Serijska komunikacija mikrokontrolera

UART = Universal Asynchronous Receiver/Transmitter

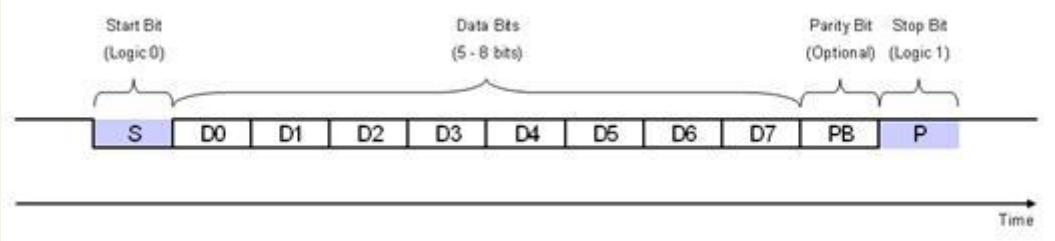
Ovo je osnova svake serijske komunikacije bilo na razini računala ili kontrolera i služi za pretvorbu paralelnog podatka (najčešće 8-bitnih odnosno 1 byte) u serijski oblik sa svrhom da se smanji potreban broj linija za komunikaciju odnosno omogući prijenos podataka kroz druge medije (npr. optiku). U predajniku se paralelni podatak pretvara u serijski niz bitova dok ga prijemnik ponovo vrača u paralelni. Za komunikaciju se koriste dva pina, Tx-predajni i Rx-prijemni. Na razini UART-a komunikacija je dvosmjerna (full duplex) što znači da je moguć istovremeni prijem i predaja podataka. S obzirom da je komunikacija asinkrona, da bi prijemnik mogao ispravno dekodirati poslani niz bitova predajnika potrebno je zadovoljiti još nekoliko osnovnih zahtjeva:

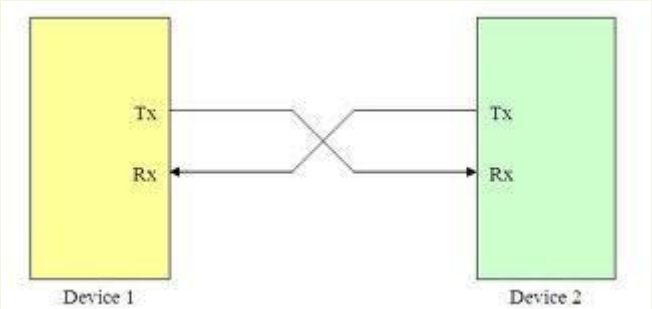
- neaktivno stanje RX/TX linije je digitalna jedinica "1"
- svako slanje podatka započinje START bitom koji je uvijek digitalna nula "0"
- nakon START bita odašilju se bitovi podatka (najčešće 8) počevši od bita najniže važnosti (LSB)
- ponekad se iza bitova podatka šalje i paritetni bit pomoću kojeg se utvrđuje ispravnost prijema podatka (za sada ću ga zanemariti)
- slanje podatka završava STOP bitom koji je uvijek "1" (vračanje u neaktivno stanje) trajanje svakog bita definirano je brzinom komunikacije u broju bitova u sekundi (Baud/kBaud) i mora biti jednako podešeno na oba kontrolera kao i broj bitova podataka, paritetni bit i broj stop bitova.

