Kako (možda) programirati STM32: HAL, CubeMX i make

Lazar Premović - za Beoaviu, sektor za elektroniku i upravljanje 21-02-2022

Rezime

Tema izveštaja je pregled alata koji olakšavaju programiranje STM32 mikrokontrolera i to: HAL biblioteke, CubeMX generatora koda i make build sistema. Ovi alati su demonstrirani na primeru eksternog led indikatora koji se pali i gasi pomoću tajmera.

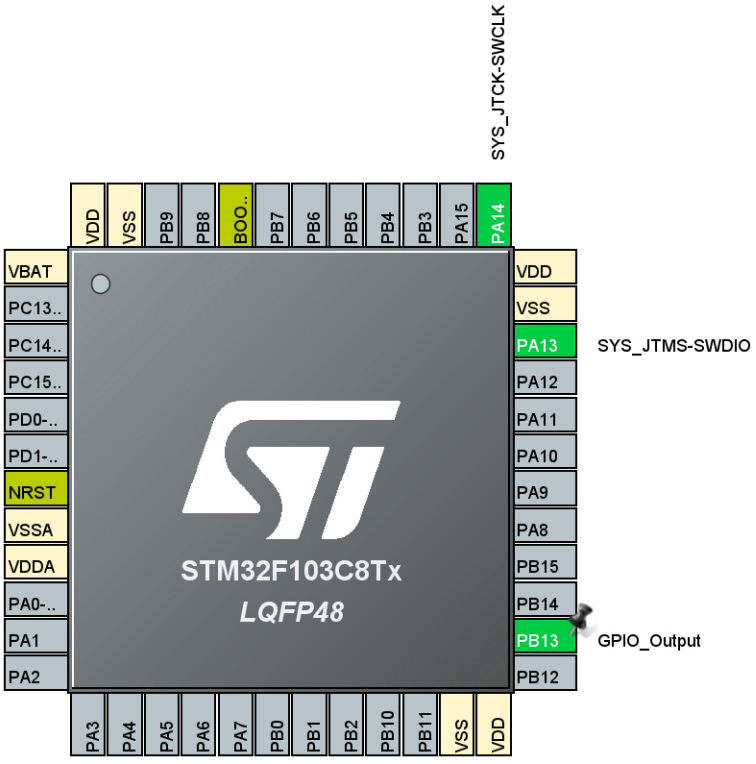
CubeMX

CubeMX je softver koji kroz grafički korisnički interfejs omogućava kreiranje skeleta projekta i konfigurisanje osnovnih parametara mikrokontrolera. Glavne komponente CubeMX softvera o kojima će ovde biti reči su:

- Konfiguracija pinova
- Konfiguracija periferija
- Konfiguracija signala takta
- Konfiguracija projekta

Konfiguracija pinova

Interfejs za konfiguraciju pinova zauzima najveći deo taba za konfiguraciju koji je podrazumevano prikazan. Na njemu se vidi izabrani mikrokontroler sa svim pinovima i ukoliko postoje, funkcijama koje ti pinovi vrše. Pin se može konfigurisati prostim klikom na njega i izborom željene funkcije za taj pin.

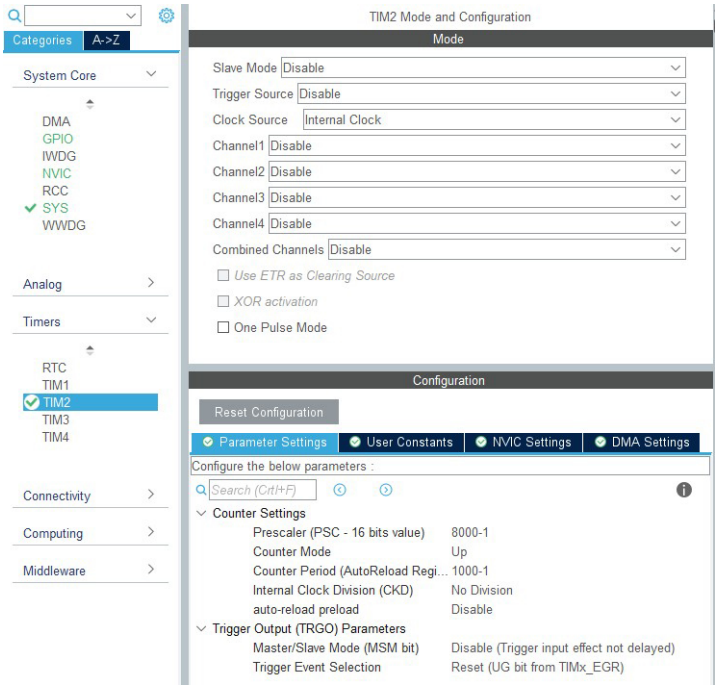


Slika 1: Interfejs za konfiguraciju pinova

Na slici se može videti `STM32F103C8Tx` mikrokontroler kome je pin B13 konfigurisan kao GPIO izlaz a pinovi A13 i A14 su konfigurisani kao linije podataka i signala takta za potrebe debagera i programatora.

