Univerzitet u Beogradu
Elektrotehnički fakultet

Jovan Blanuša, 47/2012


# EUROBOT RS485 komunikacija

*projekat iz predmeta Namenski Računarski sistemi*

mentor:
dr Nenad Jovičić

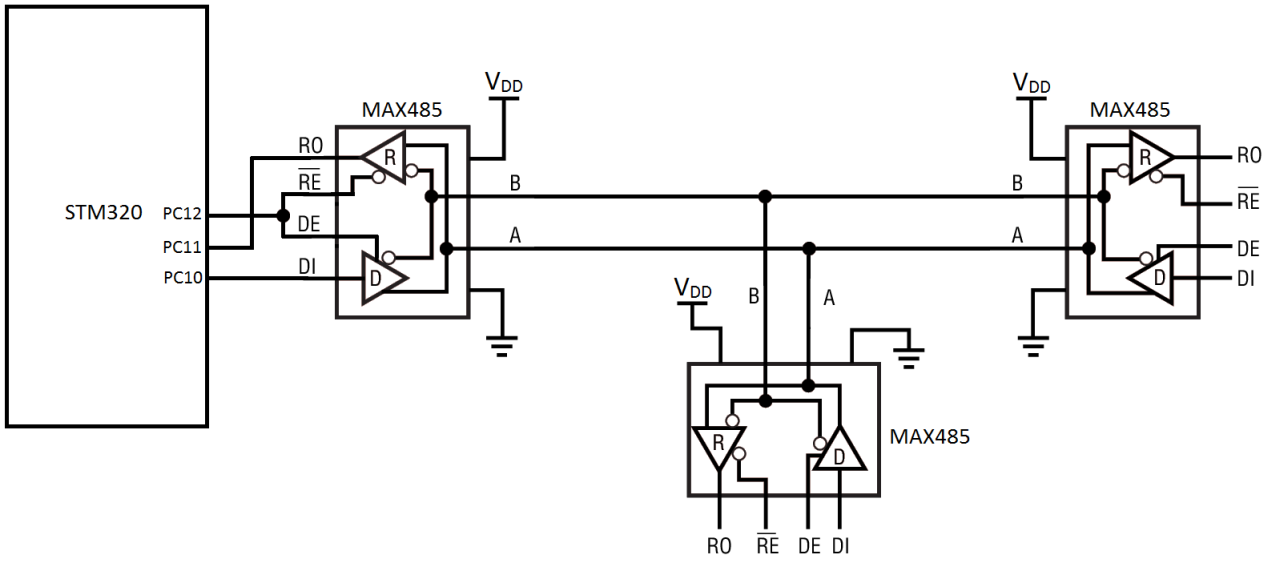
Beograd, april 2016.

# Sadržaj

# Glava 1

# Uvod

Ovaj dokument predstavlja opis drajvera za RS485 komunikaciju korišćenu za EUROBOT takmičenje. Komunikacija se obavlja izmedju više različitih ploča na kojima se nalazi STM32. Na slici 1 je data blok šema primera kako se više različitih ploča povezuje pomoću RS485 magistrale. Drajver za komunikaciju je razvijan za STM32 u programskom jeziku C korišćenjem *Standard Peripheral Library*-ja.




Slika 1.1: Blok šema dela za komunikaciju

# Glava 2

# Protokol poruke

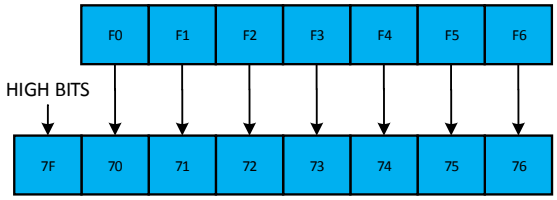
Komunikacioni protokol prema kom se šalju poruke preko magistrale je dat na slici 2.


Slika 2.1: Prikaz komunikacionog protokola

Prvo se šalje *START* bajt, čija je vrednost $0xFF$, i on označava početak poruke. Sledeći bajt predstavlja adresu uređaja na koji je poruka upućena. Potom sledi bajt koji predstavlja dužinu poruke $N$ koja sledi. Prati ga korisni deo poruke veličine $N$ bajtova. Na kraju se nalazi bajt koji predstavlja *check* sumu.

