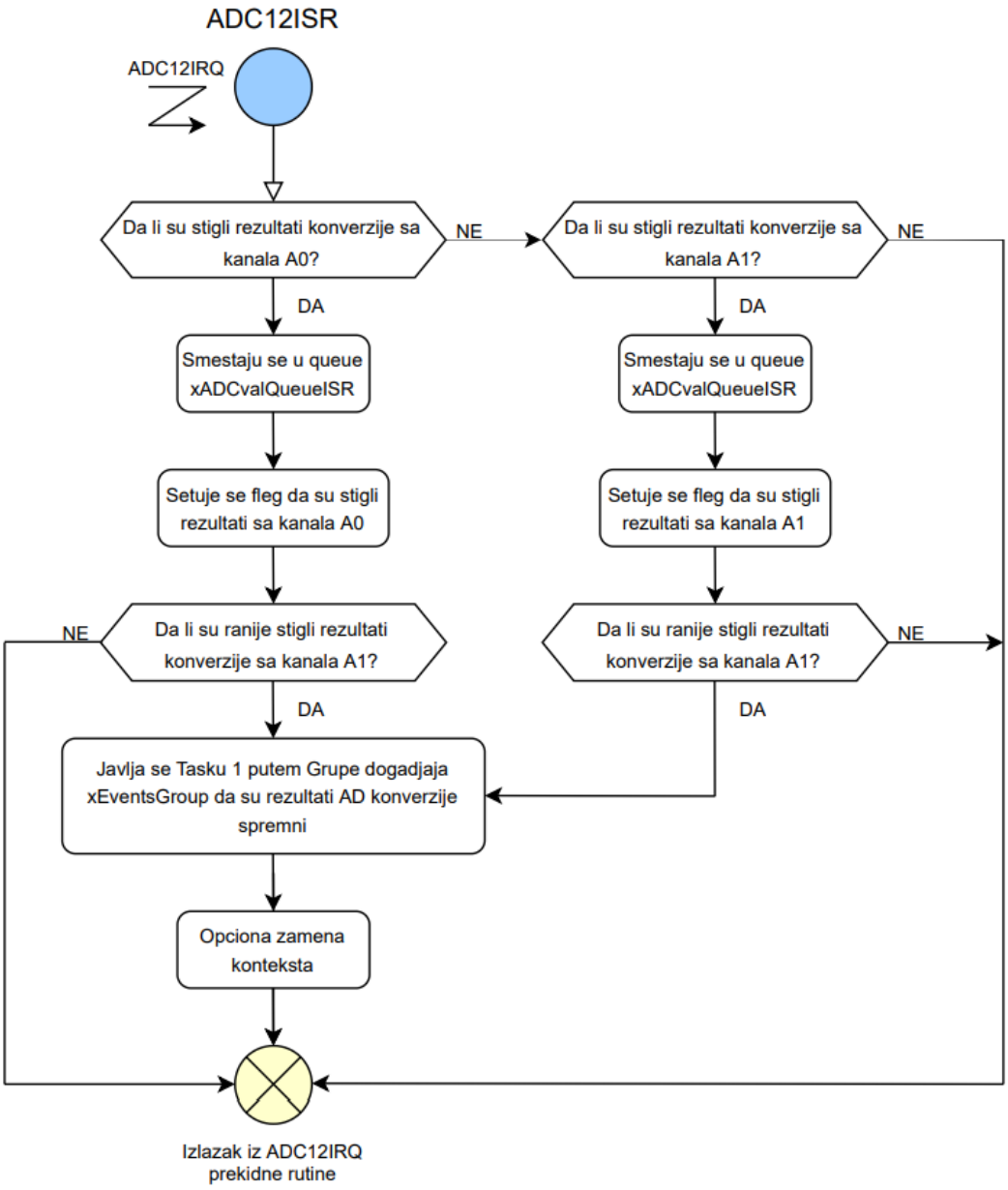
- Grupa događaja

xEventsGroup - je grupa sa događajima, veličine od 8 bita, od kojih se koriste praktično samo tri bita, da bi signalizirali tri različita događaja.

