Univerzitet u Beogradu
Elektrotehnički fakultet
Katedra za elektroniku

Predmet: 32bitni mikrokontroleri

Projekat: Programator fleš memorije

Student:
Vlastimir Đokić 3226/2013

Profesor:
Dr Dragan Vasiljević

Beograd, Februar 2013.

Sadrzaj:

1 Uvod

Ovaj projekat je deo ispita „32-bitni mikrokontroleri“ master studija Katedre za elektroniku Elektrotehničkog fakulteta u Beogradu. Cilj kursa je upoznavanje sa ARM-ovom familijom procesora *Cortex*, konkretnto serijom M. Projekat je realizovan na razvojnoj ploči STM32VLDiscovery koju pokreće ST-ov mikrokontroler STM32F100RBT6B. Tema projekta je demostracija aplikacije koja omogućava *in-application programming* (IAP). Izlaganje je organizovano u jasno definisanim celinama:

Prva celina predstavlja osvrt na korišćenu tehnologiju.
Druga celina iznosi problem programiranja fleš memorije.
Treća celina opisuje teorijska znanja na kojim se bazira rešenje.
U četvrtoj je kratak komentar sors kodova.

2 Korišćene tehnologije

2.1 Zašto ARM?

Na ovogodišnjem CES-u (*Consumer Electronics Show*) predstavnici ARM-a su objavili informaciju da 95% tablet računara i pametnih telefona pokreću ARM čipovi. Inače 2013. godinu su najavili kao godinu kad će se prodati više tablet računara (koje pokreću ARM čipovi) nego laptop računara (koje pokreće x86 tehnologija). Takođe je fascinantan podatak da se svake sekunde proda 125 uređaja koje pokreću ARM čipovi, to je skoro 4 milijarde uređaja godišnje a ako se ima u vidu da pojedini uređaji imaju više od jednog ARM čipa, pravi broj prodatih ARM čipova izlazi na neverovatnih 5 milijardi godišnje. 80% digitalnih kamera prodatih širom sveta ima ARM-ovu tehnologiju u sebi. Svi gore navedeni podaci predstavljaju jasan motiv i podstrek za izučavanje arhitekture ARM-ovih procesora[1].

2.2 Šta je Cortex?

Ranije ARM-ovi procesori, zasnovani na arhitekruri ARM7 i ARM9, su podrazumevali samo dizajn centralne procesorske jedinice. Time se podrazumeva da su inzenjeri proizvođača mikrokontrolera sami dizajnirali sistem za servisiranje prekida, mapiranje memorije, sistem za debagovanje… Kod *Cortex* familije procesora ARM-ovi inzenjeri su u čip procesora pored centralne procesorske jedinice stavili i *Nested Vector Interrupt Controller*, kontroler zadužen za servisiranje prekida, *SysTick* tajmer, 24-bitni tajmer koji je namenjen za kernel RTOS-a (*Real-Time Operating System*), *CoreSight*, sistem za debagovanje kao i memorijsku mapu.

Cortex familija ima tri serije:

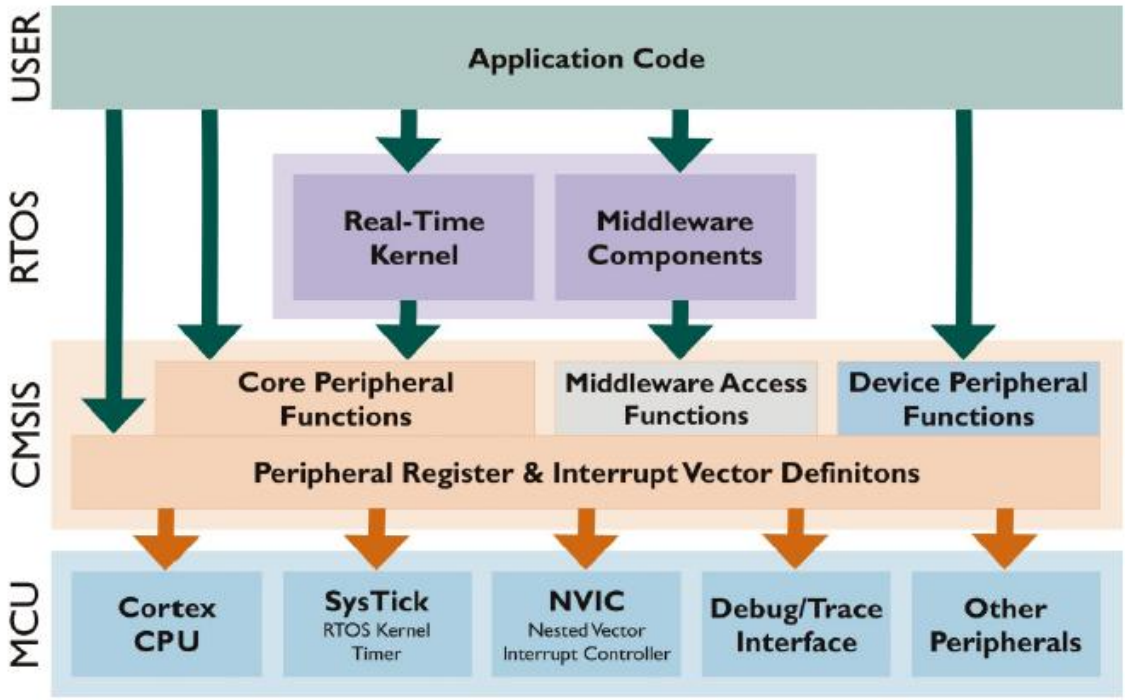
- Serija A je namenjena apllikacijama koje zahtevaju veliku računarsku snagu tj. pokreću kompleksne operacione sisteme i zahtevne interaktivne grafičke aplikacije napr. tablet računari i pametni telefoni.
- R serija je namenjena *real-time* aplikacijama. Kako bi se izbegli gubitak podataka ili mehanička šteta poštovanje strogih vremenskih ograničenja je presudno. Ova serija *Cortex* procesora je dizajnirana u tu svrhu.
- Serija M je energetski efikasna serija procesora, pogodna za razvoj savremenih embedid sistema. Konkretno ova serija obuhvata veliko bogatstvo mogućnosti po povoljnoj ceni, veliku mogućnost povezanosti sa drugim sistemima, veću uštedu energije kao i mogućnost lakšeg razvoja koda (uz CMSIS).