HARDVERSKA SUČELJA

TTL

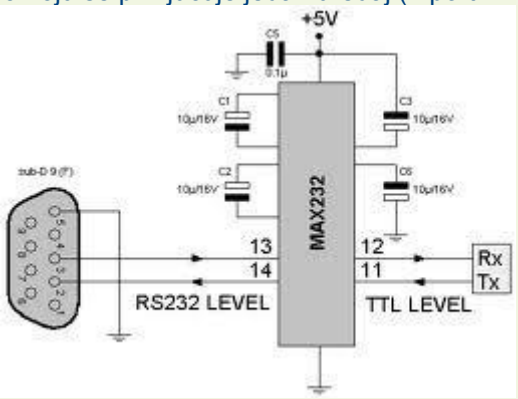
Ovo su priključci samog UART čipa odnosno mikrokontrolera. Komunikacijski pinovi imaju razine napona napajanja "1"=5 (3,3)V "0"=0V. Najjednostavniji hardwerski način da serijskom komunikacijom povežemo dva kontrolera je da TX pin kontrolera A povežemo s RX pinom kontrolera B i naravno TxB na RxA. Navedeno je osnova koja se može realizirati na relativno kratkim udaljenostima (rekao bih do možda pola metra ali nisam eksperimentirao jer sam za kratke udaljenosti koristio druge brže standarde (I2C SPI)) te se za veće udaljenosti koriste standardni konverteri opisani u nastavku. Važno je napomenuti da sve udaljenosti u ovom opisu ovise o brzini prijenosa podataka na način da manja brzina omogućuje veću udaljenost dok je maksimalna udaljenost ovisna o upotrjebljenom tipu konvertera.





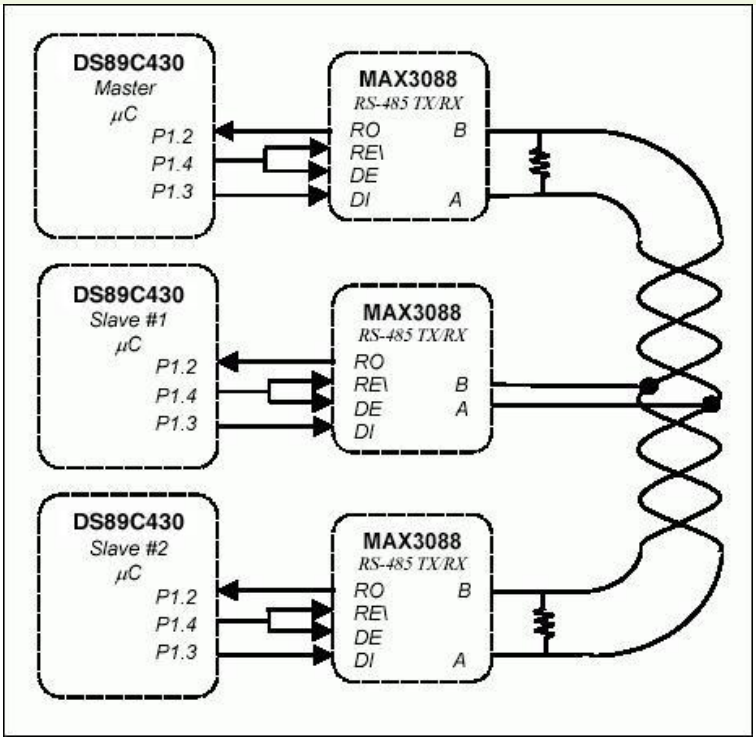
RS232

Donedavno standard koji se na PC računalima koristio za priključivanje printera, miša, modema ... za uređaje do udaljenosti cca 15m a danas je sve više zamijenjen USB komunikacijom. Osnovna hardverska razlika između UART-a i RS232 su naponi logičkih razina digitalnih signala. TTL razina UARTa 5V ("1") konvertira se u -12V dok se logička "0" konvertira i +12V. Ova promjena naponskih razina omogućila je povećanje udaljenosti uređaja koji komuniciraju. Za ovu konverziju koriste se standardni čipovi poput MAX232 i sl. Kod RS232 standarda ponekad se koriste i dodatne linije (pinovi) za hardversku kontrolu komunikacije koji prosljeđuju signale poput RTS-zahtjev za predaju, CTS-Spreman za slanje, DTR-Spreman za prijem, RI-indikacija zvonjave modema. RS232 je "full duplex" komunikacija na koju se priključuje jedan uređaj (1 port= 1 uređaj).



