Univerzitet u Beogradu

Elektrotehnički Fakultet

OE4SRV - Projekat

Cvijović Martin 558/2017

Sadržaj

1 Projektni zadatak

Potrebno je realizovati sistem kojim se na svakih 1500ms startuje akvizicija sa kanala `A0` i `A1` pomoću taska `xTaskTimer` koji koristi funkcionalnost `vTaskDelayUntil`.

Potrebno je implementirati odloženu obradu prekida (eng. *deferred interrupt processing*) AD konvertora tako što se rezultat konverzije u prekidnoj rutini upisuje u red sa porukama (eng. *queue*) i obaveštava se task `xTask1` o prispeću nove poruke putem binarnog semafora (eng. *binary semaphore*). Poruka treba da sadrži informaciju o kanalu koji je očitan i gornjih 12 bita rezultata AD konverzije.

Task `xTask1` računa srednju vrednost zadnja četiri primljena odbirka za svaki kanal i upisuje izračunatu srednju vrednost u *mailbox*-ove (*queue* dužine 1) sa *overwrite*-om.

Preko UART-a je moguće zadati graničnu vrednost za signalizaciju. Iz *callback* rutine UART-a se podatak o željenoj graničnoj vrednosti smešta u red sa porukama `xQueueT2` i task `xTask2` treba da očita tu vrednost.

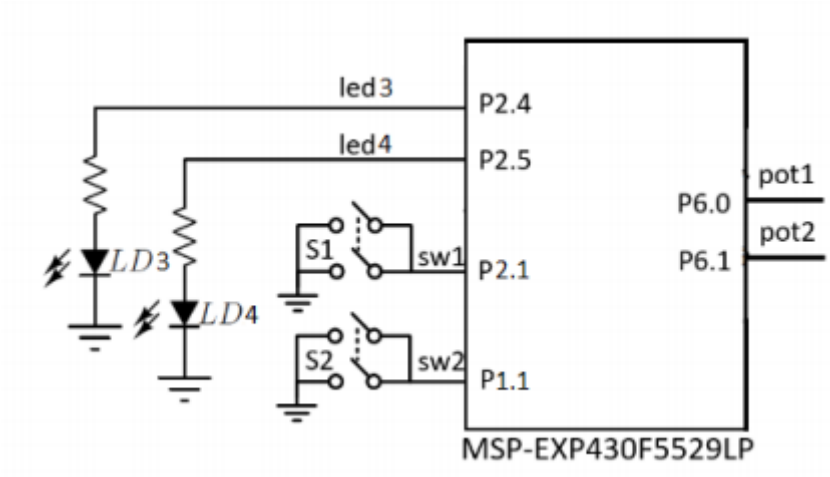
Ukoliko je očitana srednja vrednost određenog kanala veća od granične vrednosti, odgovarajuća dioda treba da se uključi.

Task `xTask3` ispituje stanje tastera `S1` i `S2` i šalje obaveštenje tasku `xTask2` preko reda `xQueueT2` ukoliko je neki od tastera pritisnut. Task `xTask2` na pritisak odgovarajućeg tastera inkrementira/dekrementira trenutnu graničnu vrednost i informaciju o trenutnom stanju dioda šalje preko UART-a.

2 Projektno rešenje

2.1 Arhitektura sistema

Na slici 1 prikazana je blok šema sistema. Sistem je baziran na `MSP430F5529` mikrokontroleru. Na portovima `P2.4` i `P2.5` nalaze se dve LED (eng. *light emitting diode*), `LD3` i `LD4`, respektivno. Na portovima `P2.1` i `P1.1` nalaze se tasteri `S1` i `S2` koji su u projektnom rešenju dodatno konfigurisani dodavanjem *pull-up* otpornika na odgovarajuće pinove. Dodatno, na portovima `P6.0` i `P6.1` vezana su dva potenciometra, `pot1` i `pot2`.