Konfiguracija periferija

Interfejs za konfiguraciju periferija se prikazuje odabirom periferije sa spiska koji se nalazi na levoj ivici prozora. On omogućava podesimo podešavanje periferija prostim izborom opcija ili popunjavanjem numeričkih vrednosti.

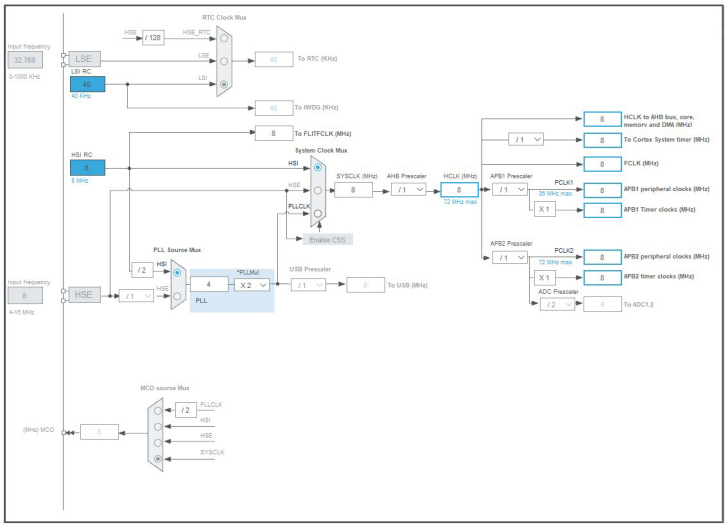


Slika 2: Interfejs za konfiguraciju periferija

Ovde se može videti konfiguracija tajmera 2, konfigurisanog tako da generiše prekid svake sekunde.

Konfiguracija signala takta

Mikrokontroleri imaju jako kompleksne sisteme za generisanje signala takta kako bi imali što manju potrošnju struje tako da ovde nećemo ulaziti u detalje tog sistema, već ćemo samo pokazati kako izgleda interfejs u slučaju da se kasnije pojavi potreba za izmenom konfiguracije signala takta.



Slika 3: Interfejs za konfiguraciju signala takta

Konfiguracija projekta

Tab za konfiguraciju projekta sadrži sve bitne parametre samog projekta. Neki od korisnih parametara su:

- Odeljak `Project -> Project Settings`
  - Ime projekta
  - Direktorijum u kome će se projekat nalaziti (CubeMX će kreirati direktorijum sa imenom projekta unutar direktorijuma specifikovanog ovde)
  - Struktura aplikacije, ovo polje neznatno utiče na strukturu projekta i preporučuje se izbor osnovne strukture projekta (polje je nemoguće promeniti nakon inicijalnog generisanja koda)
  - Izbor lanca alata koji će biti korišćen (u ovom slučaj ćemo koristiti Makefile)
- Odeljak `Project -> STM32Cube MCU packages and embedded software packs`
  - Ovde se može izabrati koji delovi HAL biblioteke će biti iskopirani u projekat, moguće je izabrati izmelju: cele biblioteke, samo neophodnih delova i uvezivanja biblioteke umesto njenog kopiranja. Ovde se savetuje izbor opcije koja kopira samo neophodne delove kako bi se smanjila veličina projekta a da bi i dalje bilo moguće prenošenje projekta kao kompletne celine.

Struktura projekta

Nakon konfigurisanja svih potrebnih parametara skelet projekta se generiše klikom na dugme `GENERATE CODE`. Struktura dobijenog projekta na prvi pogled može delovati komplikovano tako da će ovde biti detaljnije objašnjena.

Datoteke *.ioc i .mxproject sadrže konfiguraciju mikrokontrolera i samog projekta, njih CubeMX koristi kako bi generisao ostatak projekta.

Datoteka Makefile sadrži instrukcije za prevođenje koda sistemom make o kome će biti više reči kasnije.