*Check* suma se koristi da bi se ispitala ispravnost poruke. Svi prethodni bajtovi sem start bajta se sabiraju i prvih 7 bita rezultata se poredi sa pristiglom *check* sumom. Ako su ove dve vrednosti jednake, poruka je validna, u suprotnom se poruka odbacuje.

Do greške u komunikaciji može da se desi ako jedan bajt u korisnom delu poruke ima vrednost $0xFF$ što ustvari predstavlja *START* bajt. To je rešeno na sledeći način: Poruka se podeli na grupe od po 7 bajtova, iz svakog bajta se izdvoji bit najveće težine i tih 7 bita se smesti u poseban bajt *HIGH BITS* koji se šalje pre grupe od 7 bajtova. Ovo je ilustrovano na slici 2.




Slika 2.2: Rešenje gore-pomenutog problema

Na ovaj način svaki bajt nosi 7 bita poruke i ne može da ima vrednost $0xFF$, pa će poruka biti ispravno poslata.

# Glava 3

# *API* drajveri za komunikaciju

Drajver za komunikaciju se nalazi u fajlovima `EUROBOT_serial.h` i `EUROBOT_serial.c`. U ovim funkcijama su definisane sledeće funkcije:

- `initEurobotRS485(uint32_t BaudRate, uint8_t master_addr, uint8_t this_addr, int is_master)`
- `void SendString(unsigned char address, unsigned char* message)`
- `void SendMessage(unsigned char address, unsigned char* message, char length)`
- `void SendACK(unsigned char address)`
- `char* GetMessage()`
- `char GetAddress()`
- `char GetMessageLength()`
- `int IsAck()`

Takođe, korisnik treba da definiše funkciju `void DecodeCommand()` koja se izvršava u prekidnoj rutini za USART3 kada pristigne cela poruka.

## 3.1 *API* funkcije

### 3.1.1 `initEurobotRS485`

| | |
|---|---|
| Ime Funkcije | `void initEurobotRS485(uint32_t BaudRate, uint8_t master_addr, uint8_t this_addr, int is_master)` |
| Opis Funkcije | Iniciranje RS485 komunikacije |
| Parametri | • Baud Rate transfera<br>• Adresa mastera na magistrali<br>• Adresa ovog uredjaja<br>• Da li je ovaj uredjaj master na magistrali |
| Povratna vrednost | Nema |
| Napomene | Proveriti prioritet prekida koji se u ovoj funkciji postavlja, izmeniti po potrebi |

### 3.1.2 SendString

| | |
|---|---|
| Ime Funkcije | `void SendString(unsigned char address, unsigned char* message)` |
| Opis Funkcije | Slanje stringa preko USART/RS485 |
| Parametri | • Adresa uredjaja na koju se salju podaci<br>• Poruka koja se salje na uredjaj. Potrebno je da bude u formatu stringa tj da se iza kraja poruke postavi znak '\0' odnosno 0x0 |
| Povratna vrednost | Nema |
| Napomene | Ako nije poznato da li će poruka u sebi sadržati bajtove vrednosti 0 bolje da se koristi funkcija `SendMessage` |

### 3.1.3 SendMessage

| | |
|---|---|
| Ime Funkcije | `void SendMessage(unsigned char address, unsigned char* message, char length)` |
| Opis Funkcije | Slanje poruke preko USART/RS485 |
| Parametri | • Adresa uredjaja na koju se salju podaci<br>• Poruka koja se salje na uredjaj u obliku niza bajtova<br>• Duzina poruke u bajtovima |
| Povratna vrednost | Nema |
| Napomene | Nema |

### 3.1.4 SendACK

| | |
|---|---|
| Ime Funkcije | `void SendACK(unsigned char address)` |
| Opis Funkcije | Slanje poruke kojom se signalizira uspesan prijem poruke |
| Parametri | • Adresa uredjaja na koju se salju podaci |
| Povratna vrednost | Nema |
| Napomene | ACK je prazna poruka dužine 0 |

### 3.1.5 GetMessage

| | |
|---|---|
| Ime Funkcije | `char* GetMessage()` |
| Opis Funkcije | Preuzimanje poslednje validne poruke koja je stigla |
| Parametri | • Nema |
| Povratna vrednost | String koji predstavlja koristan deo pristigle poruke |
| Napomene | Koristiti je u funkciji `DecodeCommand` |