Procesor koji pokreće mikrokontroler STM32F100RBT6B je *Cortex*-M3. *Cortex*-M3 je zasnovan na Harvard arhitekturi što podrazumeva različite magistrale za kod i podatke. Ovakav pristup memoriji omogućuje veći paralelizam u radu što povećava efikasnost procesora. 4GB memorijskog prostora je podeljeno u jasno definisane delove za kod, SRAM, periferije i sistemske periferije. Specifičnost *Cortex* familije su i dve "oblasti" fleš memorije od po 1MB koje omogućavaju upis i čitanje u pojedine registre i memorijske lokacije u jednom atomskom operacijom. Dok su ARM7 I ARM9 imali dva različita instrukcijsa seta (ARM 32-bit set i *Thumb* 32-bit set) *Cortex* familija podržava ARM *Thumb*-2 instrukcijski set. Ovaj set instrukcija kombinuje 16-bitne i 32-bitne instrukcije kako bi iskoristio performanse ARM 32-bitnog seta insktrukcija sa gustinom koda *Thumb* 16-tnog seta instrukcija. *Thumb*-2 set instrukcija je bogat set instrukcija dizajniran za C/C++ kompajlere. Ovo znači da *Cortex* aplikacije cele mogu biti napisane u C-u.

Jedna od glavnih karakteristika *Cortex*-M3 serije je NVIC (*Nested Vector Interrrupt Controller*). NVIC obezbeđuje sistem za servisiranje izuzetaka i prekida za sve *Cortex*-M3 bazirane mikorkontrolere. NVIC obezbeđuje vektore prekida za do 240 izvora prekida gde svaki izvor može dobiti sopstveni prioritet. NVIC je dizajniran za ekstremno brzo obrađivanje prekida. Vreme potrebno za primanje zahteva za prekid do dohvatanje prve linije koda prekidane rutine je samo 12 ciklusa. U slučaju *back-to-back* prekida NVIC koristi metod *tail chaining* koji omogućuje da se uzastopni prekidi servisiraju sa samo 6 ciklusa zakašnjenja. Za veme odlaganja registara na stek prekid većeg prioriteta (u odnosu na prekid koji čeka na izvršavanje), koji se pojavio za vreme odlaganja registara, može izvršiti svoju prekidnu rutinu bez čekanja. Servis prekida je usko povezan sa *low-power* režimima *Cortex*-M3. Moguće je podesiti CPU da automatski ulazi u *low-power* mod nakon izlaska iz prekida. Jezgro zatim ostaje u stanju neaktivnosti dok se ne pojavi novi izuzetak[2].

2.3 Cortex Microcontroller Software Interface Standard (CMSIS)

CMSIS predstavlja skup bibliotaka koje olakšavaju razvoj softvera za mikrokontrolere bazirane na procesorima Cortex-M0, Cortex-M1 i Cortex-M3. Namera je i da naredne generacije procesora budu kompatibilne sa ovom bibliotekom. Struktura CMSIS-a je predstavljena na slici 1.[3].



Slika 1. CMSIS

3 "Spuštanje" koda na fleš memoriju

Bitna karakteristika embedid sistema je mogućnost izmene firmvera u finalnom proizvodu. Ova karakteristika je poznata kao IAP (*in-application programming*). Fleš memorija mikrokontrolera STM32F100RB može biti programiran koristeći ICP (*in-circuit programming*) metod ili IAP (*in-application programming*). ICP je metod koji se koristi za programiranje fleš memorije pomoću JTAG (*Joint Test Action Group*), SWD (*Serial Wire Debug*) protokol ili *bootloader* kako bi se pokrenula korisnička aplikacija (*firmware*). ICP omogućava brzo i jednostavno instaliranje i debagovanje korisnicke aplikacije. Na drugoj strani IAP koristi komunikacioni interfejs mikrokontrolera (I/O, UART, I2C, SPI, itd.) kako bi "spustio" podatke na fleš memoriju mikrokontrolera. IAP omogućava reprogramiranje mikrokontrolera dok je na njemu već aktivna postojeća aplikacija. Bitno je napomenuti da se sam IAP drajver mora programirati ICP metodom. [4]

3.1 Kako C-Spy programira fleš?

Za programiranje mikrokontroler STM32F100RBT6B su dostupne dve varijante *bootloader*-a. Jednu koristi C-Spy-a a drugi je ugrađen u sistemsku memoriju. Kada kod (program) treba "spustiti" (*download*-ovati) i debagovati fleš memorija ne moze biti programirana direktno od strane C-Spy-a. Programiranje fleš memorije mora da izvrši specijalni program – *flashloader*, koji se izvršava na sistemu koji se programira (*target system*). Mehanizam *flashloader*-a je pogodan za upis u bilo koju vrstu memorije koja ne moze dirktno biti "popunjena" od strane debagera. Ukratko mehanizam je sledeći [5]:

- Debager spušta *flashloader* u RAM.
- Kod koji se želi upisati u fleš se spušta pored *flashloade*-a u RAM
- Zatim *flashloader* prebacuje kod u fleš memoriju (naravno ako je potrebno drugi i treći korak se ponavljaju dok se ne unese cela aplikacija)
- Nakon što je aplikacija dostupna u fleš memoriji debager oslobađa RAM čime se proces programiranja fleš memorije završava

