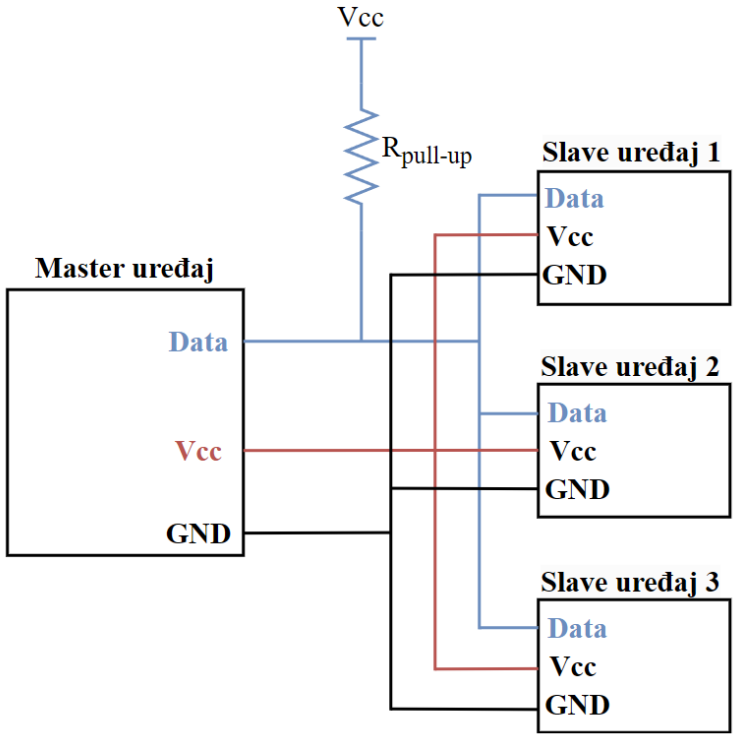
1-wire protokol

1 1-wire protokol

Jedna od prednost serijske u odnosu na paralelnu komunikaciju svakako jeste zahtev za znatno manjim brojem linija za prenos podataka. Komunikacioni protokoli serijske komunikacije do sada obrađeni na vežbama zahtevali su određeni broj linija za komunikaciju:

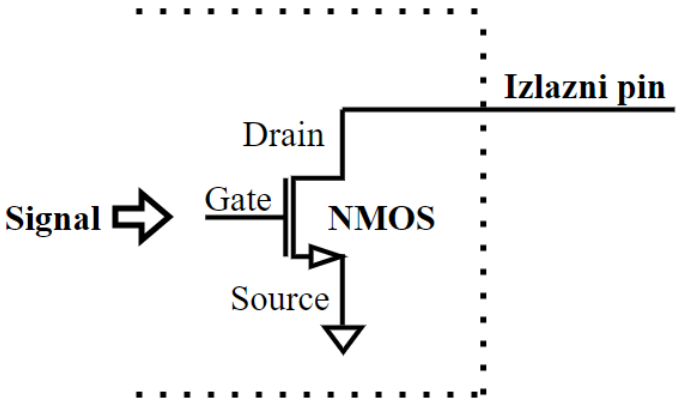
- UART - Tx, Rx
- I2C - SDA, SCL
- SPI - SCK, MISO, MOSI, CS

One-wire protokol predstavlja protokol serijske komunikacije koji koristi jednu liniju za prenos podataka, i dodatne linije za napajanje. Ovaj protokol razvijen je u kompaniji *Dallas Semiconductor* (danas u vlasništvu kompanije *Maxim Integrated*). Kod 1-wire komunikacije vrši se razmena podataka između jednog master uređaja i jednog ili više slave uređaja. Kako se razmena podataka vrši putem jedne linije, a moguća je bidirekciona razmena podataka, reč je o half-duplex komunikaciji. Dodatno, potrebne su i linije za napon napajanja uređaja (Vcc) i masu (GND). Povezivanje master i slave uređaja za 1-wire komunikaciju prikazano je na slici 1. U ovakvim sistemima postoji mogućnost napajanja slave uređaja putem linije za podatke, što obezbeđuje još manji broj linija neophodnih za komunikaciju uz određena podešavanja pri povezivanju.



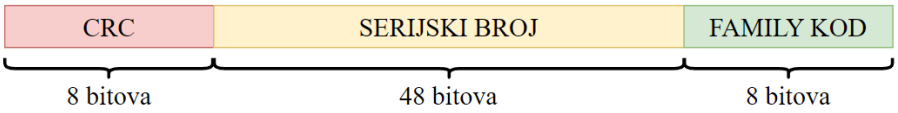
Slika 1: Povezivanje master i slave uređaja za one-wire komunikaciju

Linija za prenos podataka realizuje se u open-drain konfiguraciji (slika 2). Open-drain povezivanje podrazumeva upotrebu jednog MOSFET-a na čiji *gate* se dovodi signal kojim se upravlja stanjem na *drain*-u koji je izvučen kao izlazni pin. Ovakav način povezivanja zahteva upotrebu *pull-up* otpornika koji definišu stanje logičke 1, kada je na pinu stanje visoke impedanse.