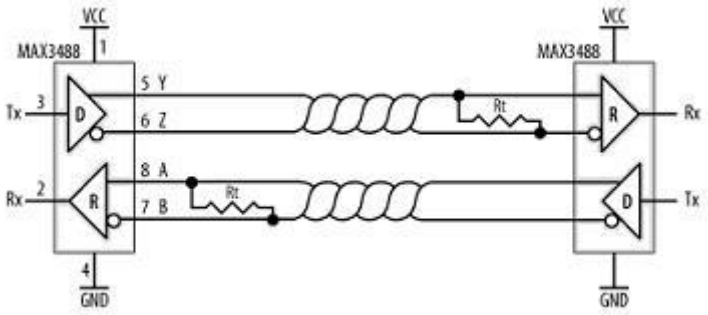
RS485

Standard koji se najčešće koristi za komunikaciju u industrijskom okruženju. Najznačajnija prednost ovog standarda je neosjetljivost na smetnje s obzirom da se za komunikaciju koristi diferencijalna balansirana linija (upletena parica-dvije žice).Duljina komunikacijske parice kod RS485 standarda može biti i do 1200m te je na istu komunikacijsku paricu moguće priključiti do 32 ili čak 256 uređaja. Kao manu moglo bi se navesti da po RS485 standardu komunikacija u jednom trenutku može biti samo u jednom smjeru (ili prijem ili predaja) odnosno nije moguć istovremeni prijem i predaja podataka s istog uređaja. Na razini kontrolera za ovu komunikaciju potrebno je koristiti još jedan dodatni pin koji definira smjer podataka (prijem/predaja). S obzirom na navedeno RS485 komunikacija se softverski izvodi na master/slave principu na način da je na sabirnici 1 master te 1 ili više slave uređaja te se u komunikaciji koristi standardizirani ili korisnički definirani protokol. Hardverski se za konverziju UART/RS485 koriste čipovi poput ADM485 ili MAX3088.



RS422

Za ovaj standard vrijedi sve gotovo isto kao i za RS485 s tom razlikom da se umjesto jedne parice (dvije žice) koriste dvije parice (4 žice) te da je kod RS422 moguć istovremeni prijem i predaja na jednom uređaju.



Strujna petlja (Current Loop)

Ovo sučelje se rijetko koristi pa ga neću posebno opisivati osim spominjanja jedne interesantne značajke ovog sučelja. Želite li istovremeno komunicirati i napajati uređaj na drugoj strani (npr. neki senzor ili display) po dvije žice ovaj tip sučelja je idealan za to.

KOMUNIKACIJSKI PROTOKOLI

definiraju što predajni uređaj treba poslati da bi prijemni uređaj znao što uraditi s primljenim podacima odnosno što odgovoriti na određeni upit ili poruku. Kod komunikacije uređaja po TTL-u ili RS232 koji u pravilu povezuju samo ta dva uređaja (point to point) ne bih govorio o nekom specificiranom komunikacijskom protokolu jer su oni vrlo raznoliki i ovisni o uređajima čak bi se moglo reći "Koliko uređaja toliko protokola".

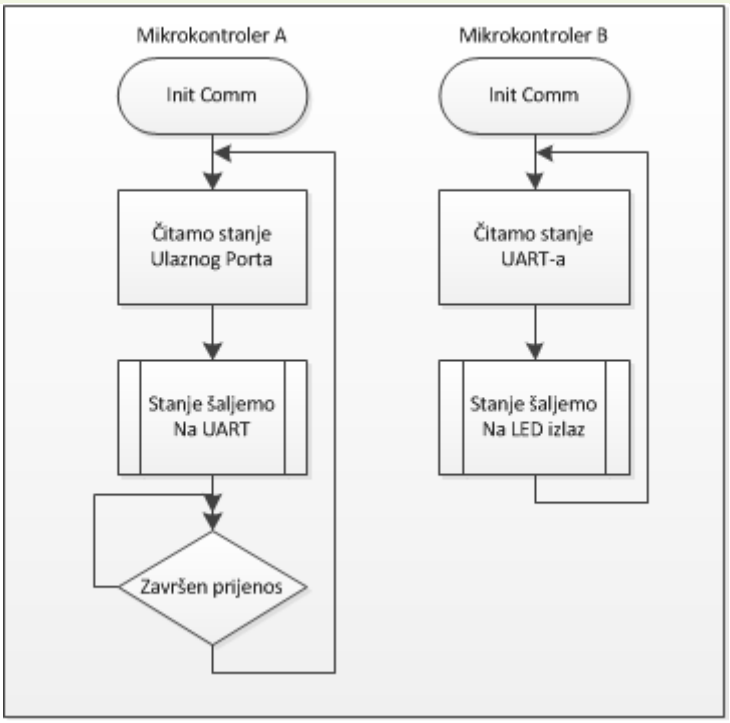
Za razliku od navedenih, kod korištenja RS485 (RS422) sučelja na jednoj komunikacijskoj liniji možemo imati više uređaja. Da bi spriječili istovremenu predaja dva uređaja obavezno je korištenje standardiziranih (npr.MODBUS) ili korisnički definiranih komunikacijskih protokola. Ukoliko želite da Vaš uređaj komunicira s uređajima (ili programima npr. SCADA) drugih proizvođača svakako koristite standardizirani protokol no ako to nije potrebno (ili ne želite) jednostavnije Vam je definirati vlastiti. Za standardizirane protokole upitajte gospodina Googla pa o njima neću pisati. Kod definiranja vlastitog protokola smo u prednosti jer ga možemo prilagoditi vlastitim potrebama i specifičnim zahtjevima. Kako definirati vlastiti komunikacijski protokol najjednostavnije ću prikazati sljedećim primjerima:

1. 8 bitova u jednom smjeru

Stanje 8 bitova (ulaz) kontrolera A treba prenijeti na kontroler B udaljen 100m te tamo upaliti LEDice (izlaz)

- s obzirom na udaljenost koristimo RS485 sučelje
- koristimo fiksirani smjer komunikacije (A=Predajnik B=Prijemnik)
- kako Ledice ne mogu napraviti nikakvu "opasnu" radnju nećemo kontrolirati ispravnost komunikacije
- format komunikacije je 1200,8,n,1 (1200 bitova u sekundi, 8 bitova podatka, bez pariteta, 1 stop bit)
- stanje ulaza će se prenositi na ledice oko 120x u sek. (1200Baud/10bit) (start + 8 data + 1 stop =10 bitova)

Program na oba kontrolera u inicijalnom dijelu definira parametre komunikacije.

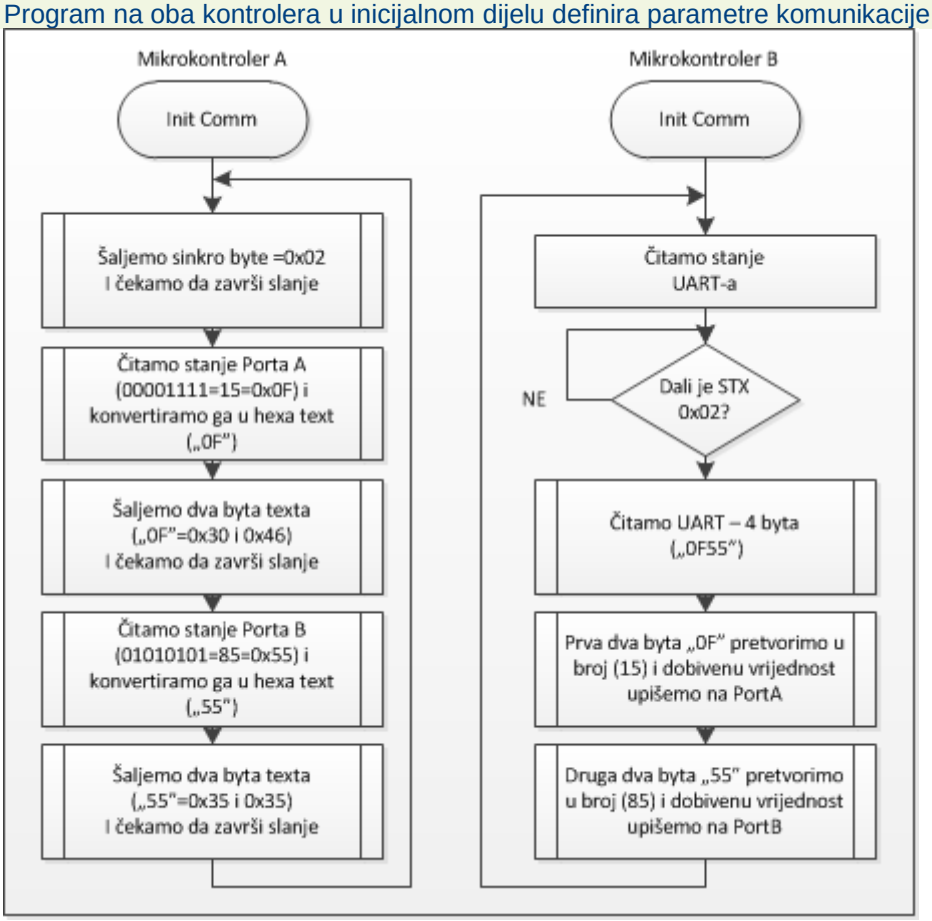


2. 16 bitova (2 byta) u jednom smjeru

Stanje 16 bitova (ulaz na dva porta) kontrolera A treba prenijeti na kontroler B udaljen 100m te tamo upaliti LEDice (Izlaz na ista dva porta)

- s obzirom na udaljenost koristimo RS485 sučelje
- koristimo fiksirani smjer komunikacije (A=Predajnik B=Prijemnik)
- kako Ledice ne mogu napraviti nikakvu "opasnu" radnju nećemo kontrolirati ispravnost komunikacije
- format komunikacije je 1200,8,n,1 (1200 bitova u sekundi, 8 bitova podatka, bez pariteta, 1 stop bit)
- logiku iz prethodnog primjera ne možemo primijeniti jer prijemnik ne zna koji byte je koji port.
- da bi to znali uvodimo dodatni sinkronizacijski byte na početku poruke STX koji ima vrijednost 02 dec
- stanje ulaznog porta može biti jednako sinkronizacijskom bajtu zbog toga trebamo
- stanja porta slati kao hexadecimalnu vrijednost (text) u dva byta 0="00" 2="02" 127="7F" 255="FF" ...
- da bi prenijeli stanje 2 byta trebamo slati 5 byta STX + 2 byta za portA + 2 byta za portB
- stanje ulaza će se prenositi na ledice oko 24x u sek. (1200Baud/10bit/5byta)

Program na oba kontrolera u inicijalnom dijelu definira parametre komunikacije.