3.2 Ugrađeni *bootloader*

Mikrokontroler STM32F100RBT6B se može programirati i pomoću *bootloader*-a koji se nalazi u sistemskoj memoriji mikrokontrolera. Da bi se *bootloader* mogao koristiti, potrebno je postaviti „Boot" pinove mikrokontrolera na odgovarajuće nivoe tako da po resetu vrednosti na njima budu stabilne. Pin BOOT1 (pin 28) je potrebno postaviti na logičku 0, a pin BOOT0 (pin 60) na logičku 1. Pored toga, potrebno je povezati USART 1 TX (pin A10 – pin 42) i RX (pin A09 – pin 43) pinove mikrokontrolera preko translatora nivoa do RS232 interfejsa koji će slati komande i podatke *bootloader*-u. U ovom projektu, za translator nivoa korišćen je Maxim MAX3232 za translaciju nivoa sa ±12V (na strani PC-a) na ±3.3V (na strani mikrokontrolera) i obrnuto.[6] Detalji o translatoru nivoa mogu se naći u datasheetu[7].

3.3 Princip IAP drajvera

IAP drajver se prvo mora "spustiti" na fleš ili preko JTAG/SWD interfejsa ili ugradjenim *bootloader*-om. IAP drajver koristi UART kako bi:

- "spustio" binarni fajl preko HyperTerminala na STM32F100RBT6B internu fleš memoriju
- "podigao" binarni fajl sa interne fleš memorije
- pokrenuo izvršavanje korisničke aplikacije

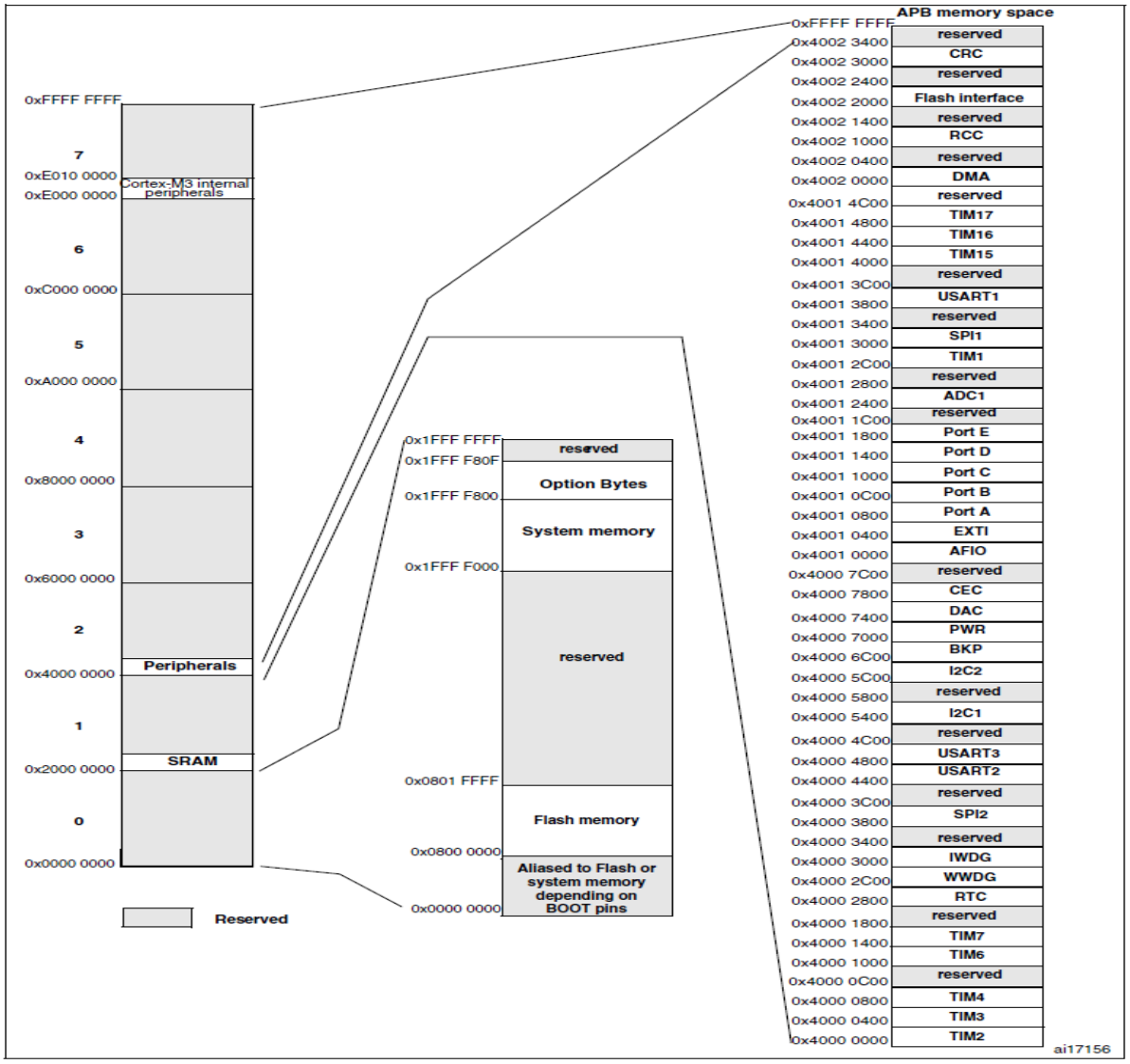
4 Terijska osnova

4.1 FPEC

Organizacija fleš memorije mikrokontrolera STM32F100RB (prikazana je na slici 2.):

- 128KB glavne memorije (izdeljene na stranice veličine 1KB)
- Informacioni blok (koji se sastoji od sistemske memorije i opcionih bajtova)
- Registri koji su zaduzeni za interfejs fleš memorije

Fleš memorija je organizovana kao niz 32-bitnih ćelija koje mogu biti iskorišćene za unos koda (programa) i podataka.