Slika 2: Open-drain povezivanje

Uređaji koji imaju mogućnost komunikacije putem 1-wire protokola u okviru ROM memorije imaju upisan jedinstveni serijski broj (6 bajtova), family kod (1 bajt), kao i odgovarajući CRC kod (1 bajt). CRC kod predstavlja metodu provere pojave greške u podacima koji se prenose. Kako je dužina ROM memorije 64 bita, postoji 2^{64} jedinstvenih kombinacija pri adresiranju slave uređaja.



Slika 3: Sadržaj ROM memorije 1-wire slave uređaja

1.1 CRC (*Cyclic Redundancy Check*)

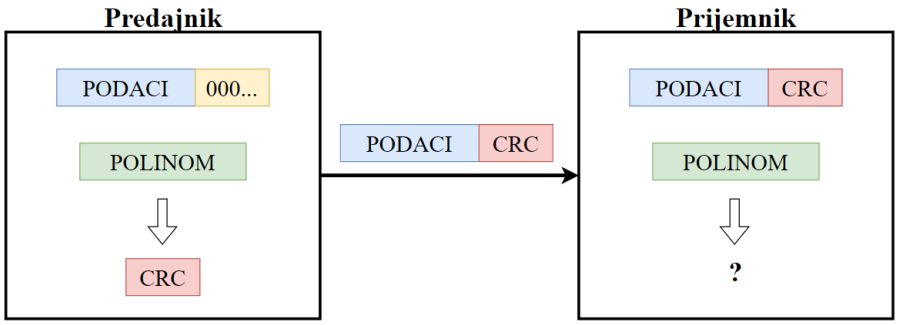
CRC (*Cyclic Redundancy Check*) predstavlja jednu od metoda za detekciju pojave greške u podacima koji se prenose po određenom komunikacionom protokolu. Metode detekcije pojave greške kao što su CRC, parity ili checksum zasnivaju se na izračunavanju koda za proveru na osnovu određenog algoritma u okviru predajnika, slanju koda za proveru uz odgovarajući set podataka, proveri primljenih podataka na prijemnoj strani, po istom algoritmu i na osnovu primljenog koda za proveru. Provera pojave greške na osnovu CRC koda zasniva se na binarnom deljenju po modulu 2 bitova podataka i određenog polinoma oblika $p(x) = p_R \cdot x^R + ... + p_1 \cdot x^1 + p_0$.

Ovaj metod provere pojave greške odvija se u nekoliko koraka:

- Na predajnoj strani podaci se proširuju sa R nula, gde je R red polinoma.

- Podaci prošireni nulama, dele se po modulu 2 (XOR) sa definisanim polinomom, a ostatak pri deljenju predstavlja CRC kod
- CRC kod se dodaje na podatke i šalje na prijemni uređaj
- Na prijemnoj strani se podaci dele po modulu 2 (XOR) sa CRC kodom i posmatra se ostatak pr deljenju
- Ukoliko je ostatak pri deljenju po modulu 2 nula nije došlo do greške pri prenosu podataka

Na slici 3 ilustrovan je blok dijagram sistema prijemnika i predajnika sa proverom pojave greške u podacima pomoću CRC koda.



Slika 4: Blok dijagram provere pojave greške na osnovu CRC koda

U zavisnosti od oblasti primene računa CRC koda i algoritama koji se primenjuju definišu se različiti polinomi, a neki od njih su:

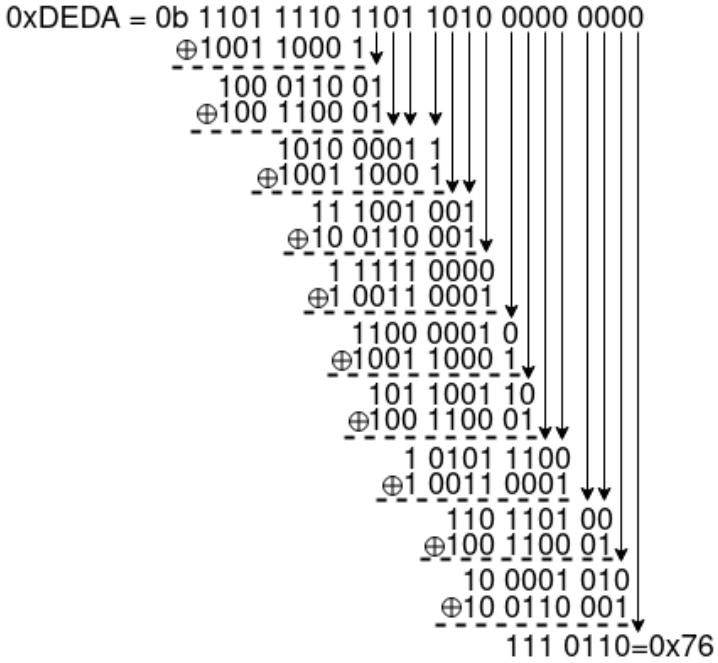
- CRC-4-ITU: $x^4 + x + 1$ (0b10011)
- CRC-5-USB: $x^5 + x^2 + 1$ (0b100101)
- CRC-6-GSM: $x^6 + x^5 + x^3 + x^2 + x + 1$ (0b1101111)
- CRC-8-Dallas/Maxim: $x^8 + x^5 + x^4 + 1$ (0b100110001)
- CRC-8-Bluetooth: $x^8 + x^7 + x^5 + x^2 + x + 1$ (0b110100111)

Kod 1-wire komunikacionog protokola koristi se CRC-8-Dallas/Maxim polinom $x^8 + x^5 + x^4 + 1$, koji odgovara binarnom zapisu 0b100110001.

Primer: Računanje CRC koda za podatke 0xDEDA i polinom $x^8 + x^5 + x^4 + 1$

Neka su podaci koje je potrebno poslati 0xDEDA, tj. 0b1101 1110 1101 1010. Polinom oblika $x^8+x^5+x^4+1$ odgovara binaronom zapisu 0b100110001. Kako je polinom 8. reda, u prvom koraku neophodno je na podatke dodati osam nula, odnosno 0x00, te se dobije binarni zapis 0b1101 1110 1101 1010 0000 0000. U sledećem koraku vrši se deljenje dobijenog zapisa po modulu 2, sa polinomom, na način prikazan na slici 5. Dakle, u svakom koraku vrši se logička eks-ili (XOR) funkcija polinoma 0b100110001 i devet bitova podataka sa pridodatim nulama na kraj. U ovom

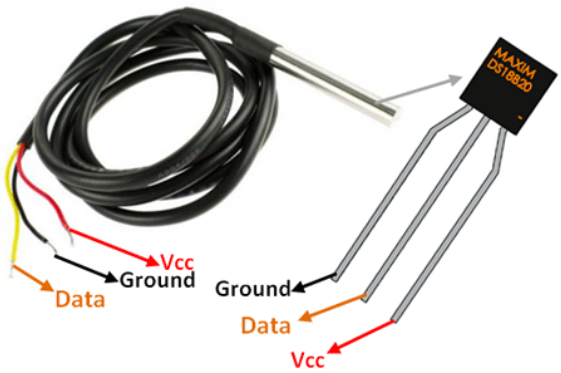
postupku dobija se CRC kod 0b0111 0110, tj. 0x76. Ovaj CRC kod potrebno je, umesto nula iz prvog koraka, dodati na podatke 0xDEDA, a zatim se takvi podaci šalju na prijemni uređaj. Sa prijemne strane vrši se provera pojave greške tako što se primljeni podaci na koje je dodat CRC kod binarno dele po modulu 2 sa polinomom. Ukoliko je ostatak pri deljenju nula nije došlo do greške, dok ostatak pri deljenju različit od nule ukazuje na pojavu greške pri prenosu podataka.



Slika 5: Primer računanja CRC koda

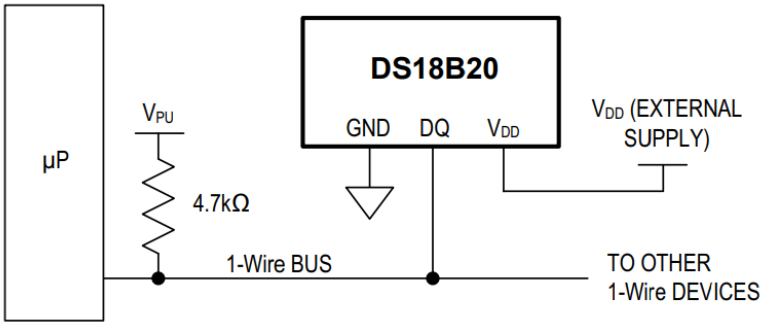
2 DS18B20 senzor

DS18B20 (slika 6) predstavlja digitalni temperaturni senzor, programabilne rezolucije od 9 do 12 bitova. Merni opseg datog senzora je od -55 °C do 125 °C. Tačnost senora je 0,5 °C na opsegu od -10 °C do 85 °C.