- mainEVENT_BIT_ADC_ISR_READY je bit iz xEventsGroup grupe događaja, koji se odnosi na događaj kada je xADCvalQueueISR u okviru ADC12ISR popunjen.
- mainEVENT_BIT_BUTTON_S1 je bit iz xEventsGroup grupe događaja, koji se odnosi na događaj kada je detektovan pritisak tastera S1
- mainEVENT_BIT_BUTTON_S1 je bit iz xEventsGroup grupe događaja, koji se odnosi na događaj kada je detektovan pritisak tastera S4

- Prekidna rutina AD konvertora

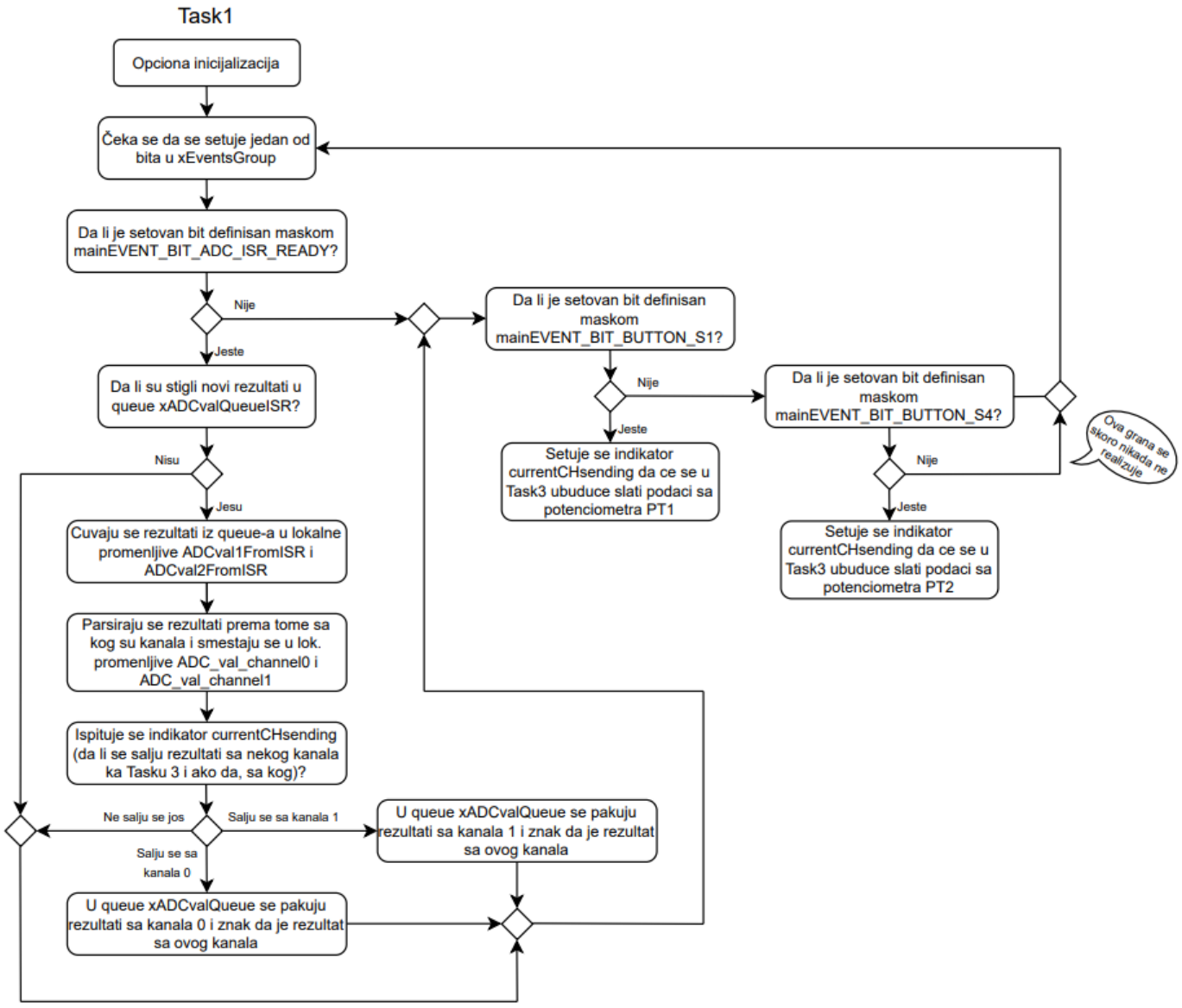
Nakon što AD konvertor završi sa konverzijom na kanalima A0 i A1 (koji odgovaraju PT2 i PT1), AD konvertor šalje zahtev za prekid, nakon koga program ulazi u prekidnu rutinu vezanu za AD konvertor. Prekidna rutina AD konvertora je na ulazu 54 u IVT (Interrupt Vector Table) tabeli. U samoj prekidnoj rutini se rezultati konverzije AD konvertora smešta u queue xADCvalQueueISR koji kasnije čita task 1. Kada se smeste rezultati konverzije sa oba potenciometra, setuje se odgovarajući bit (bit definisan makroom mainEVENT_BIT_ADC_ISR_READY) u grupi događaja xEventsGroup čime se obaveštava task 1 o prispeću rezultata AD konverzije. Takođe u okviru ove prekidne rutine se vrši i provera da li su rezultati sa oba potenciometra smešteni u xADCvalQueueISR, nakon čega se setuje odgovarajući bit u grupi događaja. Po izlasku iz prekidne rutine se poziva funkcija koja treba da inicira opcionu zamenu konteksta od strane Scheduler-a. Za to se koristi API f-ja portYIELD_FROM_ISR() koja podstiče taj proces. Na slici 6 je prikazana opisana algoritamska realizacija prekidne rutine ADC12ISR.