Slika 2. Memorijska mapa

Upis u glavnu memoriju i opcione bajtove obavlja FPEC (*Flesh Program/Erase Controller*). Napon potreban za operacije upisa i brisanje se interno greneriše. Glavna fleš memorija može biti zaštićena od neovlašćenog pristupa (čitanje/upis/brisanje).

FPEC je kontroler koji reguliše operacije upisa (progrmiranja i brisanja) i čitanja fleš memorije. Sastoji se od sedam registara (tačne adrese date su na slici 3.):

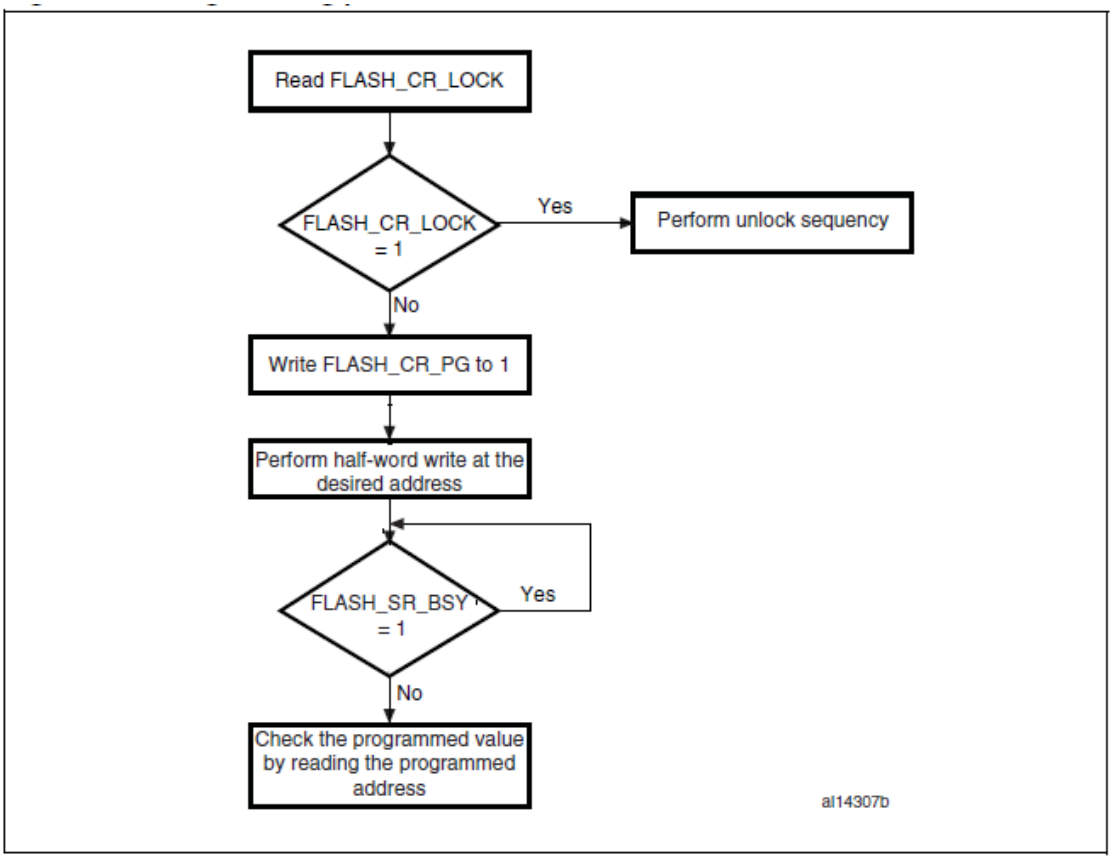
- FLESH_KEYR
- FLESH_OPTKEYR
- FLESH_CR
- FLESH_SR
- FLESH_AR
- FLESH_OBR
- FLESH_WRPR

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Slika 3. Tačne adrese fleš memorije

Posle reseta FPEC blok je zaštićen. FLASH_CR registar nije dostupan za upisivanje. Sekvenca za otključavanje mora biti upisana u FLASH_KEYR da bi se otključao FPEC blok. Sekvenca se sastoji od dva ciklusa upisivanja. Upisuju se vrednosti KEY1 (0x45670123) pa zatim KEY2 (0xCDEF89AB) u FLASH_KEYR registar. Bilo koja pogrešna sekvenca upisa blokira fleš memoriju i FLASH_CR registar do sledećeg reseta. FPEC blok i FLASH_CR registar takođe mogu biti blokirani od strane korisnikovog softvera upisom 1 u LOCK bit FLASH_CR registra. Metod otključavanja je isti kao i posle reseta (upis sekvence za otključavanje).

Glavna fleš memorija može biti programirana 16 bita odjednom. Operacija programiranja je započeta kad CPU setuje PG bit FLASH_CR registra i upiše prvi 16-bitni podatak. Bilo kakav pokušaj upisa podatka koji nije veličine 16 bita rezultovaće greškom na magistrali generisanom od strane FPEC-a. Ako se pokuša operacija upisa/čitanja fleš memorije tokom programiranja (bit BSY registra FLASH_SR je 1) CPU se zaustavlja dok se ne završi tekuća operacija programiranja.