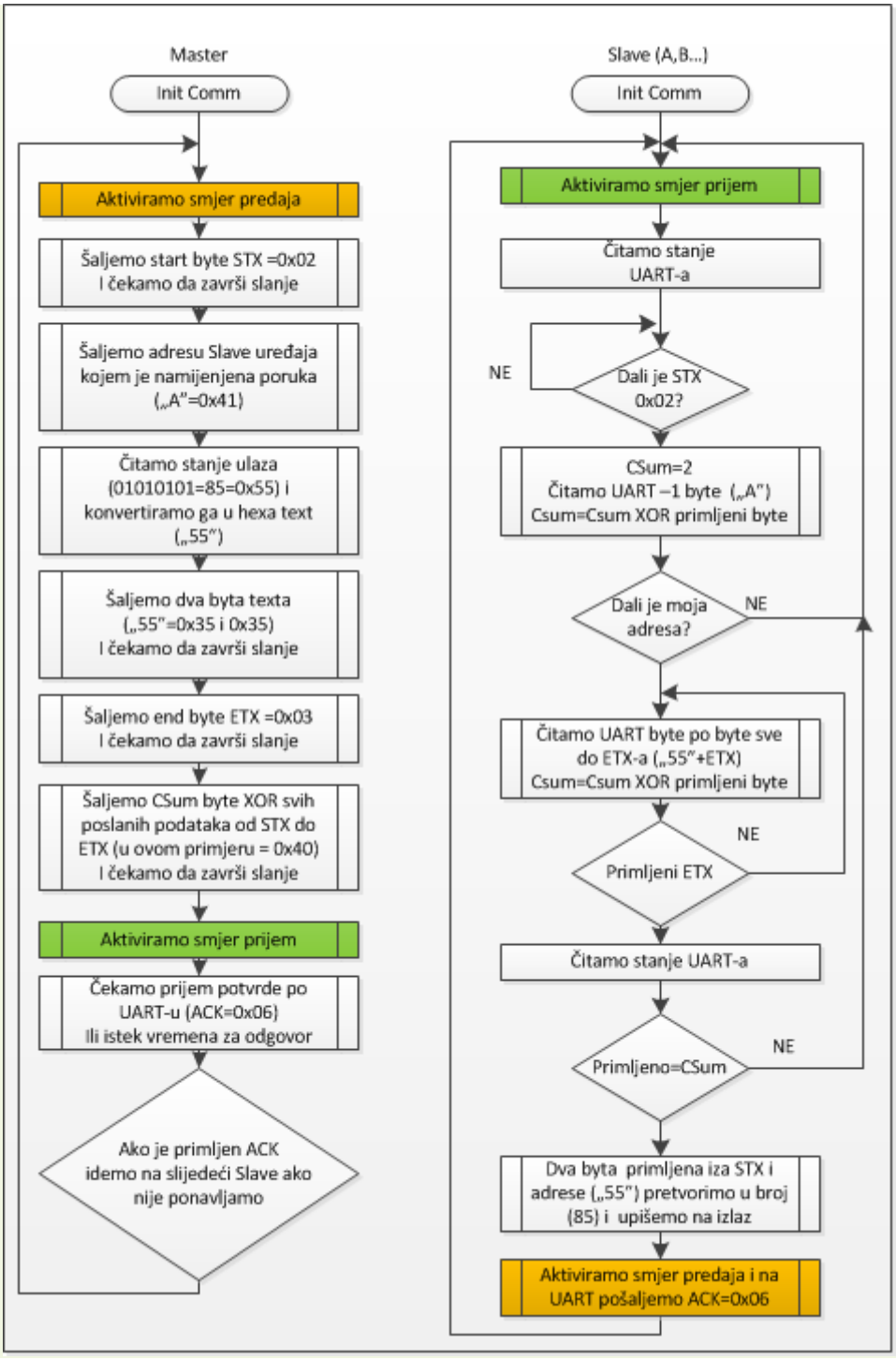


3. 8 bitova (1 byte) na više Slave uređaja

Stanje 8 bitova (ulaz) s master kontrolera treba prenijeti na slave kontrolere "A" i "B" ili više njih te tamo upaliti releje (Izlaz na jednom portu). Slave kontroleri trebaju poslati potvrdu izvršenja.

- s obzirom više Slave uređaja koristimo RS485 sučelje
- smjer komunikacije je potrebno kontrolirati dodatnim pinom na masteru i slave kontrolerima
- kako releji mogu napraviti "opasnu" radnju moramo kontrolirati ispravnost komunikacije
- format komunikacije je 1200,8,n,1 (1200 bitova u sekundi, 8 bitova podatka, bez pariteta, 1 stop bit)
- s obzirom na više slave-ova trebamo dodati adresni byte koji definira za koji slave se odnosi poruka mastera
- uvodimo dodatni sinkronizacijski byte na početku poruke STX koji ima vrijednost 02 dec
- stanje ulaznog porta može biti jednako kontrolnim bajtovima zbog toga trebamo
- stanja porta slati kao hexadecimalnu vrijednost (text) u dva byta 0="00" 2="02" 127="7F" 255="FF" ...
- iako bi se u ovom primjeru mogao izbjeći uvodimo i završni byte na kraju poruke ETX (0x03)
- iza ETX-a šaljemo kontrolni bajt koji je XOR svih poslanih bajtova uključujući STX i ETX

Program na oba kontrolera u inicijalnom dijelu definira parametre komunikacije.



Navedeni primjer je osnova "pravog" komunikacijskog protokola. Naravno da svaki korisnički definirani protokol može značajno odstupati od ovog primjera. U nastavku bih želio još samo dodati nekoliko napomena i smjernica (iskustava) koji će Vam pomoći da ne krenete u krivom smjeru odnosno lakše uočite moguće probleme.