Slika 6 - Algoritamska realizacija ADC12ISR prekidne rutine

- Task 1

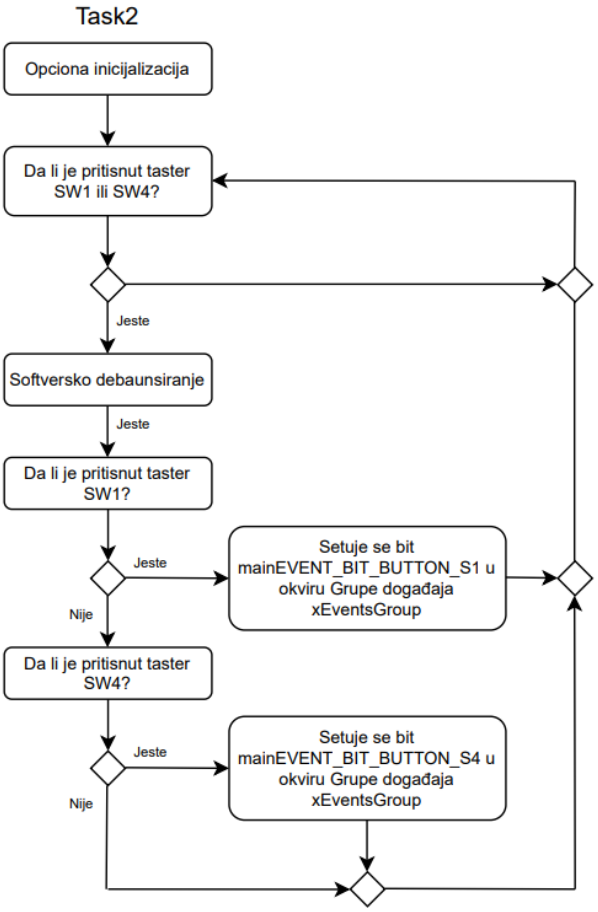
Task 1 ima ulogu da filtrira podatke koje dobija iz prekidne rutine AD konvertora i da ih prosledi tasku 3. Najpre task 1 se blokira na grupi događaja xEventsGroup i ostaje u blokiranom stanju sve dok se ne promeni neki od bita u njoj. Kada se ta promena desi, vrši se provera koji od događaja se desio. Ukoliko se desio događaj da je neki od tastera pritisnut, ta informacija se pamti u okviru taska 1. Ukoliko se desio događaj da je AD upakovao rezultate u AD queue, vrši se preuzimanje tih rezultata. Konačno, ka tasku 3 se šalju rezultati AD konverzije sa odgovarajućeg potenciometra. Na slici 7 prikazana je algoritamska realizacija taska 1.