Slika 1: Blok šema sistema

Naponski signali sa oba potenciometra se dovode direktno na ulaze `A0` i `A1` AD konvertora. Taster `S1` se koristi za inkrementiranje granične vrednosti za uključivanje LED dioda dok se taster `S2` koristi za dekrementiranje iste granične vrednosti. Od dodatnog hardvera unutar `MSP430F5529` mikrokontrolera inicijalizovan je UART sa taktom od 1MHz, *baud rate* je postavljen na 9600bps, alocirano je 8 bita za podatak[1] sa jednim *stop* bitom i bez provere parnosti.

Inicijalna vrednost praga za uključivanje dioda (`ledThresholdValue`) postavljena je na polovinu opsega, odnosno na 2048. Pritiskom na dugme `S1` ili `S2`, ta vrednost se inkrementira/dekrementira za 50 dok je moguće direktno specificirati broj u opsegu $[0, 255]$ preko UART-a. Specificirani broj se preskalira u opseg koji pokriva AD konverzija, $[0, 4095]$.

2.2 Taskovi

2.2.1 Task 1

Task 1 inicijalno čeka binarni semafor `xFinishedADCConversionSemaphore`. Taj semafor je oslobođen iz taska *Timer* nakon uspešne AD konverzije. Semafor se oslobađa na svakih $1500\text{ms} + \Delta_{ADC_{A0}}$ i $1500\text{ms} + \Delta_{ADC_{A1}}$, odnosno na svakih 1500ms i kašnjenja AD konverzije za svaki od kanala. Pre oslobađanja semafora, u prekidnoj rutini AD konvertora se u *queue* `xQueueT1` upisuju podaci o izvršenoj AD konverziji. Taj podatak se sastoji iz informacije o kanalu koji je izvršio konverziju i dobijenoj vrednosti.

Kada se semafor oslobodi, za odgovarajući kanal se dobijena vrednost upisuje u odgovarajući niz `conversionValues[channel][i]`, $i \in [0, 3]$.

Na kraju se računa nova srednja vrednost za odgovarajući kanal i upisuje se u odgovarajući *mailbox* (`xQueueMailboxChannel0` ili `xQueueMailboxChannel1`).

2.2.2 Task 2 i Task 3

Task 2 čita iz tri različita *queue*-a i ne blokira se na njima[2], to su redovi `xQueueMailboxChannel0`, `xQueueMailboxChannel1` i `xQueueT2`. *Mailbox queue*-ovi se popunjavaju od strane taska 1 i objašnjeni su u prethodnom tekstu. Ukoliko task uspešno učita novi podatak iz nekog *mailbox*-a, zadužen je da proveri i, po potrebi, ažurira stanje dioda. `xQueueT2` predstavlja red sa porukama dobijenim od strane taska 3. Taj red sadrži informaciju o novoj vrednosti praga uključivanja dioda.

Za parsiranje i debaunsiranje (eng. *debouncing*) tastera zadužen je task 3. Ukoliko je pritisnut taster `S1` ili `S2`, u tasku 3 se generiše podatak tipa `userInteractionData`, koji je kompleksni podatak koji se sadrži iz dva polja:

- `enum THRESHOLD_UPDATE_TYPE { INCREMENT, DECREMENT, SPECIFY_DIRECT };`
- `uint16_t directSpecifiedValue;`

[1] 8 bita za podatak koji se može poslati/primiti preko UART-a podrazumeva da je opseg neoznačenog celog broja koji se može poslati/primiti jednak $[0, 255]$. Gornjih 12 bita rezultata AD konverzije (odnosno vrednost praga za uključivanje dioda) predstavljaju neoznačeni ceo broj u opsegu $[0, 4095]$ što znači da je nemoguće pokriti ceo opseg vrednosti pomoću UART-a. Rešenje ovog problema je izvedeno skaliranjem podatka primljenog preko UART-a tako što je primljeni podatak pomnožen sa 16.

[2] Ova implementacija čini da je task 2 *free running* task jer se nikada ne blokira, međutim, taj problem je rešen pomoću kontrole prioriteta taskova i rešenje je objašnjeno u daljem tekstu.