RS 485 SOFTWARE

Pri definiranju komunikacijskog protokola probajte dobro definirati natuknice iz prethodnih primjera. Vrlo je zgodno koristiti istu strukturu poruke na master i slave uređajima. U primjeru 3 to nije poštivano već je korišteno samo slanje potvrdnog odgovora na ispravno primljenu poruku (ACK). Ovu potvrdu koristimo da bi master bio siguran da je komanda izvršena ali i da se utvrdi ispravnost rada slave uređaja. Mogli smo koristiti i negativan odgovor NAK (0x15) na neispravno primljenu poruku no to izbjegavajte ukoliko imate više slave uređaja. Razlog za to je mogućnost da neki slave uređaj neispravno primi poruku upućenu na drugom slave uređaju (a kod neispravne poruke nismo sigurni kome je ona upućena) pa ni se moglo desiti da istovremeno jedan uređaj odgovara s ACK a drugi s NAK. Važno je za primijetiti da svaki uređaj u prijemu "čuje" svu komunikaciju koja se odvija na liniji bez obzira dali se odnosi na njega ili ne. Zbog toga Vam predlažem sljedeću konfiguraciju poruka:
(svaki byte poruke je upisan unutar znakova <> ; poruka je niz između <STX> i <ETX>)
<STX> <Adresa> <Komanda> <Artibut1> <Atribut2> ... <ETX> <CSum>

<STX>
označava početak poruke i svaki uređaj sve što je primio do ovog znaka zanemaruje i ne upisuje u prijemni buffer i tek nakon prijema ovog znaka razmatra prijemne podatke.<STX> znak ne smije biti primljen unutar poruke sve do <ETX> znaka a ukoliko je primljen smatramo da je tu početak poruke. Valja napomenuti da je <STX> moguće dobiti na poziciji CSum byta nakon kraja poruke.

<Adresa>
je sljedeći byte poruke i označava kome je poruka namijenjena (kod mastera) ili tko šalje poruku ako se radi o predaji slave uređaja. Adresu je najbolje podesiti tako da predstavlja niz ("A"=prvi slave , "B"=drugi slave ... ili "1" za prvi slave, "2" za drugi ...). Kao adresu a i ostale dijelove poruke (komandu i atribute) ne smijemo koristite <STX> ili <ETX> znakove. Temeljem ovog byta master zna tko je poslao odgovor a slave koji ima ovu adresu nastavlja razmatranje poruke dok svi ostali zanemaruju sve primljeno do sljedeće poruke. Podešavanje adrese na slave uređajima najzgodnije je uraditi DIP prekidačima.

<Komanda>
ovaj byte označava što master želi od adresiranog slave uređaja. Npr. "R" (Read) bi moglo biti čitanje stanja ulaza slave-a; "W" (Write) upis podatka na izlaze slave-a i sl. U primjeru 3 komanda nije korištena jer slave i onako njie mogao uraditi ništa drugo osim postaviti izlaz prema primljenim podacima. Ja u praksi najčešće koristim "slovne" komande kao u primjeru s tom razlikom da master šalje velika slova a slave u odgovoru šalje isto malo slovo.

<Artibut1> <Atribut2>
naredni niz byte-ova ovisan je komandi i našim željama i potrebama. Primjer: imamo slave A koji očitava temperaturu i vlagu trenutno (24°C i 78%). Na poruku mastera "AR" slave odgovara porukom "Ar2478". Isto bi mogli odraditi i na način da za očitanje temperature master šalje "AT" a slave odgovara s "At24" odnosno "AV" za očitanje vlage na što dobijemo odgovor "Av78"