Slika 7 - Algoritamska realizacija Taska 1

- Task 2

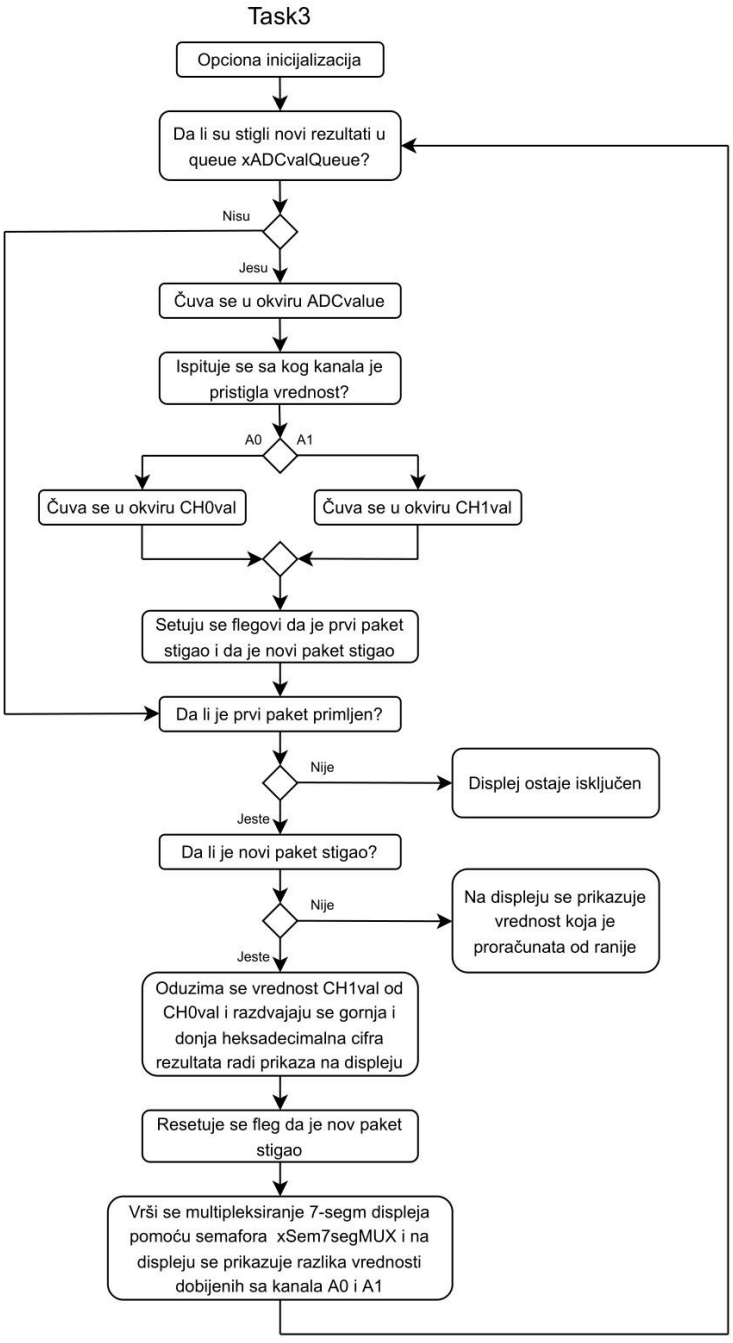
Task 2 ima ulogu da teketuje pritisak tastera i da javi tasku 1 koji je od tastera pritisnut. To se u okviru taska 2 čini mehanizmom poliranja. On svaki put kada dobije procesorsko vreme, ispituje da li je neki od tastera pritisnut. Ako detektuje pritisak tastera, on vrši softversko debaunsiranje u vidu kretanja u praznoj petlji. Debaunsiranje je potencialno bilo moguće bolje realizovati, preko softverskog tajmera, ili upotrebom nekog od hardverskih tajmera. Na slici 8 prikazana je algoritamska realizacija taska 2.