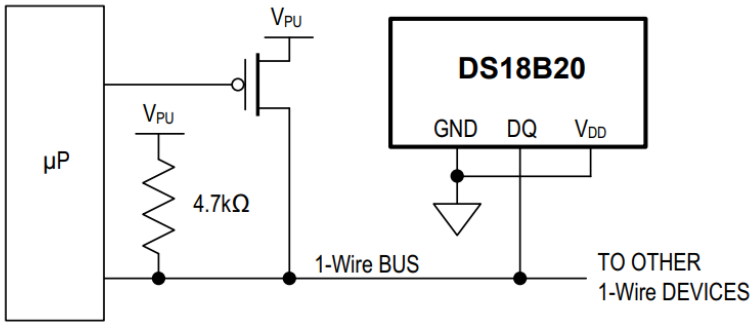
Slika 6: DS18B20 senzor

DS18B20 ima mogućnost komunikacije putem 1-wire komunikacionog protokola, za šta koriste linije V_{cc}/V_{DD}, Ground/GND i Data/DQ (slika 6). Postoje dva načina napajanja ovakvih senzora - putem *eksternog napajanja* koje se dovodi na V_{DD} liniju ili u takozvanom *parasite power* modu. Na slici 7 prikazano je povezivanje DS18B20 senzora sa mikrokontrolerom (master uređajem) za 1-wire komunikaciju, kada se DS18B20 senzor napaja putem V_{DD} linije. U ovom slučaju razmena podataka između master i slave uređaja vrši se putem 1-wire magistrale tj. linije za podatke koja je povezana u open-drain konfiguraciji uz upotrebu pull-up otpornika.

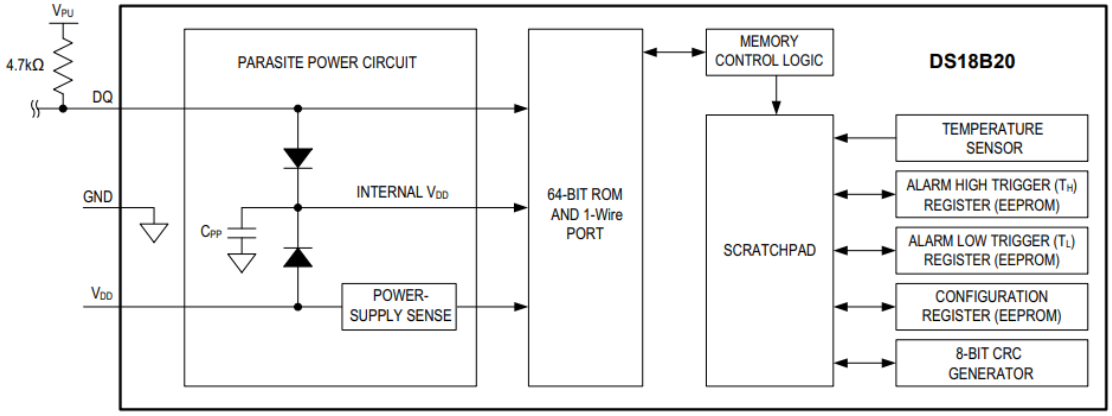


Slika 7: Povezivanje DS18B20 senzora u *external power supply* modu

Parasite power mod napajanja senzora podrazumeva napajanje putem linije za prenos podataka. Na slici 7 prikazano je povezivanje DS18B20 senzora sa mikrokontrolerom (master uređajem) za 1-wire komunikaciju, kada se DS18B20 senzor napaja putem komunikacione linije. U datoj postavci, pinovi GND i V_{DD} povezani su na masu. Kako bi senzor bio napajan kroz liniju za prenos podataka, neophodno je da njeno stanje bude logička 1, tj. visok naponski nivo. Dodatno, u okviru DS18B20 postoji kondenzator kapacitivnosti Cpp - slika 9 koji se za vreme stanja logičke 1 na liniji za podatke puni, a kada je stanje na liniji logička 0, omogućava napajanje senzora. Kako je maksimalna struja koja može proteći kroz DQ pin DS18B20 senzora pri vrešenju određenih operacija 1,5 mA, pad napona koji se u toj situaciji ostvaruje na pull-up otporniku od 4,7 kΩ iznosi oko 7,05 V. Ovo iziskuje upotrebu dodatnog MOSFET-a za upravljanje linijom za prenos podataka.