### 3.1.6 GetAddress

| | |
|---|---|
| Ime Funkcije | `char GetAddress()` |
| Opis Funkcije | Preuzimanje adrese na koju treba da stigne poslednja validna poruka |
| Parametri | • Nema |
| Povratna vrednost | Adresa na koju je adresirana poslednja validna poruka |
| Napomene | Nema |

### 3.1.7 GetMessageLength

| | |
|---|---|
| Ime Funkcije | `char GetMessageLength()` |
| Opis Funkcije | Dohvatanje duzine posledenje validne poruke |
| Parametri | • Nema |
| Povratna vrednost | Duzina poslednje validne poruke |
| Napomene | Nema |

### 3.1.8 IsAck

| | |
|---|---|
| Ime Funkcije | `int IsAck()` |
| Opis Funkcije | Provera da li je poslednja validna poruka ACK |
| Parametri | • Nema |
| Povratna vrednost | Informacija da li je ACK poslednja poruka |
| Napomene | Koristiti je u funkciji `DecodeCommand` |

## 3.2 Primer korišćenja *API*-ja

```
1  //*********************************************************
2  //******************** MASTER *****************************
3  //*********************************************************
4
5  #include "stm32f10x.h"
6  #include "EUROBOT_serial.h"
7
8  #define MASTER_ADDR ...
9  #define MOTOR_DRIVER_ADDR ...
10
11 volatile int ACKflag = 0;
12
13 void DecodeCommand(){
14   if(IsAck()) ACKflag = 1;
15   // . . . .
16 }
17 void delay(int milis){ . . . }
18
19 int main(void)
20 {
21     initEurobotRS485(115200, MASTER_ADDR, MASTER_ADDR, 1);
22
23   int encoder_tick = ... ;
24   int i;
25     while(1)
26     {
27
28     char message[5];
29     message[0] = 'f'; //Forward
30     message[1] = encoder_tick & 0x000000FF;
31     message[2] = (encoder_tick >> 8) & 0x000000FF;
32     message[3] = (encoder_tick >> 16)  & 0x000000FF;
33     message[4] = (encoder_tick >> 24)  & 0x000000FF;
34     // Probaj 5 puta da posaljes poruku
35     for(i = 0; i < 5; i++){
36       SendMessage(MOTOR_DRIVER_ADDR, message, 5);
37       delay(20);
38       if(ACKflag == 1) {
39         ACKflag = 0;
40         break;
41       }
42     }
43     if(i == 5) // Greska u prenosu
44     // . . . .
45
46     }
47 }
```

master.c

```
1
2  //*************************************************************
3  //********************* SLAVE *********************************
4  //*************************************************************
5
6  #include "stm32f10x.h"
7  #include "EUROBOT_serial.h"
8
9  #define MASTER_ADDR ...
10 #define MOTOR_DRIVER_ADDR ...
11 #define LENGTH ...
12
13 volatile int flag = 0;
14 volatile char message[LENGTH];
15 volatile char* ptr;
16
17 void DecodeCommand(){
18   ptr = GetMessage();
19   // Kopiranje poruke
20   for(int i = 0; i < GetMessageLength(); i++)  message[i] = ptr[i]
21
22   flag = 1;
23 }
24
25 int main(void)
26 {
27   // . . . .
28     initEurobotRS485(115200, MASTER_ADDR, MOTOR_DRIVER_ADDR, 1);
29
30   char command;
31   int encoder_tick;
32     while(1)
33     {
34     // . . . .
35     if(flag == 1){
36       command = message[0];
37       encoder_tick = message[0] + message[1] * 0x00000100 +
             message[2] * 0x00010000  + message[3] * 0x01000000;
38       flag = 0;
39     }
40
41     switch(command){
42       case 'f' :
43       // ...
44     }
45     // . . . .
46
47     }
48 }
```

Ovi primeri ilustruju korišćenje *API* funkcija u koje šalju broj otkucaja enkodera sa Mastera na Slave. Master pakuje u jednu poruku komandu koja je predstavljena jednim bajtom i otkucaj enkodera koji je predstavljen sa 4 bajta. To se obavlja pomoću funkcije `SendMessage`.

