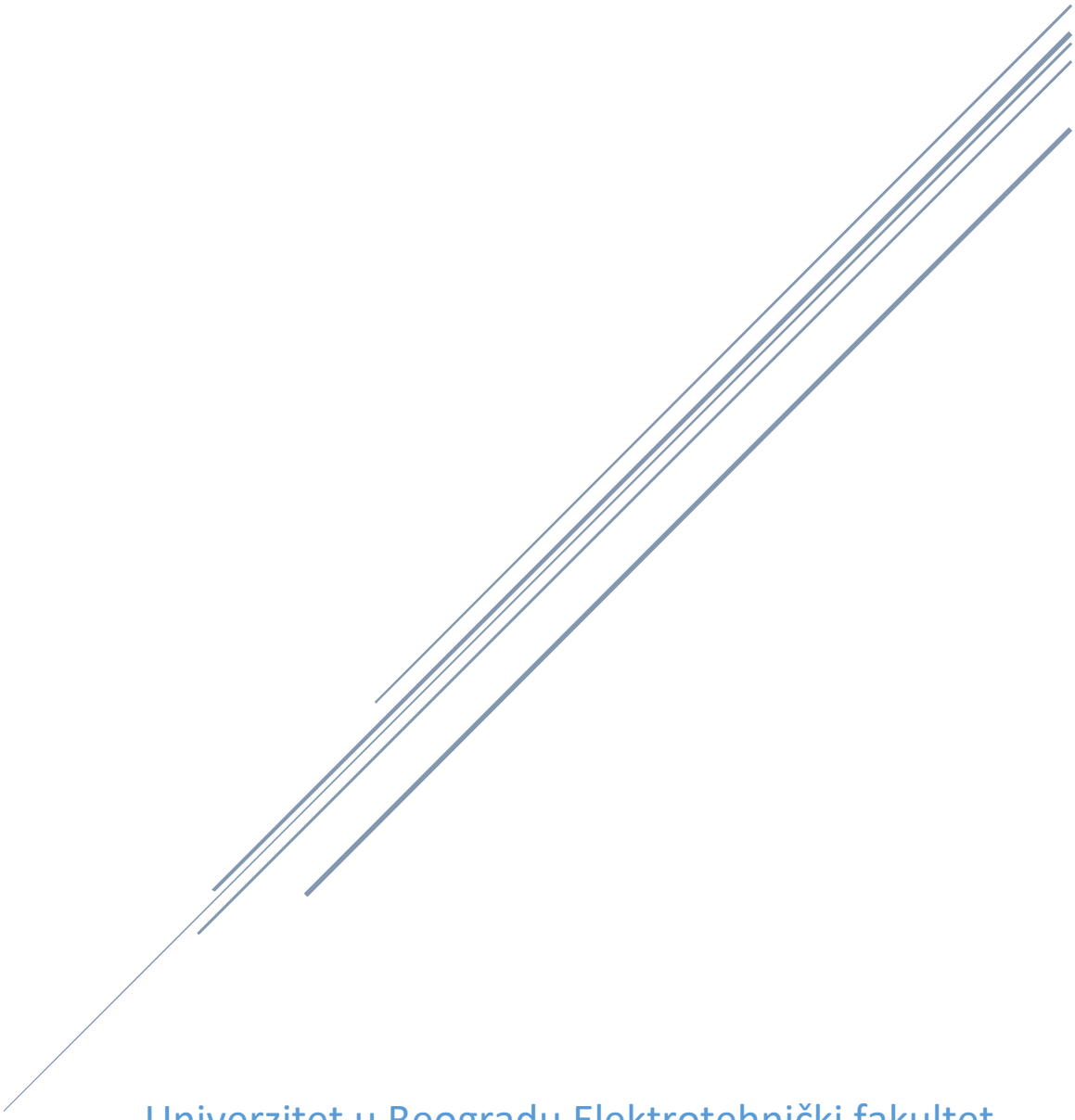
MORZEOVA AZBUKA

STM32F407-VG



Univerzitet u Beogradu Elektrotehnički fakultet.
Mikroprocesorsorski sistemi
Lazar Vasić 2013/0298

Sadržaj

Uvod

O projektu

U ovom projektu se koristi mikrokontroler STM32F407-VG[1]

Korišćenjem STM32F407-VG mikrokontrolera, implementirati Moreov koder i dekoder.