Slika 8 - Algoritamska realizacija Taska 2

- Task 3

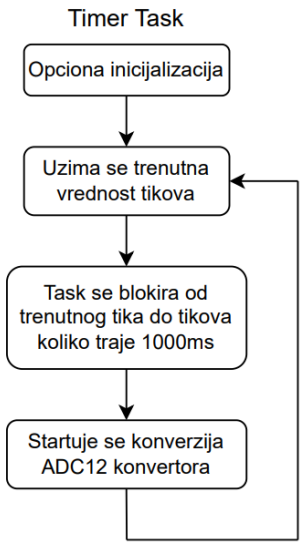
Task 3 ima ulogu da prikaže rezultat na displeju. On čita podatke iz queue-a xADCvalQueue (na neblokirajući način), vrši raščlanjivanje cifara i prikazuje ih na displeju. Sam prikaz na displeju se vrši multipleksiranjem, tako što se u toku 5ms prikaže prva cifra, pa se potom isključi prvi displej i na drugom se potom 5ms prikaže druga cifra. Taj postupak se ponavlja ciklično. Samo multipleksiranje displeja je realizovano uz pomoć binarnog semafora xSem7segMUX koga oslobađa softverski tajmer 7SEG MUX Timer. Po ispisivanju jedne od dve cifre, Task3 pokušava da uzme semafor xSem7segMUX na kome se zablokira 5ms (dok ga ne oslobodi 7SEG MUX Timer). Potom task nastavlja sa svojim regularnim izvršavanjem. Ovaj postupak multipleksiranja displeja nije kritičan za prihvatanje rezultata iz queue-a, pošto rezultati u queue pristižu svakih 1000ms, dok se multipleksiranje vrši svakih 5ms.



Slika 9 - Algoritamska realizacija Taska 3

- Timer Task

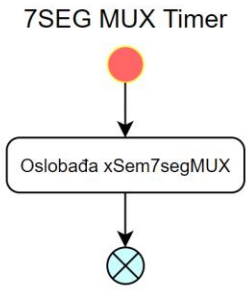
Timer Task ima ulogu da pokreće periodično akviziciju AD konvertora. On to čini tako što se u okviru njega poziva API funkcija vTaskDelayUntil(&xLastWakeTime,ADC_TICKS_PERIOD) koja blokira task do trenutka xLastWakeTime + ADC_TICKS_PERIOD. Vrednost xLastWakeTime je vreme pre dolaska do ove funkcije u kodu i ono se dobija API funkcijom xTaskGetTickCount() koja daje vrednost u tikovima do koliko je trenutno sistemskih tikova došao sistem. Vrednost ADC_TICKS_PERIOD određuje vrednost koliko sistemskih tikova će task biti blokiran. Nakon toga task se odblokirava i prelazi u stanje Ready (spreman za izvršavanje). Nakon što ovaj task ponovo dobije procesorsko vreme (pređe u stanje Running), startuje se AD konverzija na kanalima 0 i 1 koji odgovaraju potenciometrima PT2 i PT1. To se čini direktnim upisom odgovarajućih bita u kontrolni registar ADC periferije mikrokontrolera. Na slici 10 prikazana je algoritamska realizacija Timer taska.



Slika 10 - Algoritamska realizacija Timer Taska

- 7SEG MUX softverski tajmer

Ovaj softverski tajmer otkucava 5ms. Kada otkuca 5ms, poziva se callback f-ja koja oslobađa binarni semafor xSem7segMUX, koji se kasnije hvata task 3 i na osnovu čega nastavlja sa svojim izvršavanjem. U toku jedne periode izvršavanja taska 3, on dva puta pokušava da uzme ovaj semafor i blokira se svaki put na njemu, dok ga ovaj tajmer ne oslobodi. Na taj način se vrši multipleksiranje displeja (pogledati detaljnije realizaciju taska 3). Na slici 11 prikazana je algoritamska realizacija tajmerske callback funkcije prv7segMUXTimerCallback softverskog tajmera 7SEG MUX Timer. Kako je ovo hook f-ja koja se poziva iz *Daemon* sistemskog taska, ona ne sme u sebi da sadrži ni jednu blokirajuću API f-ju i poželjno je da se u okviru nje obavlja što manje posla, a da se sav mogući posao izmesti u okviru nekog taska. Tako je i ovde učinjeno. Ova f-ja samo oslobađa semafor, a sam postupak kontrole pinova u okviru multipleksiranja displeja se obavlja u okviru taska 3.