Datoteke startup_stm32f103xb.s i STM32F103C8Tx_FLASH.ld predstavljaju nešto kompleksnije varijante fajlova sa istom ekstenzijom iz prošlog izveštaja[1]. To jest, one sadrže asemblerski kod za pokretanje procesora i instrukcije za linker respektivno.

Direktorijum Drivers sadrži izabrani middleware, u ovom slučaju to je HAL biblioteka.

Direktorijum Inc sadrži header fajlove koji se koriste u projektu a ne pripadaju HAL biblioteci, ovi fajlovi obično odgovaraju .c fajlovima u direktorijumu Src.

Direktorijum Src sadrži izvorne fajlove uključujući i fajlove generisane od strane CubeMX-a koji vrše inicijalnu konfiguraciju.

Datoteka main.c

Skelet projekta koji CubeMX generiše sadrži glavni program koji se nalazi u datoteci `main.c`, unutar nje je takođe generisan skelet koda koji obavlja inicijalizaciju i konfiguraciju mikrokontrolera. Generisani skelet koda sadrži posebne komentare koji označavaju segmente gde korisnik može dopisati svoj kod, bitno je naglasiti da kod treba isključivo pisati unutar ovih segmenata kako ne bi nastali problemi sa CubeMX softverom. Pregled generisanog koda i način njegovog funkcionisanja prevazilazi obim ovog izveštaja.

HAL biblioteka

Hardware Abstraction Layer biblioteka sadrži konstante i funkcije koje znatno olakšavaju rad sa periferijama mikrokontrolera i predstavljaju alternativu direktnom rukovanju registrima pokazanom u prethodnom izveštaju.

Kako je HAL biblioteka veoma opširna preporučuje se upotreba dokumentacije kao što je `Description of STM32F1 HAL and low-layer drivers` a ovde će biti prikazano nekoliko funkcija korišćenih u ovom programu.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */
```

[1] Pogledati Kako (ne) programirati STM32.

Na ovom mestu, pre ulaska u beskonačnu petlju se poziva funkcija koja startuje tajmer tako da su prekidi omogućeni, njen argument je pokazivač na strukturu koja predstavlja taj tajmer (promenljiva htim2 je generisana od strane CubeMX-a).

```
/* USER CODE BEGIN 4 */

void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
{
  if(htim==&htim2)
  {
    HAL_GPIO_TogglePin (GPIOB, GPIO_PIN_13);
  }
}

/* USER CODE END 4 */
```

Implementacija prekidnih rutina je znatno olakšana činjenicom da ih CubeMX automatski generiše i kao argument prosleđuje pokazivač na strukturu tajmera koji je izazvao prekid. Sve što je potrebno uraditi unutar prekidne rutine je proveriti koji tajmer je izazvao prekid i ukoliko je to tajmer 2, funkcijom `HAL_GPIO_TogglePin` menjamo stanje pina. Argumenti funkcije `HAL_GPIO_TogglePin` su port na kome se pin nalazi i broj pina čije stanje se menja, ove konstante su definisane unutar HAL biblioteke.

make

Make je build sistem koji značajno olakšava automatizaciju build procesa u odnosu na proste skripte[2]. Make koristi proste tekstualne fajlove čije ime mora biti `Makefile` (bez ekstenzije) kako bi čuvao informacije o build procesu. Make se pokreće iz komandne linije pokretanjem naredbe make u folderu koji sadrži Makefile, naredbi make se opciono može dodati argument koji specifikuje make cilj koji će se izvršiti (ukoliko se ne navede cilj, izvršava se prvi cilj u fajlu).

Princip funkcionisanja

Princip funkcionisanja make alata se zasniva na konceptu ciljeva i zavisnosti. Make cilj ima ime i predstavlja skup naredbi koje taj cilj izvršava i zavisnosti koje su neophodne tom cilju. Kada se make cilj pokrene izvršavanjem naredbe `make <ime cilja>` (ili samo `make` ukoliko želimo da pokrenemo podrazumevani cilj) make će prvo proverili da li su zavisnosti ažurne[3] pokrenuće istoimeni cilj ukoliko nisu. Kada su sve zavisnosti ažurne, make će konačno izvršiti naredbe definisane za taj cilj.

Proširivanje Makefile-a

Automatski generisan makefile obavlja prevođenje programa ali ne sadrži cilj koji bi preneo program na mikrokontroler, tako da je taj cilj ručno dodat.