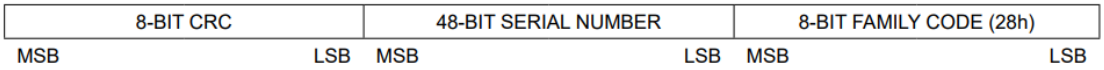


Slika 8: Povezivanje DS18B20 senzora u *parasite power supply* modu



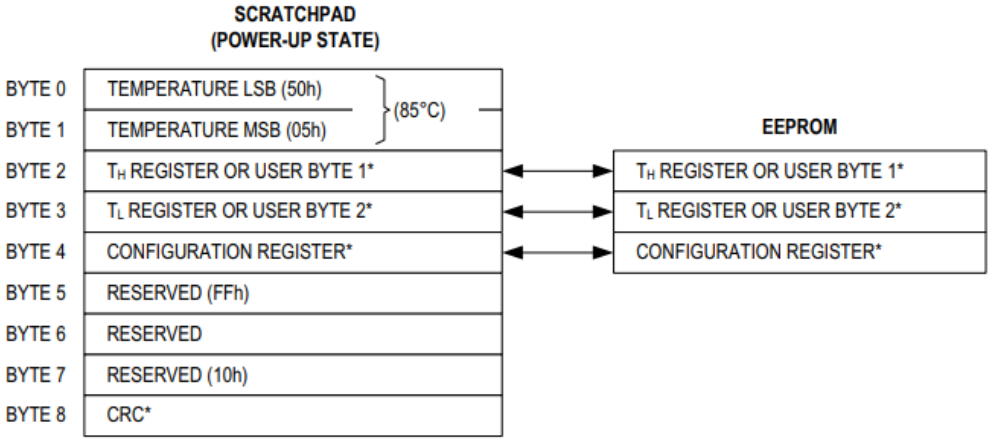
Slika 9: Blok dijagram DS18B20 senzora

Svaki DS18B20 senzor sadrži jedinstveni 64-bitni kod upisan u ROM memoriju (slika 11). Donjih 8 bitova ROM memorije čine 1-wire family kod, koji je u slučaju DS18B20 senzora 0x28. Narednih 6 bajtova ROM memorije predstavljaju serijski broj senzora, dok gornjih 8 bitova čine CRC kod koji se računa u odnosu na preostalih 56 bitova ROM memorije.



Slika 10: Sadržaj ROM memorije DS18B20 senzora

Organizacije SRAM memorije i odgovarajuće EEPROM memorije DS18B20 prikazana je na slici 11.



Slika 11: Sadržaj scratchpad memorije DS18B20 senzora

Nulti i prvi bajt takozvane *scratchpad* memorije predstavljaju niži i viši bajt podataka o izmerenoj temperaturi, i ovi registri se mogu samo očitavati (*read-only*). Podaci o temperaturi čuvaju se u fixed point formatu. DS18B20 senzori imaju mogućnost zadavanja gornje i donje granične temperature i poređenja izmerene temperature sa zadatim granicama. Ukoliko je izmerena temperatura manja od donje granice ili veća od gornje granice setuje se odgovarajući alarm flag. Granične temperature moguće je zadavati kroz registre TH i TL, odnosno, bajtove 2 i 3 scratchpad memorije. Nakon TH i TL registara sledi konfiguracioni registar, kroz koji je moguće podešavati rezoluciju senzora. Podešavanjem bitova R1R0 konfiguracionog registra (slika 12) na vrednosti 00, 01, 10 ili 11 ostvaruju se rezolucije od 9, 10, 11 ili 12 bitova, redom. U registre TH, TL i konfiguracioni registar moguće je upisivati vrednost, kao i očitavati njihov sadržaj. Takođe, sadržaj ova tri registra moguće je upisati u EEPROM ili je sadržaj EEPROM-a moguće upisati u ova tri registra.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Slika 12: Sadržaj konfiguracionog registra DS18B20 senzora

Peti, šesti i sedmi bajt scratchpad memorije su rezervisani za internu upotrebu i nije ih moguće koristit. Osmi bajt namenjen je CRC kod, koji se računa u odnosu bajtove od 0 do 7.

Podešavanje DS18B20 senzora, kao i razmena podataka vrše se prema 1-wire protokolu putem određenog seta komandi. Pri svakoj komunikaciji sa DS18B20 senzorom neophodna su tri koraka: inicijalizacija, slanje ROM komande, slanje funkcijskih komandi. Inicijalizacija komunikacije omogućava detekciju prisustva slave uređaja, a njen opis, kao i opis samog protokola dati su u okviru narednog poglavlja. Nakon detekcije prisustva slave uređaja, master uređaj šalje ROM komande, pomoću kojih master uređaj može da adresira određeni slave uređaj, detektuje broj i tip uređaja sa kojima ima mogućnost komunikacije, ili da dobije informaciju da li je neki od uređaja detektovao alarm flag. Pet ROM komandi za DS18B20 senzor dato je u tabeli 1.