Prilikom implementacije koristiće se LED i tasteri koji se nalaze na pločici mikrokontrolera, displej LCD-WH1602B [2] i UART kabal za serijski prenos.

Projekat se sastoji iz main.c, gpio.h, gpio.c, config.h, led.c, led.h, lcd.c, lcd.h, uart.h, uart.c, timers.h, timers.c, definitions.h fajlova.

Opis implementacije

Konfiguracioni fajl

U fajlu config.h se nalaze makroi kojim se definiše koji pin (i port) se koristi za koju periferiju. Veoma je važno da ovaj fajl bude ispravno inicijalizovan.

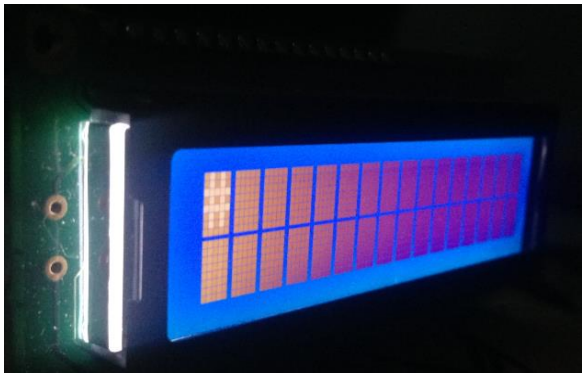
U fajlu definitions.h nalaze se makroi raznih vrednosti koje upisujemo u registre kako bi realizovali neku funkcionalnost. Takođe tu se definišu vremena: vreme pauze i vreme za potrvdu sekvence i razne druge konstante potrebne za izredu zadatka i bolju preglednost i čitljivost koda.

[1] STM32F407-VG – zasniva se na high-performance ARM®Cortex®-M4 32-bit RISC jezgrom na frekfenciji do 168MHz.
[2] LCD-WH1602B – LCD displey sa 2x16 segmenta.

Koriscenje

Slanje

Prilikom ukljucavanja mikrokontrolera na LCD-u se ispisuje ‘#’. Korisnik moze pritiskom na levi taster da izabere da li hoće tačku (dot) ili crticu (dash) zavisno od toga da li prilikom pritiska tastera zadrži kraće ili duže, respektivno. Ako korisnik unese sekvencu od 5 crtica ili tačka ili ako prođe određeno konfigurisano vreme za komfirmaciju odradice se konverzija sekvence u neki karakter, ako sekvecna nije validna ispisaće se znak ‘!’ na LCD-u kao i u UART terminalu. Pritiskom na desni taster menjaju se vremena za potrvdu (CONFIRM_TIME), vreme za tačku/crticu u 4 predefinisane vrednosti (koje mogu da se promene u config.h fajlu). Ako se nakon konverzije napravi pauza trajanja veća od vremana tranja 7 tačaka, smatra se da je korisnik hteo da unese blanko znak (oznacava kraj rečenice).