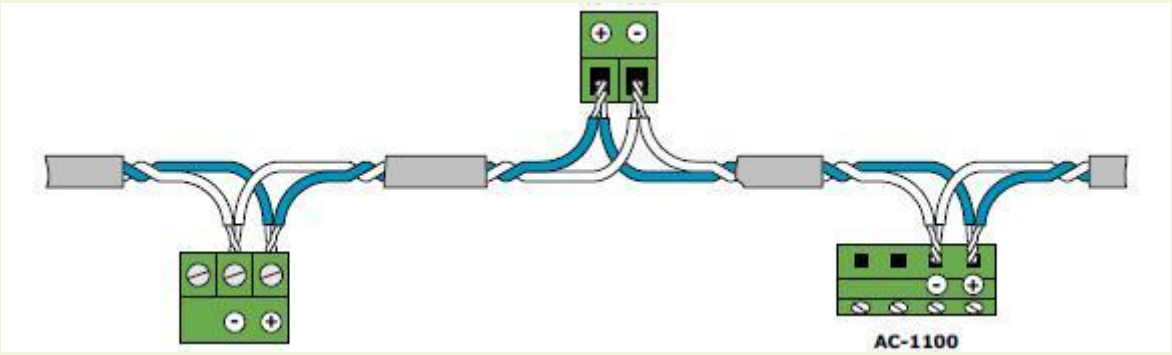
<ETX>
na kraju svake poruke šaljemo ovaj znak kako bi znali da je tu kraj poruke i da **iza ovog byta sljedi** byte kontrolnog zbroja CSum.

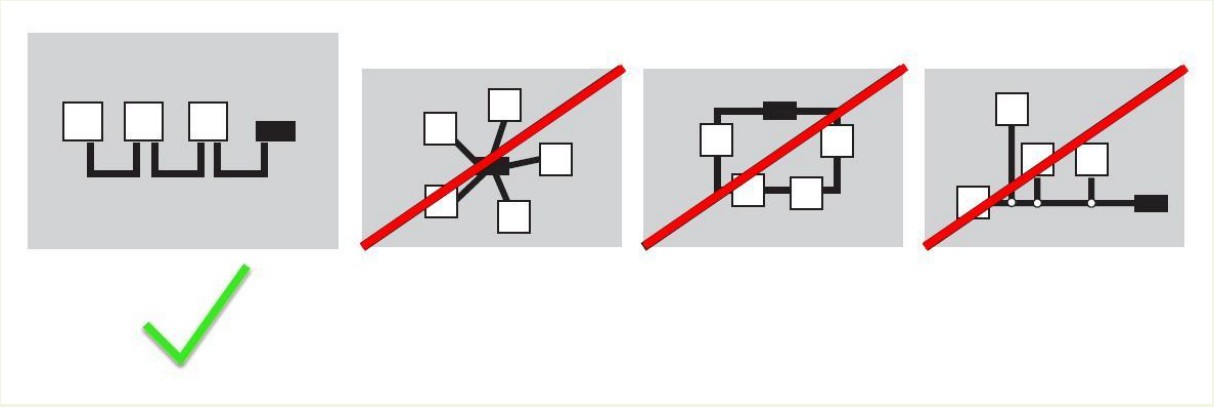
<CSum>
kontrolni zbroj je byte koji se najčešće računa kao zbroj na razini byta svih primljenih byte-ova ili XOR operacija istih. Kako sam već spomenuo ovaj byte može imati i vrijednost kontrolnih byte-ova <STX> i <ETX>.

RS485 HARDWARE

Kao što je već rečeno komunikacija po RS485 je vrlo pouzdana u industrijskom okruženju no da bi smanjili utjecaj smetnji ne trebamo pretjerivati s brzinama komunikacije ako to nije nužno potrebno. Npr. nema smisla očitavati senzor temperature 100 puta u sekundi jer se ona ne mijenja tako brzo.

Za međusobno povezivanje uređaja obavezno koristite upletenu paricu. Danas je to najčešće jedan par iz UTP kabela. Najbolji način za povezivanje je u nizu bez odvojaka kao što je prikazano na donjoj slici.





Topologija u obliku zvijezde, prstena ili stabla **nije preporučena.** Na prvom i posljednjem uređaju u nizu potrebno je **postaviti završene otpore** (120 oma) radi sprječavanja refleksija na kabelu. Ponekad se osim osnovne parice uvodi i treći vodič (obično na dugim linijama) kao referentni vodič - masa.
Moja preporuka za uređaje koji komuniciraju po RS485 (naročito kod dugih linija) je da budu izolirani od uzemljenja. Isto tako, ako koristite PC računalo kao master, svakako ga povezujte na komunikacijsku liniju preko galvanski izoliranog adaptera jer kod atmosferskih pražnjenja (gromova) vole stradati.