Komanda	**Kod**	**Opis**
Search ROM	0xF0	Identifikacija ROM kodova svih slave uređaja
Read ROM	0x33	Očitavanje 64-bitnog ROM koda (koristi se kada je prisutan samo jedan slave uređaj)
Match ROM	0x55	Ova komanda praćena 64-bitnim ROM kodom omogućava adresiranje određenog slave uređaja
Skip ROM	0xCC	Adresiranje svih slave uređaja povezanih na master
Alarm Search	0xEC	Detekcija onih slave uređaja koji imaju setovan alarm flag

Tabela 1: ROM komande DS18B20 senzora

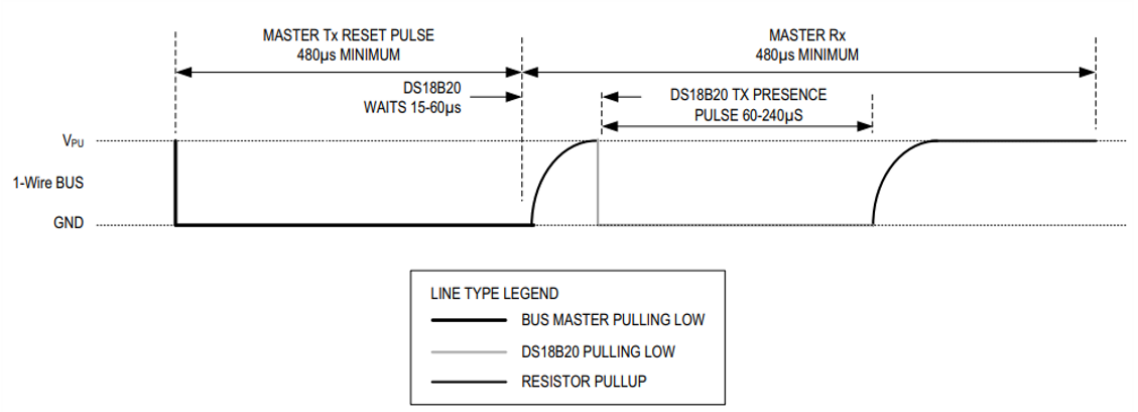
Nakon slanja ROM komandi, master uređaj može da pošalje neku od šest funkcijskih komandi, pomoću kojih ima mogućnost očitavanja sadržaja scratcpad memorije, upisa u scratchpad memoriju i sl. Funkcijske komande za DS18B20 sentor date su u tabeli 2.

Komanda	**Kod**	**Opis**
Convert T	0x44	Konverzija temperature i smeštanje rezultata u dva temperaturna registra scratchpad memorije
Write Scratchpad	0x4E	Upis tri bajt podataka u scrtatchpad memoriju - prvi bajt upisuje se u TH registar, drugi bajt upisuje se u TL registar, dok se treći bajt podataka upisuje u konfiguracioni registar
Read Scratchpad	0xBE	Očitavanje sadržaja scratchpad memorije
Copy Scratchpad	0x48	Upis sadržaja TH, TL i konfiguracionog registra u EEPROM
Recall E^2	0xB8	Upis sadržaja iz EEPROM memorije u TH, TL i konfiguracioni registar
Read Power Supply	0xB4	Određivanje tipa napajanja senzora - nakon ove komande senzor koji se napaja parazitski sa linije za podatke obara stanje na liniji na logičku 0, dok senzor koji se napaja putem eksternog napajanja drži liniju na logičkoj 1

Tabela 2: Funkcijske komande DS18B20 senzora

3 Implementacija 1-wire protokola

Na slici 13 prikazan je način inicijalizacije komunikacije tj. generisanja reset stanja na 1-wire mreže.