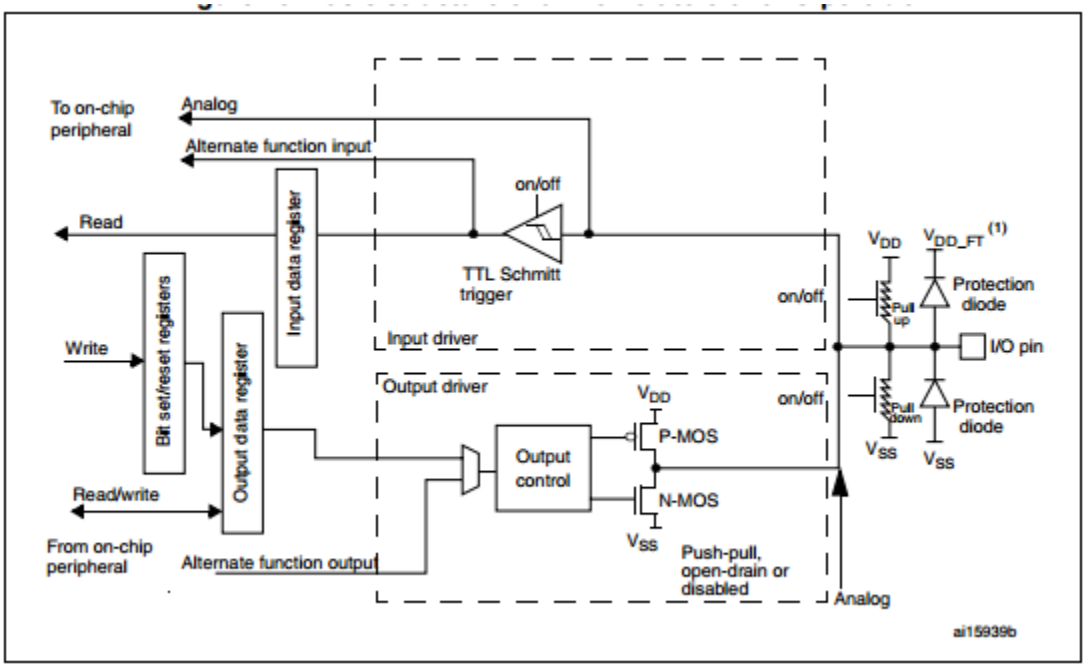


Prijem

Drugi režim rada mikrokontorela je kada korisnik želji da određeni karakter prevede u Morezov kod. On tada iz UART terminala unosi određene karaktere (mala , velika slova i brojeve). Taj karatker se preko UART-a (serijskog prenosa) prenese do mikrokontrolera gde se taj karakter kodira u Moursov kod i ispisuje na LCD displej. Kodirane tačke i crtice se smeštaju u bafer, i čitanjem iz tog bafera na levu LED prikazujemo tačku(kraće blinkanje) ili crticu (duže blinkanje).

GPIO

Svaki pin na mikrokontroleru je GPIO (General Pin Input Output). On sadrži 32-bitne konfiguracione registe: GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR i GPIOx_PUPDR), 2 32-bitna registara za podatke: GPIOx_IDR i GPIOx_ODR. Registar za setovanje i resetovanje GPIOx_BSRR, registar za zaključavanje GPIOx_LCKR i 2 32-bitna registra za selekciju alternativnih funckija.[3]



- U fajovima *gpio.h* i *gpio.c* se nalaze funkcije za inicijalizaciju većinu od ovih regista (onih koji su poterebni za izradu ovog zadatka).
- Registar MODER nam omogućava da odredimo da li naš pin radi u izlaznom, ulaznom, alternativnom ili analognom režimu rada.
- Registar OTYPER nam gotovri da li imamo push-pull optornik ili otvoreni drejn.
- Pomoću registra OSPEEDR regulišemo jednu od brzine.
- PUPDR registar nam služi da odredimo da li koristimo pull-up ili pull-down otpornik.
- Registir IDR (Input Data Register) i ODR (Output Data Register) nam služe kao registi u kojima se smeštaju podaci kojima određeni pin raspolaze u toku njegovor rada.

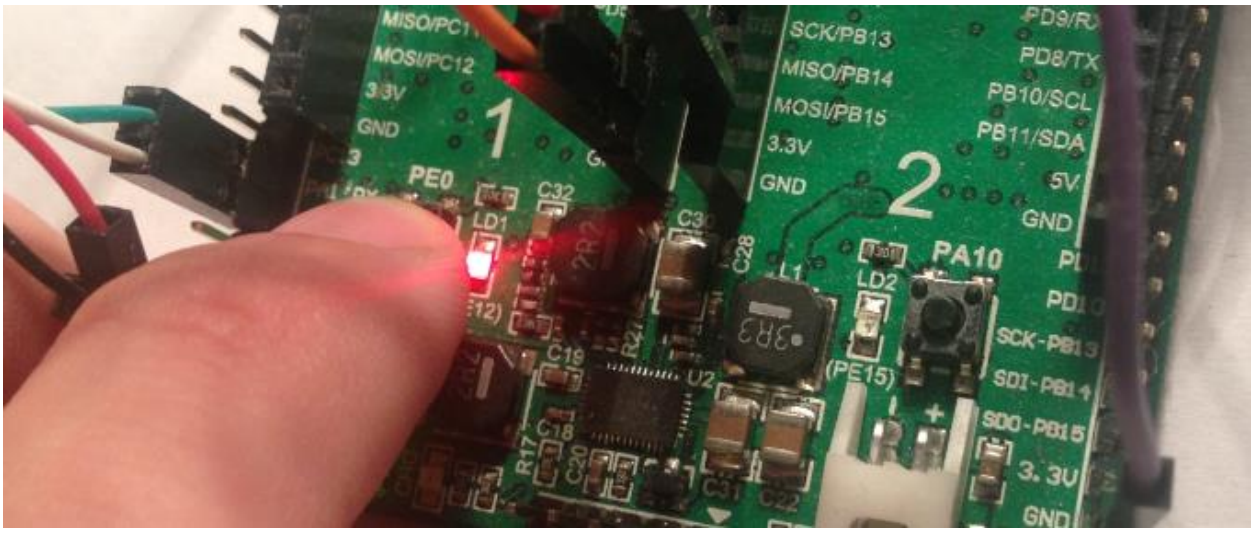
[3] Detaljnije o ovim registrima kaoo vrednostima koje oni koriste moze da se procita u Reference manual za STM32 mikrokontroler.

