



Vhodno-izhodne naprave (VIN)

Predavanja

4. Serijski prenos podatkov

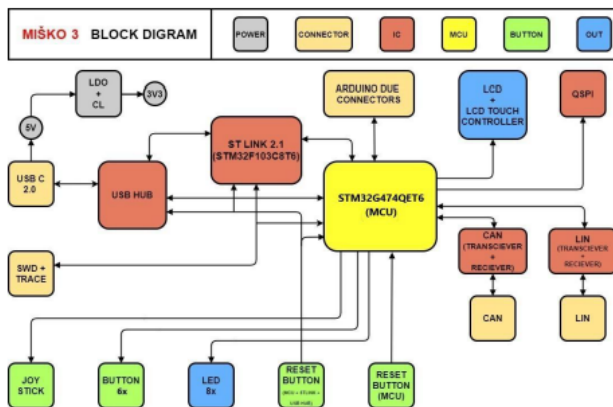
Robert Rozman

rozman@fri.uni-lj.si

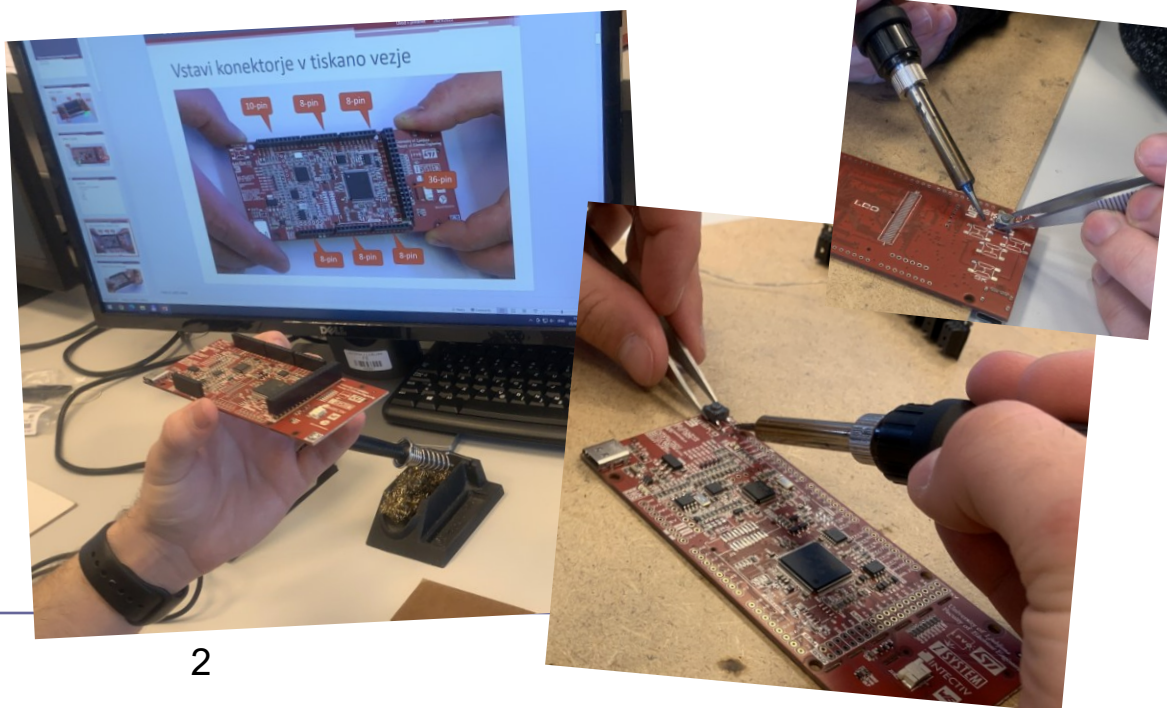
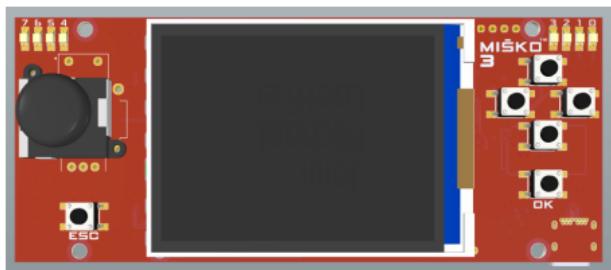
Miško 3