Slika 11 - Algoritamska realizacija 7SEG MUX Tajmera

Zaključak

Implementirani softver nije pokazao većih nepravilnosti u toku rada. Manje nepravilnosti i moguće izmene su pomenuta u nastavku pod naslovom Moguća poboljšanja.

Ova realizacija projekta zauzima u memoriji ukupno 16108B (približno oko 16KB) i to u sledećim razmerama:

- RAM: 5588B od 8192B (oko 68% ukupnog kapaciteta)
- FLASH1: 390B od 48000B (manje od 1% ukupnog kapaciteta)
- FLASH2: 10130B od 82936B (oko 12% ukupnog kapaciteta)

U okviru zauzete RAM memorije najviše mesta (oko 96%) je zauzela .bss sekcija u koju se smeštaju neinicijalizovani podaci (npr. static int i). Takođe u okviru same .bss sekcije najveći deo (oko 95%) zauzima heap memorija koja se koristi za dinamičku alokaciju memorije. Heap je podeljen u više manjih blokova fiksne veličine, gde se prilikom poziva funkcije za alokaciju memorije (npr malloc) alocira odgovarajući broj blokova memorije. Osim .bss sekcije, na RAM-u su još smešteni i .data sekcija koja sadrži inicijalizovane podatke (npr. int a = 3), i memorija namenjena za stek.

U okviru zauzete FLASH memorije, u FLASH1 delu se nalaze smeštene sekcije .text (sadrži izvršni kod prekidne rutine ADC12ISR), .const (sekcija u kojoj se nalaze smeštene konstante) i .cinit.

U FLASH2 delu se nalazi smeštena sekcija .text koja sadrži ostatak izvršnog koda (glavni izvršni kod).

Moguća poboljšalja

1) Izmena u okviru taska 1. Bilo bi bolje da se prvo vrši reakcija na događaj pritiska tastera, pre nego što se rezultati pošalju ka tasku 3. Zbog toga se u trenutnoj realizaciji dešava da na početni pritisak tastera (prvi pritisak nakon uspostavljanja napajanja sistema), displej kasni sa početnim prikazom čak 2s (umesto 1s).
2) Bilo bi moguće neki od preostalih tastera (S2/S3) iskoristiti da se sistem uvede u LPM kada nije neophodno da vrši akviziciju, a da ne mora da se ističe sa kabla za napajanje. Ovaj mikrokontroler ima 5 LPM režima, od kojih neki uvode sistem u dublje “sleep” režime, a neki u kraće. To je važno sa strane odziva sistema, odnosno buđenja sistema iz LPM režima (mikrokontroleru je potrebno više vremena da se probudi iz dubljeg LPM režima). Takođe bi bilo praktično u tom slučaju pridodati još jednu eksternu periferiju u vidu nekog RF prijemnika/primopredajnika, gde bi se sistemu moglo eksterno putem nekog daljinskog uređaja javiti da ode u LPM režim.
3) Dodatni 7-segmentni displeji bi proširili rezoluciju prikaza rezultata. Dodatno LCD displej bi bio još pogodniji u tom slučaju, gde bi bilo moguće ispisati i osnovne informacije o sistemu (od kog PT se trenutno rezultati šalju na obradu i slično)
4) Ukoliko je potrebno lako je moguće izmeniti da rezultati sa oba PT istovremeno se ažuriraju svake sekunde (umesto do sada samo jednog koji se ažurirao). Ta izmena se ogleda u proširenju queue-a koji prenosi podatke do Taska 3, gde bi se sada prenosili rezultati sa oba PT i bilo bi potrebno u okviru taska 3 omogućiti prihvatanje i drugog rezultata iz queue-a.
5) Na plavom shield-u sa dodatnim periferijama je pogrešno označen pin koji vodi do potenciometra PT2.