LED

Imamo 2 LED, i one su ugrađene na ploči mikrokontrolera na fiksnim pinovima PE12 i PE15. GPIO za diode se inicijalizuje tako da su ti pinovi izlazni pinovi. U fajlovima led.h i led.c moze da se vidi konfiguracija pinova za LED.

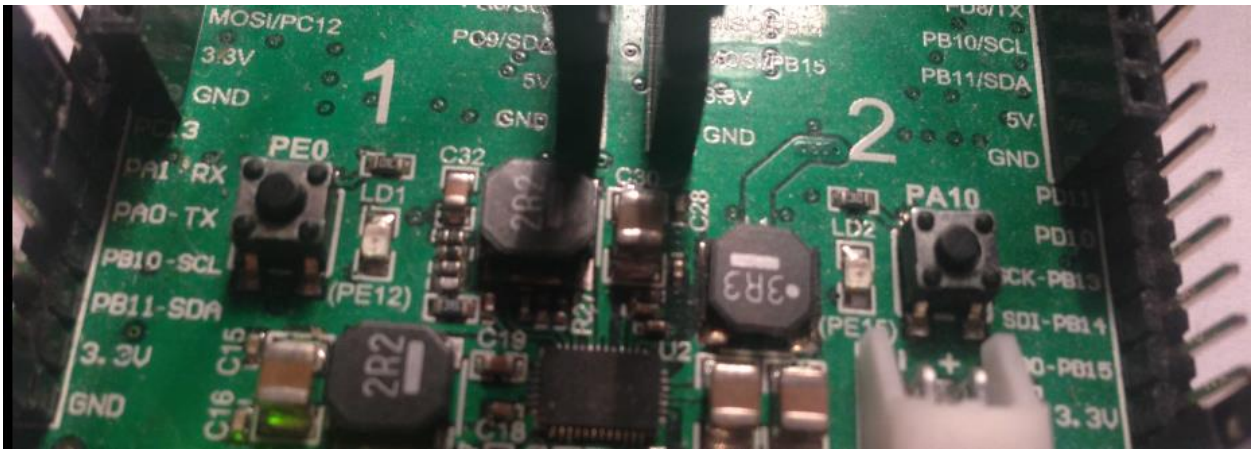
```
#include "led.h"

// GPIO: PE12(left), PE15(right)

void init_LEDs(){
    RCC_AHB1ENR    |= (1UL << 4);       // Enable GPIOD clock

    init_GPIO_Pin(MODER,   LEFT_LED_PORT,  LEFT_LED_PIN,  OUTPUT);
    init_GPIO_Pin(OTYPER,  LEFT_LED_PORT,  LEFT_LED_PIN,  PUSH_PULL);
    init_GPIO_Pin(OSPEEDR, LEFT_LED_PORT,  LEFT_LED_PIN,  HIGH);
    init_GPIO_Pin(PUPDR,   LEFT_LED_PORT,  LEFT_LED_PIN,  PULL_UP);

    init_GPIO_Pin(MODER,   RIGHT_LED_PORT, RIGHT_LED_PIN, OUTPUT);
    init_GPIO_Pin(OTYPER,  RIGHT_LED_PORT, RIGHT_LED_PIN, PUSH_PULL);
    init_GPIO_Pin(OSPEEDR, RIGHT_LED_PORT, RIGHT_LED_PIN, HIGH);
    init_GPIO_Pin(PUPDR,   RIGHT_LED_PORT, RIGHT_LED_PIN, PULL_UP);

}
```

Leva dioda radi u dva slucaja, prvi je kada preko levog tastera unosimo sekvence crtice/tačke i u zavisnosti od dužine pritiska dioda ce toliko svetleti. U drugom slučaju kada preko serijskog prenosa dobijemo neki karakter on se dekodira i onda ce dioda da pokaze kodiranu sekvencu, tako što za tačku svetli manji 3 puta manji interval nego za crticu (ovo vreme je definisano u config.h fajlu).