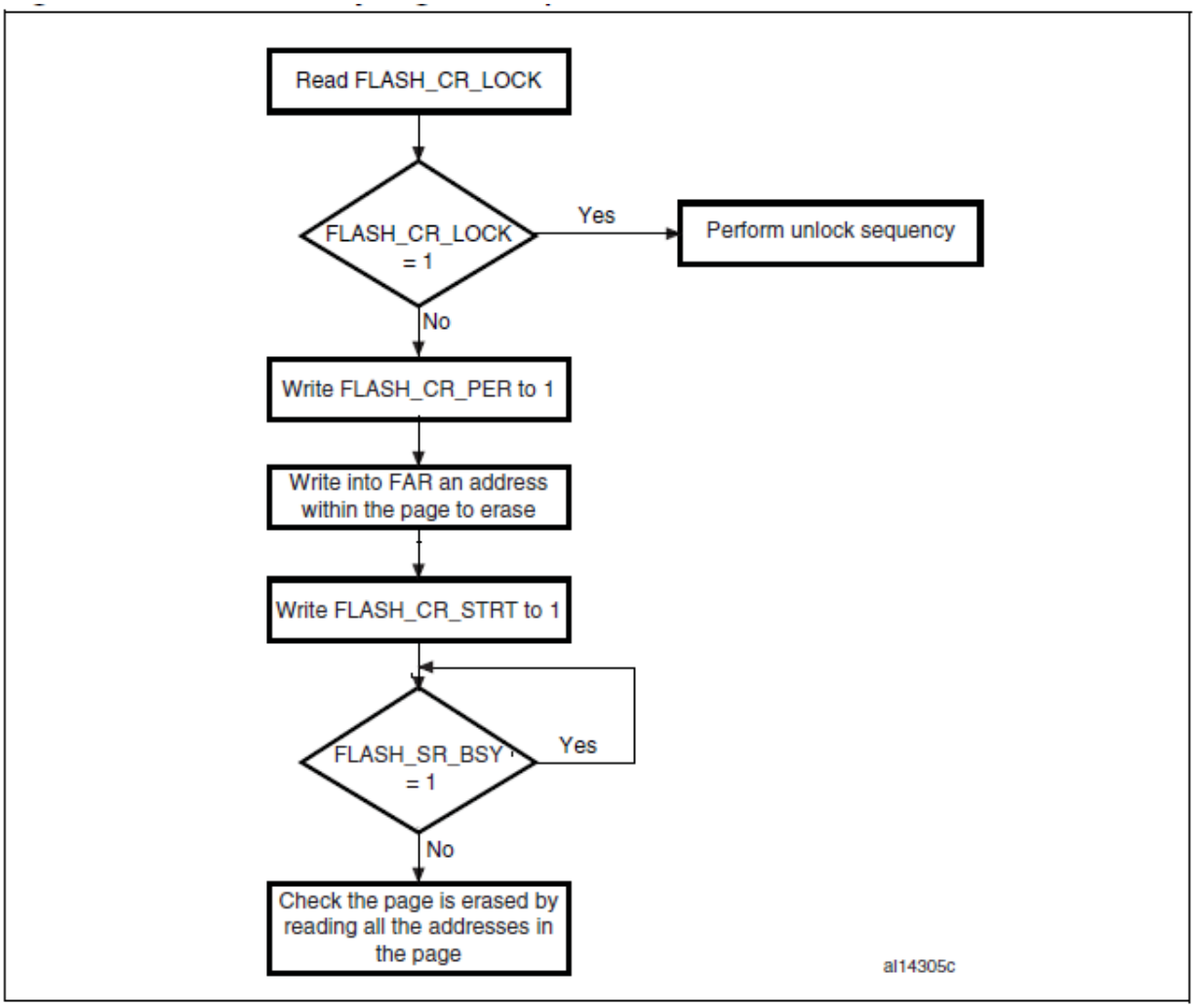
Slika 4. Operacija upisa 16-bitnog podatka u fleš memoriju

Programiranje 16-bitnog podatka u glavnu fleš memoriju (sliuka 4.):

- Proverava se da li je trenutno na snazi operacija sa fleš memorijom proveravanjem bita BSY u registru FLASH_SR
- Setuje se bit PG u registru FLASH_CR

- Izvrši se upis 16-bitnog podatka na potrebnu adresu
- Sačeka se da se bit BSY resetuje
- Pročita se programirana vrednost i proveri

FPEC pre upisa čita sadržaj sa lokacije na kojoj se vrši upis i proverava da li je sadržaj lokacije obrisan. Ako sadržaj ciljne lokacije nije obrisan operacija upisa se preskače i oglašava se upozorenje setovanjem bita PGERR u registru FLASH_SR (jedini izuzetak je upis vrednosti 0x00000000). Inače svaka ćelija fleš memorije je nominalno popunjena sa 0xFFFFFFFF. Ako je adresirana lokacija zaštićena od upisa sadržajem registra FLASH_WRPR operacija programiranja se preskače i oglašava se upozorenje setovanjem bita WRPRTERR u registru FLASH_SR. Kraj operacije programiranja je indikovan setovanjem EOP bita registra FLASH_SR. Registri fleš memorije nisu dostupni za upisivanje kad je bit BSY registra FLASH_SR setovan.