Po slanju poruke, Master čeka se ACK signal koji javlja da je poruka uspešno poslata. Ako nije, šalje se ponovo. To se ponavlja na primer 5 puta, i ako ponovo ne stigne ACK signal, postoji greška u komunikaciji. ACK signal se prima u `DecodeCommand` funkciji, i preko flega se signalizira glavnom programu da je stigao.

Slave prima poruku takođe u funkciji `DecodeCommand`. Prvo se kopira poruka u interni niz pomoću `GetMessage` funkcije, i određenim flegom se signalizira glavni program da je poruka stigla. U glavnom programu se raspakuje poruka koja je dobijena i obavljaju se određene akcije u zavisnosti od poruke.

# Glava 4

# Opis rada internih funkcija

## 4.1 Slanje poruke

Poruka se šalje pomoću funkcija `SendMessage` ili `SendString`. One interno pakuju podatke za slanje u sledeću strukturu:

```
1 typedef struct {
2   uint8_t *data;                    // Podatak koji se salje
3   int message_length;                 // Duzina poruke
4   unsigned int iter;                  // Iterator za prenos
      podataka
5 } send_package;
```

Ova struktura se popunjava tako da protokol bude zadovoljen. Kad se struktura popuni, ona se ubacuje u red za čekanje. Ako je red za čekanje bio prazan, prvi bajt poruke se prosleđuje direktno na USART, a ostali bajtovi poruke se šalju pomoću prekidne rutine. Deo prekidne rutine koji se bavi slanjem poruke je:

```
1  if (peekQ()->iter < peekQ()->message_length && !is_emptyQ())
2  {
3    USART_SendData(RS485,  peekQ()->data[peekQ()->iter]);
4    peekQ()->iter++;
5  }
6  else
7  {
8    delete_lastQ();
9    if(!is_emptyQ())
10   {
11     USART_SendData(RS485,  peekQ()->data[peekQ()->iter]);
12   }
13   else
14   {
15     DisableRS485();
16   }
17 }
```

Funkcije reda za čekanje su sledeće:

| | |
|---|---|
| `int is_emptyQ()` | Provera da li je red prazan |
| `void insertQ(send_package* data)` | Ubacivanje elementa na kraj reda |
| `void delete_lastQ()` | Brise prvu poruku u redu |
| `send_package* peekQ()` | Nedestruktivno čita prvu poruku u redu |

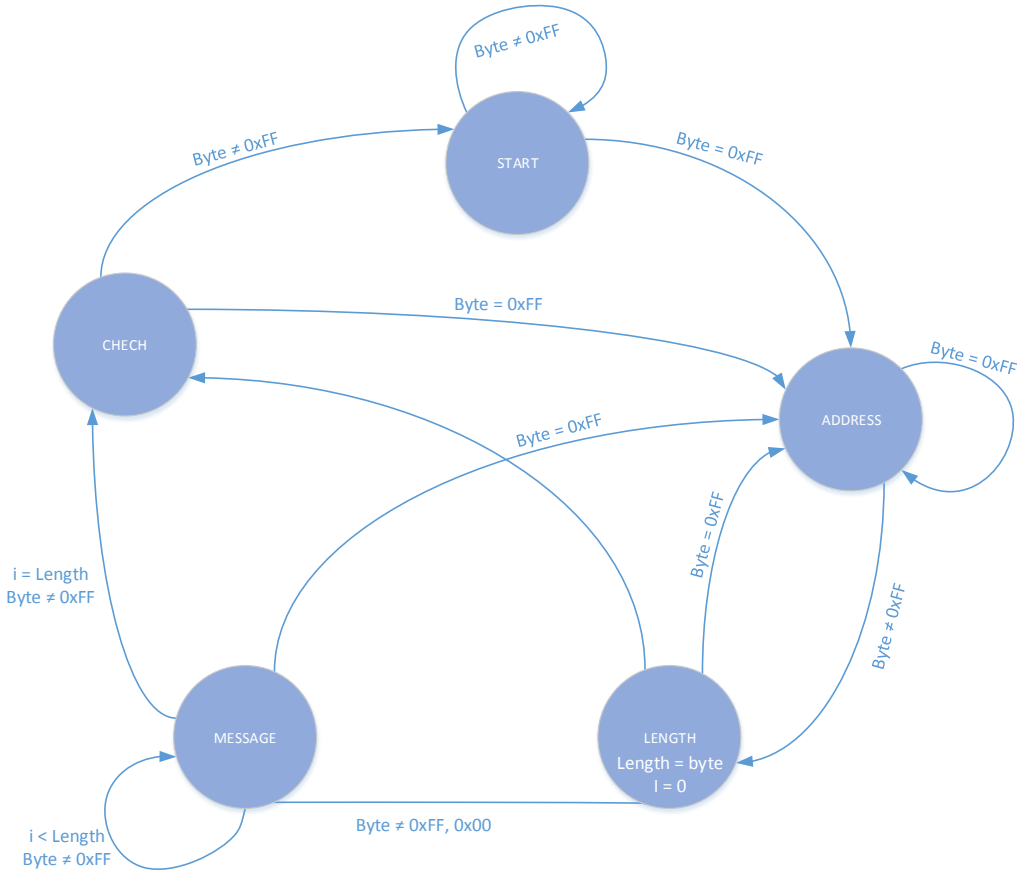
Prekidna rutina šalje bajt po završetku prenosa prethodnog bajta. Kada se pošalju svi bajtovi jedne poruke, proverava se da li je red za slanje prazan. Ako nije, započinje se slanje sledeće poruke, inače se isključuje RS485 prenos.