Button

Postoje 2 tastera, i oni su fiksirani na mikrokontoleru na pinovima PE0 i PA10. U fajlovima button.h i button.c se nalazi konfiguracija ovih pinova. GPIO za taster se postavlja da radi u ulaznom režimu.



Levi taster (button 1) za funkciju ima davanje Morzeuvog signala. Kracim pritiskom ovog tastera generise se tačka (dot) dok se dužim pritiskom generiše crtica (dash).

Vremena trajanje crtice i tačke (kao i blanko znaka) mogu da se postavljaju drugim desnim tasterom (button 2). U fajlu config.h postoje unapred definisana vremena. Ako zelimo da za crticu zelimo manje ili veće vreme pritiska tastera koristicemo ovaj taster. Na LCDu prilikom pritiska ovog tastera bice nam ispisana brzina sa komom se radi u formatu : SPEED:x , gde je x e (1,2,3,4). Duzim pritiskom tastera 2 brišemo sve sto je bilo na LCD i iskljucavamo diode.

Realizacija tastera i njihovih funkcionalnosti data je u prekidnim rutinama. Svaki put kada neko pritisne ili otpusti neki od tastera ulazi se u prekidnu rutinu i tu se nalazi odgovarajuća obrada.

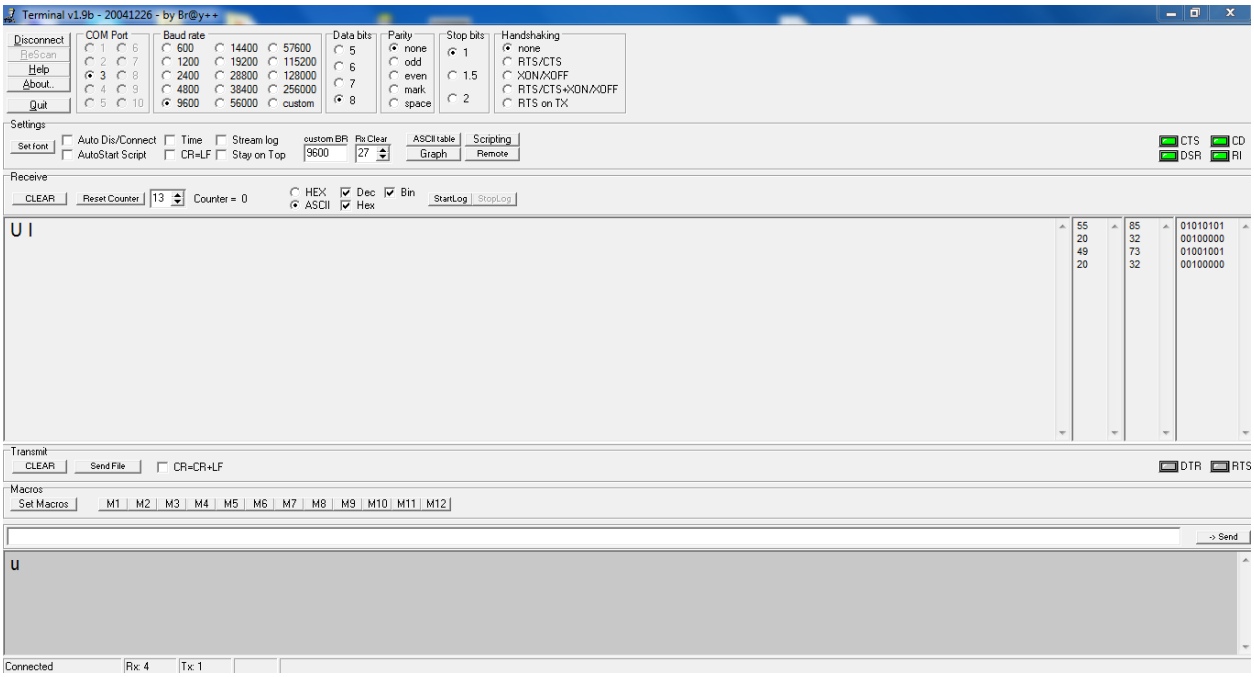
LCD

LCD (LCD-WH1602B) je priključen na pinovima PC0, PC1, PC2, PC3, PC4 i PC13. (Odabir ovakvih pinova moze da bude drugačiji, u mom slucaju izabrao sam pinove tako da budu sto gušće raspoređeni na mikrokontorleru). Ako želimo da izaberemo drugačiji raspored pinovi u config.h fajlu mora da se specifiira svaki od tih pinova. LCD radi sa 4-bitnim prenosom, postoji i 8-bitni prenos samo sto on zahteva angažovanje još 4 pinova. 4-bitnim prenosom se jedan karakter od 8b prenosi u 2 ciklusa. U *lcd.h* i *lcd.c* fajlovima nalazi se konfiguracija pinova, i dodatne funkcije za ovu periferiju. Kako bi se uspedno koristio LCD prvo mora da se inicijalizuje, a zatim da se koriscenjem određenih sekvenci izvršavaju razlicite operacije (čiscenje ekrana, setovanje i razliciti režimi kursora, ispis teksta…). Sve ovo je moze da se nađe u dokumentaciji.

RB	Opis	Simbol	Pin
1	Napanjanje	Vss	5V
2	Uzemljenje	Vdd	GND