Slika 13: One-wire: početak komunikacije

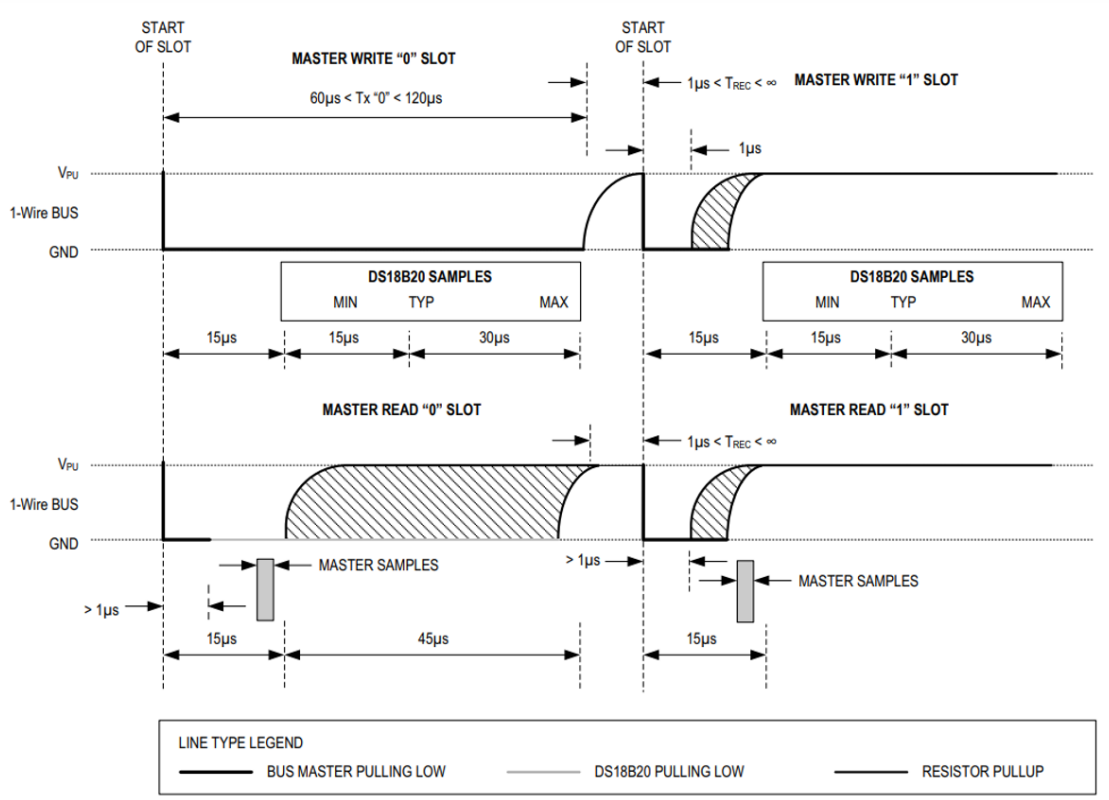
Master uređaj generiše reset signal povlačenjem komunikacione linije 480 μs na nizak naponski nivo, nakon čega otpušta liniju. Otpuštanje linije se odvija prebacivanjem pina u stanje visoke impedanse (ulazni pin), nakon čega će stanje na pinu biti definisano na pull-up otporniku. Nakon pauze između 15 μs i 60 μs periferni uređaj, u ovom slučaju DS18B20, povlači komunikacionu liniju na nizak nivo, gde će impuls trajati od 60 μs do 240 μs. Potom periferni uređaj postavlja svoj pin u stanje visoke impedanse, pa će pull-up otpornik ponovo definisati stanje na pinu. Implementacija reset sekvence data je listingom 1. Važno je istaći da funkcija *one_wire_reset* vraća *bool* čime se definiše prisustvo senzora (*false* - senzor je povukao liniju na nizak nivo - senzor je prisutan, *true* - pull-up otpornik definiše stanje na pinu - senzor nije prisutan ili se ne odaziva). Provera prisustva se nakon 480 μs inicijalne pauze generiše nakon 70 μs čime je dato dovoljno vremena da u slučaju da je pauza do odziva 60 μs senzor povuče liniju na nizak nivo. Kako bi se ubrzala tranzicija sa logičke nule na jedinicu, pre nego što je pin prebačen u stanje visoke impedanse upisano je visoko stanje na pinu. Takođe, kako se pri redefinisanju smera toka signala na pinu kao izlaznog vrednost, ako je u LAT registru upisana vrednost 0 pin će liniju povući na nizak nivo. Pošto je u stanju pripravnosti 1-wire linija podataka na visokom nivou, to je dodatan razlog da se ovde pin podigne na visok nivo pre promene da je pin ulazni. Funkcijom *ow_GetValue* očitava se vrednost koja se nalazi na pinu i ako je ona 0 senzor je prisutan, a ako je 1 senzor se ne daje odgovor. Nakon pauze od 410 μs kompletira se vremenski slot predviđen za generisanje reset signala.

Listing 1: Funkcija za inicijalziaciju 1-wire komunikacije

```
1 bool one_wire_reset(void)
2 {
3     bool device_detected;
4     ow_SetDigitalInput();
5     while(!ow_GetValue());
```

```
6
7       ow_SetDigitalOutput();
8       ow_SetLow();
9
10      __delay_us(480);
11
12      ow_SetHigh();
13      ow_SetDigitalInput();
14
15      __delay_us(70);
16
17      device_detected = ow_GetValue();
18
19      __delay_us(410);
20
21      return device_detected;
22  }
```

Na slici 14 prikazan je vremenski dijagram upisa i očitavanja logičkih stanja na pin i sa pina DS18B20 temperaturnog senzora.