Slika 5. Operacija brisanja jedne stranice fleš memorije

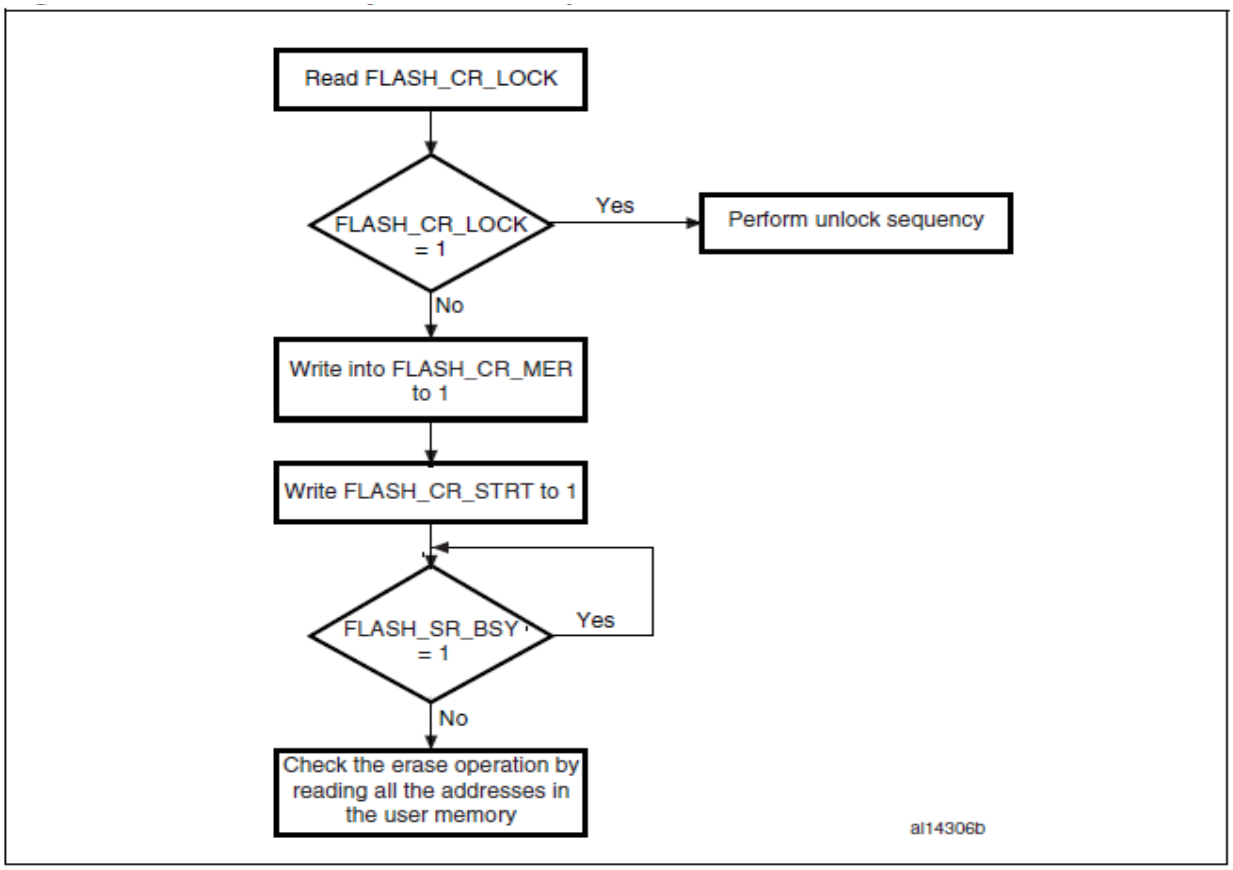
Brisanje fleš memorije je moguće stranicu po stranicu ili može biti cela izbrisana.

Brisanje stranice (prikazano na slici 5.) može biti izvršeno biranjem opcije FPEC-a - brisanje stranice. Procedura je sledeća:

- Proverava se da li je trenutno na snazi operacija sa fleš memorijom proveravanjem bita BSY u registru FLASH_SR
- Setuje se PER bit registra FLASH_CR
- Programira se FLASH_AR registar brojem stranice koja se želi izbrisati
- Setuje se STRT bit registra FLASH_CR (ovaj bit započinje operaciju brisanja)
- Sačeka se da se bit BSY resetuje
- Pročita se programirana strana i proverava

Za brisanje celokupne glavne fleš memorije (prikazano na slici 6.) se koristi komanda-*Mass erase*. Procedura je sledeća:

- Proverava se da li je trenutno na snazi operacija sa fleš memorijom proveravanjem bita BSY u registru FLASH_SR
- Setuje se MER bit registra FLASH_CR
- Setuje se STRT bit registra FLASH_CR (ovaj bit započinje operaciju brisanja)
- Sačeka se da se bit BSY resetuje
- Pročita se programirana strana i proverava



Slika 6. Operacija brisanja cele glavne fleš memorije

4.2 Komunikacija sa računarom

Kako bi se obezbedila komunikacija sa računarom potrebno je mikrokontroler povezati na način kao kada se koristi ugradjen *bootloader*. Podaci se primaju i salju sa računara preko *HyperTerminal*-a. Komunikacija preko USART periferije mikrokontrolera sa spoljnim svet je krajnje jednostavna. U jednom trenutku je moguće poslati ili primiti 1B. Podatak se smešta/čita u donjih 8 bita USART_DR registra. Kada se želi proveriti da li je podatak stigao čita se bit RXNE registra USART_SR, ako je setovan podatak se nalazi u registru USART_DR . Kada se podatak šalje, posle smeštanja podatka u USART_DR, proverava se bit TXE, ako je bit TXE setovan moguće je poslati sledeći bajt. Protokol za slanje fajlova je YMODEM. Detaljnije o YMODEM može se naći u [8] i [9].