U slučaju pritiska na neki od tastera vrednost `directSpecifiedValue` se ne inicijalizuje već se samo bira tip ažuriranja praga, `INCREMENT` ako je pritisnut `S1`, odnosno `DECREMENT` ako je pritisnut `S2`. Taj podatak se potom smešta u `xQueueT2` koji se obrađuje u Tasku 2. Još jedan način generisanja poruke koja se smešta u `xQueueT2` je iz prekidne rutine UART-a. U ovom slučaju `directSpecifiedValue` je preskalirana vrednost primljena preko UART-a dok se tip ažuriranja praga postavlja na `SPECIFY_DIRECT`.

Na osnovu podatka `userInteractionData` task 2 je zadužen da ažurira vrednost `ledThresholdValue`. Ukoliko je pritisnut taster `S1` ili `S2`, task 2 je takođe zadužen da obavesti računar o stanju dioda preko UART-a.

2.2.3 Task *Timer*

Task *Timer* je task koji je zadužen da startuje AD konverziju na kanalima `A0` i `A1` i nakon toga pređe u blokirano stanje narednih 1500ms.

2.3 Prioriteti taskova i dijagram toka programa

Prioriteti taskova definisani su sledećom tabelom:

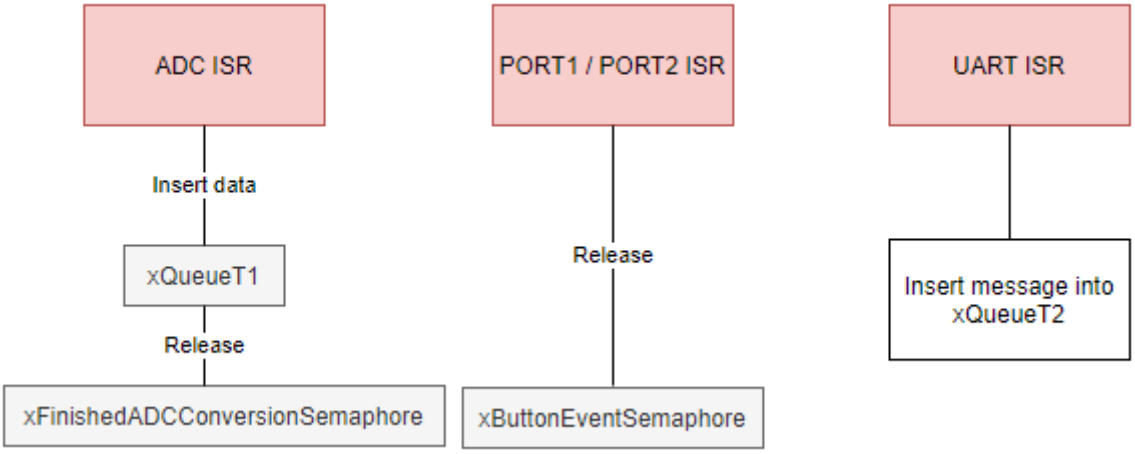
Task	Prioritet
`mainTASK_1_PRIORITY`	2
`mainTASK_2_PRIORITY`	1
`mainTASK_3_PRIORITY`	3
`mainTASK_TIMER_PRIORITY`	1

Najviši prioritet ima task 3. Task 3 se blokira odmah pri uključenju programa jer čeka na semafor `xButtonEventSemaphore`. Prilikom pritiska na taster `S1` ili `S2`, prekidna rutina odgovarajućeg porta će osloboditi semafor i task 3 će debaunsirati taster, smestiti podatak u `xQueueT2` i blokirati ponovo na istom semaforu.

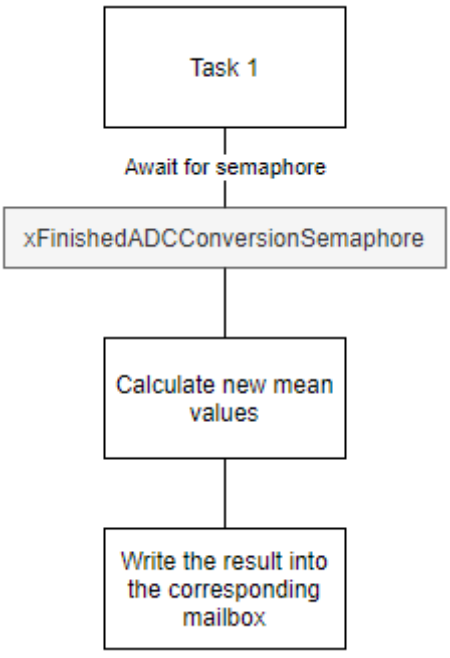
Sledeći task po prioritetu je task 1 koji se odblokira na svakih 1500ms, izračunava srednje vrednosti i pakuje ih u *mailbox*-ove.