3	Kontrast	Vo	GND
4	RS	RS	PC4
5	Read/Write	R/W	GND
6	Enable	E	PC13
7	Pin za podatke	DB0	NU[4]
8	Pin za podatke	DB1	NU
9	Pin za podatke	DB2	NU
10	Pin za podatke	DB3	NU
11	Pin za podatke	DB4	PC
12	Pin za podatke	DB5	PC
13	Pin za podatke	DB6	PC
14	Pin za podatke	DB7	PC
15	Pozadinsko osvetljenje	A	3,3V + otpornik
16	Pozadinsko osvetljanje (0V)	K	GND

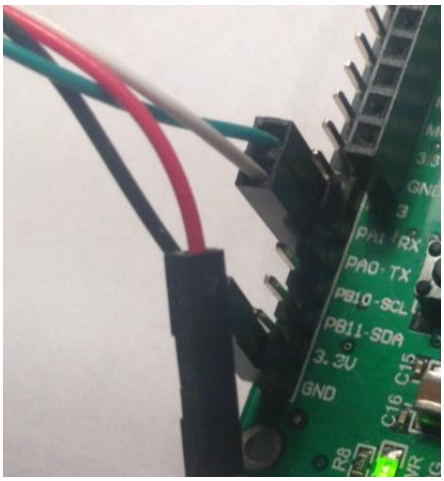
[4] NU – not used

UART



UART terminal

UART je konfigurisan u *uart.h* i *uart.c* fajlovima. Koristi se baud rate 9600bpr i shodno tim je izracunata vrednost koja treba da su upise u UART4_BRR registar. UART koji koristimo na ovom mikrokontoleru je UART4. Prenos i prijem je realizovan preko prekidnih rutina, tako sto je prekid za prijem uvek aktivan jer nikad ne znamo kad korisnik može iz terminala da pošalje neki karakter, dok prekid za slanje nam se aktivira samo kada imamo nešto da pošaljemo. UART koristi 3 pina, jedan za masu (GND), transmiter i receiver (PA0 i PA1).



Raspred pinova za UART4

$$UART4\_BRR_{15:0} = MANTISA_{15:4} + FRACTION_{3:0}$$

$$baud = \frac{fck}{16 * USARTDIV}$$

$$baud = 9600$$

$$fck = 16000000\ (16MHz)$$

$$USARTDIV = 104.17$$

$$FRACTION = Round\big(16 * decimal(USARTDIV)\big) = 16 * 0.17 = Round(2.73) = 3$$

$$MANTISA = 104 = 68h$$

$$BRR = 683h$$

Glavni program

Glavni program inicijalizuje sve navedene periferije. Posto je svaka obrada odrađena preko prekida, svaki put kada uđemo u idle petlju glavnog programa naš mikrokotroler ce ući u sleep režim rada. Jedan od načina da se ovo odradi je tako što se pozove instrukcija WFI (Wait For Interrupt). Ovako mikrokontoler spava sve dok ga neka prekidna rutina ne probudi.

```
void doSomething(){
    while(1)
        asm WFI; // The wait for interrupt instruction, WFI, causes
immediate entry to sleep mode

}

void main(){
    init_all();     // Initalize LCD, LEDs, Buttons, UART, Timer...
    Delay_ms(10);   // Delay for stabilization initialization
    doSomething();  // Infinite loop (dummy loop)

}
```