4.3 Konfigurisanje projekta koji želimo da "spustimo" fleš

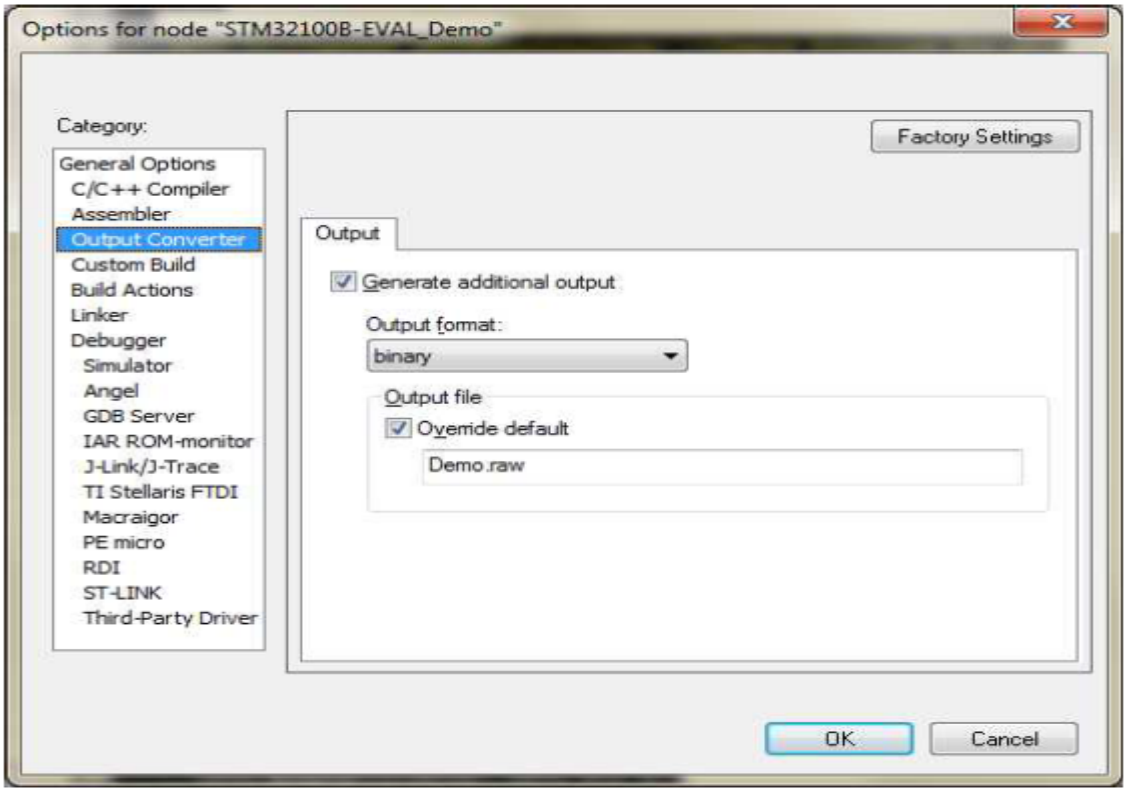
Da bi se rešio problem sa tabelom vektora prekida radi se sledeće:

- U projektu fajla koji treba kasnije da se unese podesi se da format izlaznog fajla bude binarani tj. da nema nikakve podatke vezane za linkovanje ili debagovanje. Smatra se da se unosi ispravna aplikacija.
- U .icf fajlu podesiti da fleš memorija počinje od zeljene lokacije i podesiti da je vektorska tabela prekida postavljena na istu.

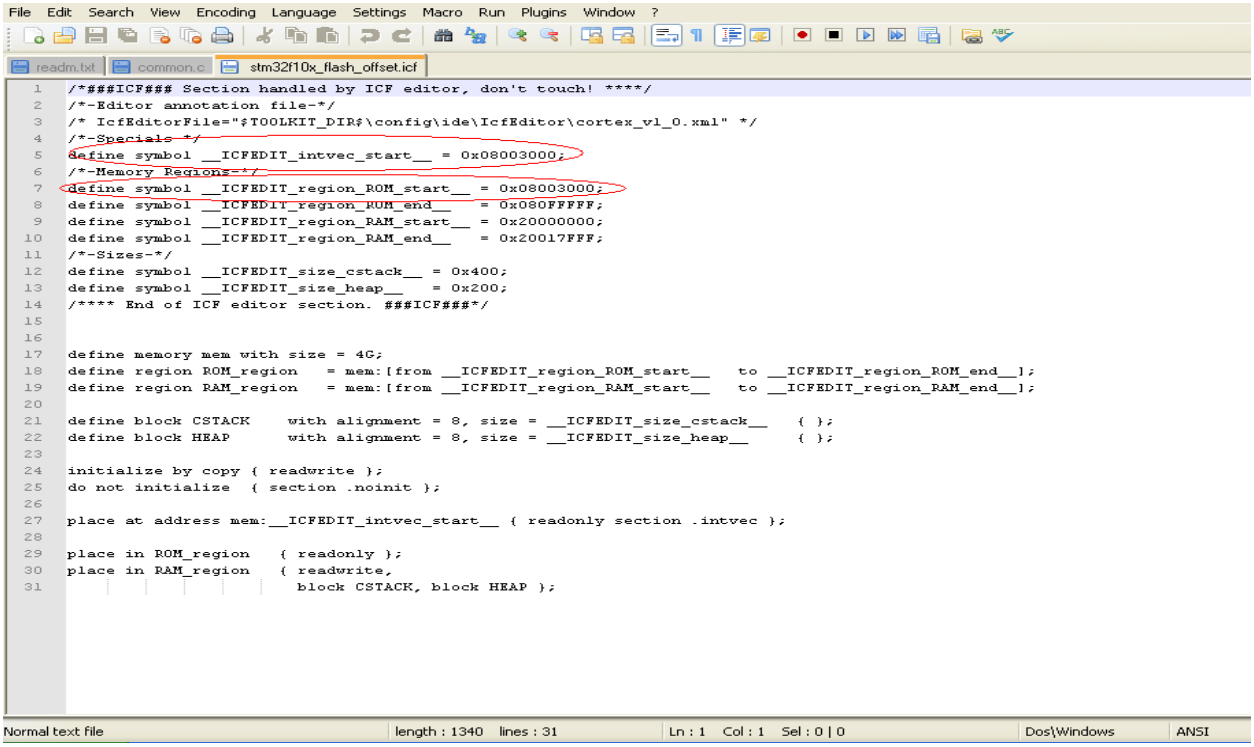
Za dobijanje binarnog fajla, potrebno je podesiti odgovarajuća dodatna podešavanja u „*OutputConverter*" sekciji EWARM projekta. Slika 7. prikazuje primer tog podešavanja.