Slika 14: One-wire: read i write slotovi

U slučaju slanja logičke nule master povlači liniju na logičku nulu a zatim narednih od 60 μs do 120 μs liniju zadržava na tom nivou. Spram korisničkog uputstva senzor će očitati stanje na pinu između 15 μs i 60 μs. Po odzivu senzora master uređaj mora otpustiti liniju kako bi je pull-up otpornik postavio u stanje logičke jedinice. U slučaju da se šalje logička jedinica linija se kratkim impulsom dužine oko 1 μs spušta na nizak nivo, a zatim pin master uređaja postaje ulazni. Senzor će odabirati signal ponovo između 15 μs i 60 μs od početka generisanja logičkog stanja.

U listingu koda 2 data je implementacija funkcije koja upisuje jedan bit. Funkciji *one_wire_write_bit* kao parametar se prosleđuje bit koji se želi upisati. U slučaju da je korisnik porsledio jedinicu, pin se spušta na nizak naponski nivo 1 μs, a zatim se linija otpušta kako bi pull-up otpornik podesio na liniji visok naponski nivo. Pin ostaje u tom stanju narednih 59 μs kako bi se ispoštovali tajminzi specificirani od strane proizvođača. Takođe, u slučaju da je prosleđeni bit nula korisnik liniju postavlja na nizak naponski nivo i zadržava ga narednih 60 μs. Kako bi se kompletirao vremenski slot između upisa dva uzastopna bita neophodno je napraviti pauzu od minimalno 1 μs, a u ovom slučaju pauza iznosi 10 μs.

Listing 2: Funkcija za upis jednog bita podataka

```
1  static void one_wire_write_bit(uint8_t b)
2  {
3      if(b)
4      {
5          ow_SetDigitalOutput();
6          ow_SetLow();
7          __delay_us(1);
8          ow_SetDigitalInput();
9          __delay_us(59);
10     }
11     else
12     {
13         ow_SetDigitalOutput();
14         ow_SetLow();
15         __delay_us(60);
16         ow_SetDigitalInput();
17     }
18     __delay_us(10);
19 }
```

Kada se želi vršiti očitavanje podataka na 1-wire liniji, korisnik mora zahtevati od senzora podatak povlačenjem linije na logičku nulu u trajanju oko 1 μs, nakon čega master odlazi u stanje visoke impedanse. Po prelasku mastera u stanje visoke impedanse senzor ili nastavlja da drži liniju na niskom nivou (senzor šalje logičku nulu) ili linija odlazi na visok nivo (senzor šalje logičku jedinicu). U uputsvu je preporuka da se odabiranje stanja vrši oko 15 μs od početka vremenskog slota.

U listingu koda 3 data je implementacija funkcije koja očitava jedan bit. Funkcija *one_wire_read_bit* vraća vrednost očitanog bita i to povlačeći liniju 2 μs na nizak naponski nivo, a zatim otpušta liniju i nakon 13 μs odabira digitalnu vrednost sa komunikacione linije. Vremenski slot se kompletira pauzom od 55 μs i kao povratna vrednost se vraća očitana vrednost.

Listing 3: Funkcija za očitavanje jednog bita podataka

```
1  static uint8_t one_wire_read_bit(void)
2  {
3      uint8_t b;
4
5      ow_SetDigitalOutput();
6      ow_SetLow();
7      __delay_us(2);
8      ow_SetDigitalInput();
9      __delay_us(13);
10     b = ow_GetValue();
11     __delay_us(45);
12     __delay_us(10);
13     return b;
14 }
```

Pošto korisnik 1-wire linijom razmenjuje podatke opisanim funkcijama bit po bit, neophodno je realizovati i funkcije koje šalju i primaju podatke bajt po bajt. Implementacija funkcije *one_wire_read* i *one_wire_write* date su u listinzima koda 4 i 5, redom. Funkcija za očitavanje proverava da li je primljeni bit jedinica i ako jeste postavlja ga na poziciju MSB-a, nakon čega se podaci pomeraju za jednu poziciju u desno. Pošto podaci pristižu počevši od LSB-a, nakon osam upisa i pomeranja u primljenom bajtu *result* nalaziće se osam primljenih bita. U slučaju upisa uvek se šalje LSB nakon čega se vrednost u celom registru pomera za jedno mesto u desno. Posle osam uzastopnih pomeranja MSB bit će biti poslat.

Listing 4: Funkcija za očitavanje jednog bajta podataka

```
uint8_t one_wire_read(void)
{
    uint8_t result = 0;
    for(uint8_t cnt = 0; cnt<8; cnt++)
    {
        result>>=1;
        if(one_wire_read_bit())
            result |= 0x80;
    }
    return result;
}
```

Listing 5: Funkcija za upis jednog bajta podataka

```
void one_wire_write(uint8_t write_data)
{
    for(uint8_t cnt = 0; cnt<8; cnt++)
    {
        one_wire_write_bit(write_data & 0x01);
        write_data >>= 1;
    }
}
```