Spisak korišćenih API f-ja

API funkcije koje se koriste za kreiranje FreeRTOS objekata

- xTaskCreate - koristi se za kreiranje taska
- xTimerCreate - koristi se za kreiranje softverskog tajmera
- xEventGroupCreate - koristi se za kreiranje grupe događaja
- xQueueCreate - koristi se za kreiranje reda sa porukama
- xSemaphoreCreateBinary - koristi se za kreiranje binarnog semafora

API funkcije koje se koriste za rad sa postojećim objektima

- xTimerStart - pokreće softverski tajmer
- xSemaphoreGive - oslobađa semafor
- xSemaphoreTake - zauzima semafor
- xQueueReceive - čita jednu poruku iz reda sa porukama
- xQueueSendToBack - upisuje poruku u red sa porukama
- xEventGroupSetBits - postavlja odgovarajuće bite u grupi događaja
- xQueueSendToBackFromISR - upisuje poruku u red sa porukama, ali to vrši iz prekidne rutine
- xEventGroupSetBitsFromISR - postavlja bite u grupi događaja, ali to vrši iz prekidne rutine
- xEventGroupWaitBits - blokira task na grupi događaja, dok se neki od bita ne promeni

Specifične API funkcije

- vTaskStartScheduler - pokreće Scheduler
- taskDISABLE_INTERRUPTS - maskira prekide
- taskENABLE_INTERRUPTS - demaskira prekide
- portYIELD_FROM_ISR - vrši opcionu zamenu konteksta
- xTaskGetTickCount - preuzima trenutnu vrednost sistemskog tajmera u tikovima
- vTaskDelayUntil - blokira task od jednog trenutka, do zadatog intervala

Izmene u narednim verzijama

- U okviru Taska 1 je dodata zaštita pristupa deljenom resursu. U toku dok Task 1 čita podatak iz queue-a koji puni ADC prekidna rutina, moguće je da stigne novi rezultat i da se Task 1 u toku čitanja iz tog queue-a prekine. U tom slučaju bi se novi podatak upisao u queue, i kada bi se program nastavio, praktično bi Task 1 nastavio sa čitanjem iz queue-a ali bi ostatak podataka koje čita bio korumpiran. Zato je dodana zabrana prekida u okviru Taska 1 dok vrši čitanje iz queue-a kojim mu ADC12ISR prosleđuje rezultate konverzije. Prakitčno u ovakvoj realizaciji su zaista male šanse da do ovakve situacije dođe jer je perioda očitavanja rezultata sa potenciometra (1s) sasvim dovoljna ADC koji znatno brže od 1s izvrši konverziju i da MCU koji radi na visokoj učestanosti stignu da odrade sve programske poslove koji su neophodni u okviru jedne periode

Literatura

[1] Haris Turkmanović, Sistemi u realnom vremenu - Vežbe 2021/2022

[2] Ivan Popović, Sistemi u realnom vremenu - Slajdovi sa predavanja (http://tnt.etf.bg.ac.rs/~oe4srv/index_files/Page323.html)

[3] Haris Turkmanović, Primeri sa vežbi (https://github.com/turkmanovic/MSP430_FreeRTOS)

[4] Nenad Jovičić, Nikola Cvetković, Integrisani računarski sistemi - Materijali sa predavanja i vežbi

[5]https://www.freertos.org/fr-content-src/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf

[6]https://www.ti.com/lit/ug/slau208q/slau208q.pdf?ts=1649203301292&ref_url=https%253A%252F%252Fwww.google.com%252F

[7]https://www.ti.com/lit/ds/symlink/msp430f5529.pdf?ts=1649164280276&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430F5529

[8]https://www.ti.com/lit/ug/slau533d/slau533d.pdf?ts=1649178404008&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FMSP-EXP430F5529LP