Ono što je neophodno podesiti jeste da "*Generate additional output*" bude štiklirano, kao i da "Output format" bude podešen na "*binary*". Reimenovanje nije obavezno. Nakon uspešnog linkovanja projekta, dobija se binarni fajl u istom direktorijumu gde se nalazi iizvršni .out fajl. To je obično "*Exe*" pod-direktorijum.

U folderu gde se nalazi fajl *workspace*-a se obično nalazi i .icf koji treba prepraviti kao na slici 8. Veličina samog IAP drajvera diktira adresu od koje će se početi sa unosom korisničke aplikacije. U ovom slučaju to je adresa 0x08003000.



Slika 7. Podesavanja za dobijanje izlaznog fajla u .bin formatu



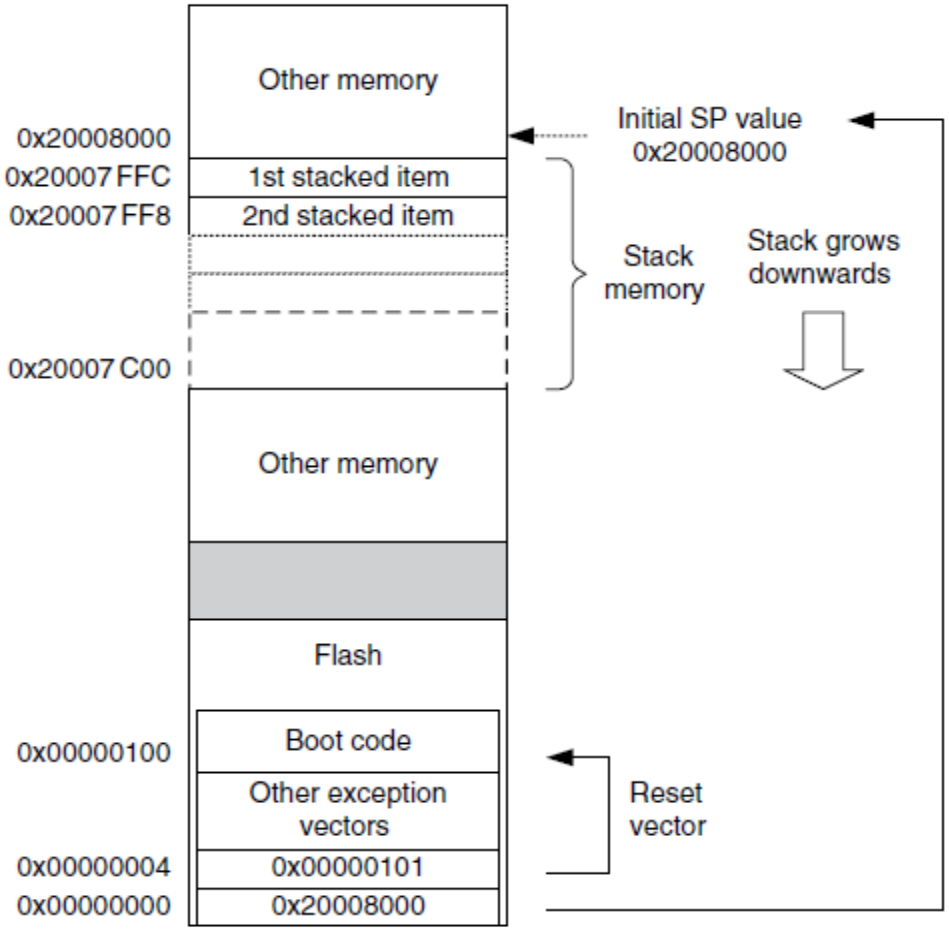
Slika 8. Izmene u .icf fajlu aplikacije koju treba uneti (kako bi uneta aplikacija “imala utisak” da je programirana na početku glavne fleš memorije)

4.4 Reset

Posle reseta čitaju se dve reči iz memorije:

- Sa ardrese 0x00000000: početna vrednost R13 (SP). Adresa 0x00000000 može biti podesavana u .icf fajlu promenom adrese sa kojom počinje vektorska tabela prekida.
- Sa sledeće adrese (nominalno je to 0x00000004): reset vektor (početna adresa izvršavanja programa).

Vektorska tabela prekida počinje posle inicijalne vrednosti SP (*stack pointer*) regista. Prvi vektor je reset vektor. Kod *Cortex*-M3 procesora vekorske adrese u vektorskoj tabeli prekida treba da imaju LSB 1 kako bi pokazali da se radi o *Thumb* kodu. Pošto je reset vektor dohvaćen, *Cortex*-M3 može da počne izvršavanje programa od adrese reset vektora. Neophodno je imati SP inicijalizovan zato što se mogu aktivirati neki izuzetci (napr. NMI) neposredno posle reseta tako da stek mora biti definisan za potrebe odlaganja registara pri servisirnju izuzetka[10].



Slika 9. Prikaz čitanja memorije nakon reseta

5 Literatura

[1] http://www.itproportal.com/2013/01/18/ces-2013-arms-state-nation/
[2]The Insider's Guide To The STM32 ARM Based Microcontroller, Trevor Martin
[3]Getting started wuth CMSIS, Doulos
[4] http://www.st.com/internet/mcu/product/216844.jsp, ST-ov sajt za dokumentaciju vezanu za mikrokntroler STM32F100RBT6B
[5] http://www.iar.com/en/Service-Center/Resources/
[6]Bootloader, Ivan Lukić
[7] http://www.mikroe.com/downloads/get/1509/max3232_manual_v100.pdf
[8] File Transfer by XModem Protocol Using UART Module
[9] XMODEM/YMODEM PROTOCOL REFERENCE, Chuck Forserbeg
[10]The definitive guide to the ARM Cortex-M3, Second Edition, Joseph Yiu