Poslednja dva taska, task *Timer* i task 2 imaju isti prioritet i njih konkuretno izvršava *scheduler* RTOS-a (eng. *real-time operating system*). Task *Timer* je, slično kao i task 1, odblokiran na svakih 1500ms što ostavlja ostatak procesorskog vremena za izvršavanje taska 2.

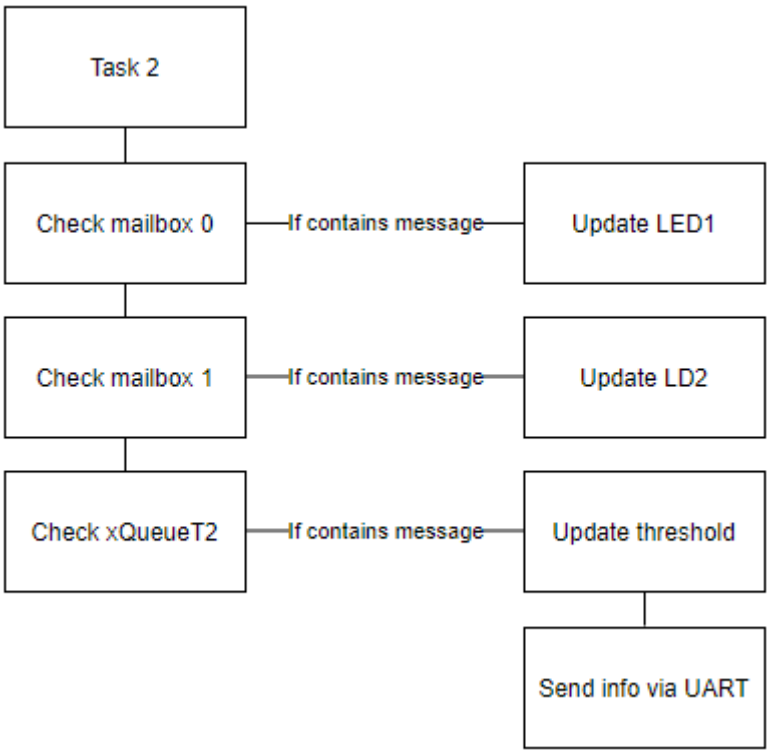
Detaljni dijagrami toka taskova i prekidnih rutina dati su na slikama ispod.



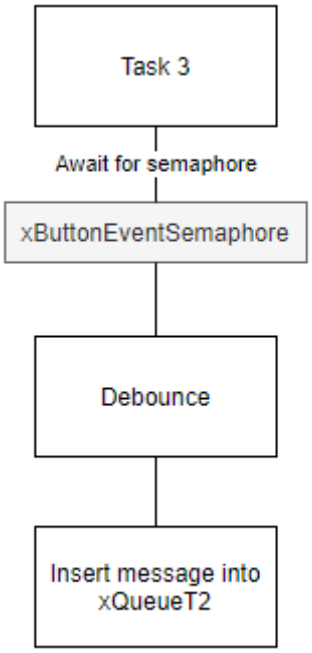
Slika 2: Dijagram toka prekidnih rutina



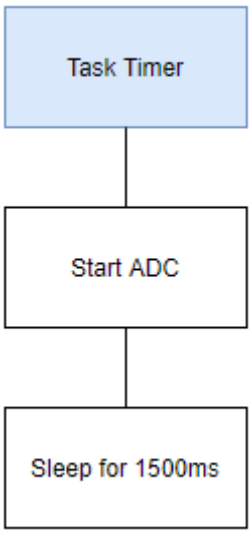
Slika 3: Dijagram toka taska 1



Slika 4: Dijagram toka taska 2



Slika 5: Dijagram toka taska 3



Slika 6: Dijagram toka taska *Timer*