## 4.2 Prijem poruke

Prijem bajta se detektuje u prekidnoj rutini za USART3 kao što je prikazano ispod:

```
1    received_byte = (char)USART_ReceiveData(RS485);
2    ProcessByte(received_byte);
```

U prekidnoj rutini se poziva funkcija `ProcessByte` koja je u stvari mašina stanja. Izgled ove mašine stanja je dat na slici 4.2.




Slika 4.1: Mašina stanja za prijem poruke

Mašina stanja je realizovana u skladu sa protokolom za slanje poruke. U stanju `CHECK` se vrši provera da li je poruka validna, tj da li je *check* suma odgovarajuća. Ako jeste izvršava se sledeći deo koda:

```
1  ExtractMessage(received.prev_data,(char*)received.data);
2  received.prev_length = received.message_length -
       (received.message_length/8 + (received.message_length % 8 != 0));
3  received.prev_address = (char) received.address;
4
5  // Ako je ovo SLAVE vrati ACK
6  if(!ThisDevice.is_master) {
7    if(!IsAck() && ThisDevice.this_addr == received.address)
       SendACK(ThisDevice.master_addr);
8  }
9
10   DecodeCommand();
11 }content
```

Funkcija `ExtractMessage` daje konvertuje poruku iz oblika koji je bio pogodan za protokol(7 korisnih bita po bajtu, slika 2) u željenu poruku. Takođe se određuje dužina korisnog dela poruke.

ACK signal se prosleđuje masteru samo ako je u pitanju Slave uređaj, i ako je adresa odgovarajuća. Ovde je pretpostavljeno da Slave uređaju ne treba ACK signal od Mastera kad mu Slave pošalje poruku. Moguće je ispraviti da se ACK šalje u svakom slučaju.

Na kraju se izvršava `DecodeCommand` funkcija koju korisnik drajvera definiše u zavisnosti od svojih potreba. U toj funkciji može da se detektuje da li je stigao ACK signal, ili da se preuzme primljena poruka. Ova funkcija se izvršava u prekidnoj rutini, tako da ne bi trebalo da se dugo izvršava.

# Napomene

- Onaj ko bude koristio ovaj drajver treba da vidi da li je način slanja ACK signala odgovarajući. Ako je potrebno da se menja(Da se doda ID poruke i ACK-a koji mu šalje itd), slanje ACK signala se nalazi u funkciji ProcessByte, case: CHECK
- Potrebno je definisati adrese uređaja koji se šalju.
- Prioritet prekida USART3 prekidne rutine je podesen u funkciji initEurobotRS485. Po potrebi ga promeniti, ja nisam znao koliki prioritet treba da bude.
- Ne bi trebalo slati dosta poruka jednu za drugom(Jedna komanda `SendMessage` iza druge). Iako bi komunikacija i tad trebala da radi, može da dodje do zatrpavanja memorije.