```
#######################################
# program
#######################################
prog: $(BUILD_DIR)/$(TARGET).bin
    ST-LINK_CLI -P $(BUILD_DIR)/$(TARGET).bin 0x08000000 -V "while_programming" -Rst
```

Kako ovaj cilj ne generiše nikakve fajlove njegovo ime može biti proizvoljno. Iza dve tačke se navode zavisnosti, u ovom slučaju jedini fajl potreban za prenošenje programa na mikrokontroler je binarni fajl, tako da je on naveden kao zavisnost. Sama naredba je veoma slična onoj pokazanoj u prošlom izveštaju.

Ovo je samo kratak pregled make build sistema, ukoliko želite da saznate više, na internetu postoji obilje dokumentacije.

[2] Nalik na one iz prethodnog izveštaja.

[3] U ovom kontekstu imena zavisnosti (a samim tim i imena ciljeva) bi trebalo da predstavljaju imena datoteka koje su rezultat tog cilja.

Potencijalni problemi prouzrokovani CubeMX generatorom koda

Kako konfiguracija u definisana u CubeMX-u u potpunosti zamenjuje podrazumevanu konfiguraciju mikrokontrolera, greške u konfiguraciji mogu da dovedu do značajnih problema.

Sistemi koji su najpodložniji greškama u konfiguraciji su sistem za generisanje signala takta i sistem za debagovanje i spuštanje koda.

Sistem za generisanje signala takta

Sistem za generisanje signala takta može pravilno da funkcioniše u dve konfiguracije: konfiguracija sa internim oscilatorom i konfiguracija sa eksternim oscilatorom.

Konfiguracija sa internim oscilatorom ne zahteva nikakva dodatna podešavanja osim da bude izabrana kao aktivna konfiguracija. Ukoliko je izabrana konfiguracija sa eksternim oscilatorom (BluePill razvojna ploča ima ugrađen kvarcni oscilator koji radi na frekvenciji od 8MHz), pored izbora ove konfiguracije kao aktivne neophodno je podesiti pinove D0 i D1 kao `RCC_OSC_IN` i `RCC_OSC_OUT` respektivno. Ukoliko ovi pinovi nisu podešeni mikrokontroler neće moći pravilno da se pokrene već je preći na rezervni oscilator i u tom stanju neće moći da izvršava kod. Dok signal takta mikrokontrolera dolazi sa rezervnog oscilatora i dalje je moguće spustiti kod na mikrokontroler tako da ova greška nije preterano problematična.

Sistem za debagovanje i spuštanje koda

Kako bi bilo moguće debagovati mikrokontroler i spuštati kod na njega neophodno je da u konfiguraciji SYS periferije polje **Debug** bude podešeno na `Serial Wire` kao i da **Timebase Source** bude podešeno na `SysTick`. Vredi napomenuti da će ovakva konfiguracija automatski koristiti pinove A13 i A14 za svoje potrebe.

Ukoliko ovaj sistem nije pravilno konfigurisan, nakon spuštanja loše konfigurisanog koda više neće biti moguće spuštanje novog koda na mikrokontroler standardnim metodima. Ukoliko ipak dođe do ovakvog problema postoje dva načina da se povrati mogućnost spuštanja koda: "Connect Under Reset" i izbor drugog moda pokretanja mikrokontrolera korišćenjem džampera povezanog na BOOT0 pin mikrokontrolera. Kako nemam iskustva sa drugom metodom, ovde će biti reči samo o "Connect under reset" metodi.

Connect under reset

Kako bi spustili kod na mikrokontroler korišćenjem connect under reset metode potrebno je da u podešavanjima `STM32 ST-LINK Utility` alata podesite **Mode** na `Connect Under Reset` (Podešavanja se nalaze u meniju `Target`). Kako bi ste se povezali u ovom modu potrebno je da držite reset taster na mikrokontroleru i pritisnete connect u `STM32 ST-LINK Utility`-u a potom otpustite reset taster. Kada se `STM32 ST-LINK Utility` povezao u ovom modu možete spustiti kod sa ispravnom konfiguracijom kako bi povratili mogućnost spuštanja koda standardnim metodima. Nakon što je novi kod spušten možete vratiti mod u podešavanjima na `Normal`.

Zaključak

Nadam se da ste uvideli koliko gore navedeni alati olakšavaju pisanje programa za mikrokontrolere i kako da rešite neke od problema koji se mogu pojaviti pri upotrebi ovih alata. Iako su ovi alati značajno olakšali proces programiranja mikrokontrolera i dalje sam proces nije najkomforniji, te će sledeći izveštaj biti posvećen integrisanim razvojnim okruženjima i procesu debagovanja.