Tehnične specifikacije

- Mikrokontrolnik STM32G474QET
 - 96kB RAM
 - 512 kB Flash pomnilnika
 - napajanje 3,3 V
 - maksimalna frekvenca ure 170 MHz
- ST-LINK V2.1 za programiranje, razhroščevanje in UART komunikacijo s mikrokontrolnikom
- USB-C s podporo USB 2.0 za USB komunikacijo z mikrokontrolnikom in napajanje sistema
- USB HUB za povezavo na ST-LINK in USB vmesnik mikrokontrolnika
- Barvni LCD zaslon 320x240 s ILI9341 krmilnikom in 16-bitnim paralelnim vodilom prek FMC vmesnika
- Rezistivna folija za detekcijo dotika s XPT2046 krmilnikom na SPI vodilu
- 8 LED in 6 tipk, 2D joystick s tipko
- CAN in LIN vmesnik
- QSPI za razširitev pomnilnika
- SWD trace vmesnik za napredno razhroščevanje
- Arduino DUE kompatibilna dimenzija tiskanega vezja, razporeditev letvic in signalov



Slika 1: Bločni diagram razvojnega Sistema Miško 3



DN1: Domača naloga – Primeri tem

OneNote zvezek – dosedANJI projekti, ideje za naprej, spisEK opreme:

VIN-VSP 202324 zvezek ▾
_Knjižnica vsebine ↶

Uporaba knjižnice vseb... DN1 22-23 ▾ +

🔍 Išči po zvezkih ▾

Zunanji pomnilnik SSD

Monday, June 05, 2023 3:02 PM

PDF predstavitev



Predstavitev

Grafični poster

SSD

Pomnilnik SSD je vrsta pomnilniške naprave, ki se uporablja v računalnikih. Ta se uporablja zaradi svojih hitrosti in zanesljivosti pri shranjevanju podatkov. To je posledica napovednega delovanja, ki temelji na tehnologiji NAND flash pomnilnika. Pomnilnik NAND flash je vrsta obtožne tehnologije shranjevanja, ki ne potrebuje energije za shranjevanje podatkov. Ker pomnilnik SSD ne uporablja gibljivih delov, kot so plošče in glavice, ki jih najdemo v trdih diskih (HDD), je ta bolj vzdržljiva in odporen na tresenje, kar ga naredi idealnega za prenosne računalnike in druge naprave, ki so pogosto izpostavljene fizični obravnavi.

NAND flash pomnilnik

To so čipi, ki dejansko hranijo podatke v SSD-ju. NAND-flash pomnilnik deluje tako, da elektrone nabije shranjuje v pomnilniških celicah. Vsaka celica lahko shranjuje več bitov podatkov, odvisno od vrste NAND-flash pomnilnika, ki se uporablja. Uporablja se tudi 3D NAND flash, ki omogoča tako imenovani vstajajoči pomnilniški celic kar poveča samo kapaciteto SSD-ja.

Krmilnik

To je čip, ki upravlja delovanje SSD-ja in nadzira prenos podatkov med gostilnijskim računalnikom in NAND-flash pomnilniškimi čipi v SSD-ju. Krmilnik je odgovoren tudi za nadzor nad funkcijami, kot so samodejno izklop, popravljanje nesil in zbiranje smisla, ki izboljšujeta zmogljivost in zanesljivost SSD-ja.

DRAM predpomnilnik

SSD-ji pogosto uporabljajo dinamični pomnilnik DRAM kot predpomnilnik za hrambo pogosto dostopnih podatkov. DRAM predpomnilnik omogoča hitrejši dostop do podatkov in izboljšuje hitrost branja in pisanja.

Vmesnik

Vmesnik omogoča povezavo med SSD-jem in gostilnijskim računalnikom. Običajno se uporabljajo vmesniki, kot so SATA, PCIe in M.2.

↗

[Dodaj stran](#) ⌵

Senzorji za merjenje kakovosti zraka ...

Zunanji pomnilnik SSD

Poročilo



Poročilo

Vsebina

1. Asinhronski serijski prenos
 - **UART** (Universal Asynchronous Receiver/Transmitter)
 - (RS232, RS422, RS485)
 - **(CANBus)**
2. Sinhronski serijski prenos
 - **USART** (Universal Synchronous/Asynchronous Receiver/Transmitter)
 - **I2C** (Inter-Integrated Circuit)
 - **SPI** (Serial Peripheral Interface)
 - **(USB)**

Gradivo:

- ❑ PROTOCOLS: UART - I2C - SPI - Serial communications

https://www.youtube.com/watch?v=lyGwvGzrqp8&t=25s&ab_channel=Electronoobs

- ❑ Drugi viri so pri posameznih opisih

Uvod

Primer STM32H7:

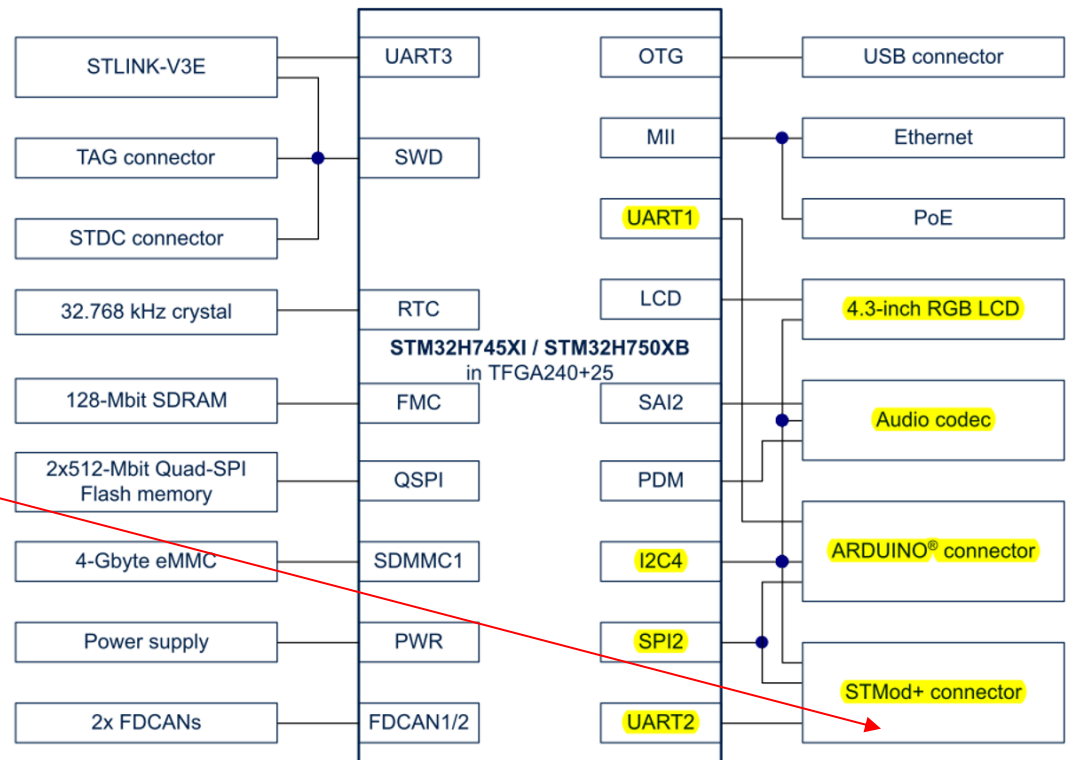
1. Asinhronski serijski prenos

- **U(S)ART** (Universal Asynchronous Receiver/Transmitter)
 - **U(S)ART1,2,3,4,5,6,7,8**
- **FDCAN 1,2**

2. Sinhronski serijski prenos

- **I2C** (Inter-Integrated Circuit)
 - **I2C 1, 2, 3, 4**
- **SPI** (Serial Peripheral Interface)
 - **SPI 1,2,3,4,5,6**

Figure 3. Hardware block diagram



2.5 mikroBUS™ compatible connectors CN10 and CN11

mikroBUS™ compatible connectors CN10 and CN11 are a pair of 1x8-pin female connectors with a 2.54 mm pitch. Table 7 shows the definition of the pins.

Table 7. Description of the mikroBUS™ connector pins

| STMod+ connector CN11 | mikroBUS™ function | Pin number | Pin number | mikroBUS™ function | STMod+ connector CN10 |
|-----------------------|--------------------|------------|------------|--------------------|-----------------------|
| STMod+#13-ADC | AN | 1 | 1 | PWM | STMod+#14-PWM |
| STMod+#12-RST | RST | 2 | 2 | INT | STMod+#11-INT |
| STMod+#1-CS | CS | 3 | 3 | RX | STMod+#3-RX |
| STMod+#4-SCK | SCK | 4 | 4 | TX | STMod+#2-TX |
| STMod+#9-MISOs | MISO | 5 | 5 | SCL | STMod+#7-SCL |
| STMod+#8-MOSIs | MOSI | 6 | 6 | SDA | STMod+#10-SDA |
| - | +3.3 V | 7 | 7 | +5 V | - |
| - | GND | 8 | 8 | GND | - |

Uvod

Primer STM32H7:

1. Asinhronski serijski prenos

- **U(S)ART** (Universal Asynchronous Receiver/Transmitter)
 - U(S)ART 1,2,3,4,5,6,7,8
- **FDCAN 1,2**

2. Sinhronski serijski prenos

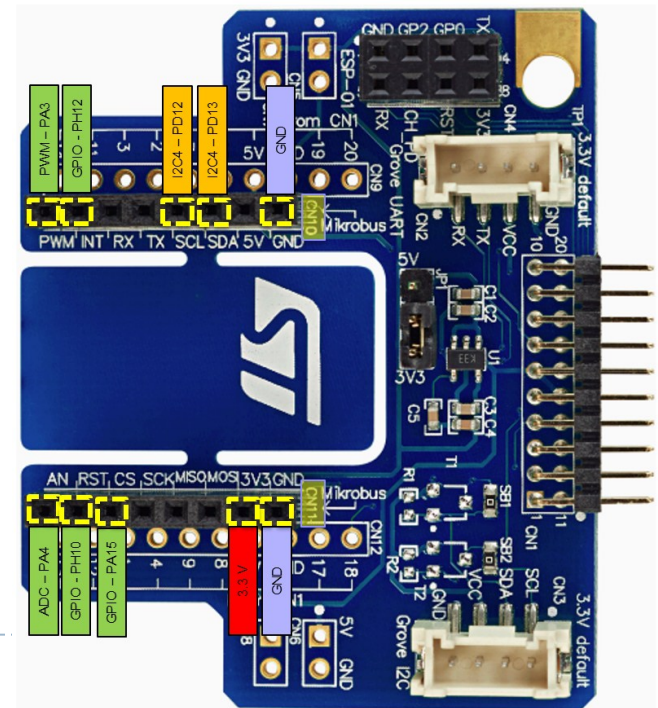
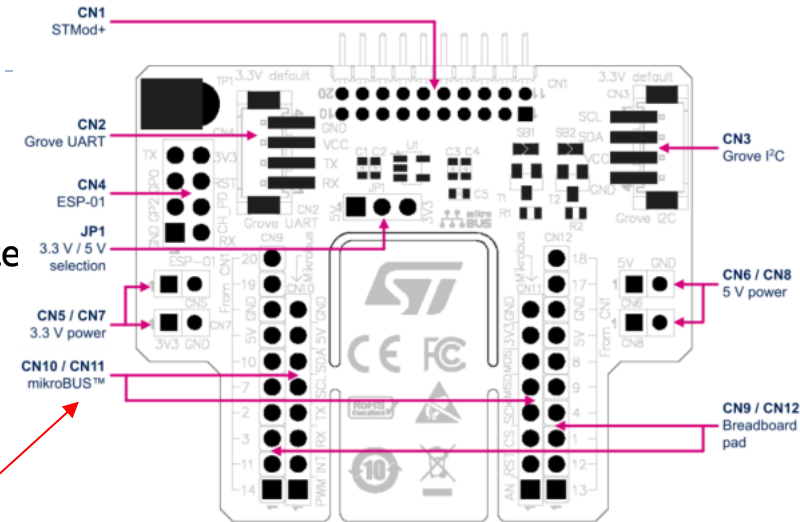
- **I2C** (Inter-Integrated Circuit)
 - I2C 1, 2, 3, 4
- **SPI** (Serial Peripheral Interface)
 - SPI 1,2,3,4,5,6

2.5 mikroBUS™ compatible connectors CN10 and CN11

mikroBUS™ compatible connectors CN10 and CN11 are a pair of 1×8-pin female connectors with a 2.54 mm pitch. Table 7 shows the definition of the pins.

Table 7. Description of the mikroBUS™ connector pins

| STMod+ connector CN11 | mikroBUS™ function | Pin number | Pin number | mikroBUS™ function | STMod+ connector CN10 |
|-----------------------|--------------------|------------|------------|--------------------|-----------------------|
| STMod+#13-ADC | AN | 1 | 1 | PWM | STMod+#14-PWM |
| STMod+#12-RST | RST | 2 | 2 | INT | STMod+#11-INT |
| STMod+#1-CS | CS | 3 | 3 | RX | STMod+#3-RX |
| STMod+#4-SCK | SCK | 4 | 4 | TX | STMod+#2-TX |
| STMod+#9-MISOs | MISO | 5 | 5 | SCL | STMod+#7-SCL |
| STMod+#8-MOSIs | MOSI | 6 | 6 | SDA | STMod+#10-SDA |
| - | +3.3 V | 7 | 7 | +5 V | - |
| - | GND | 8 | 8 | GND | - |

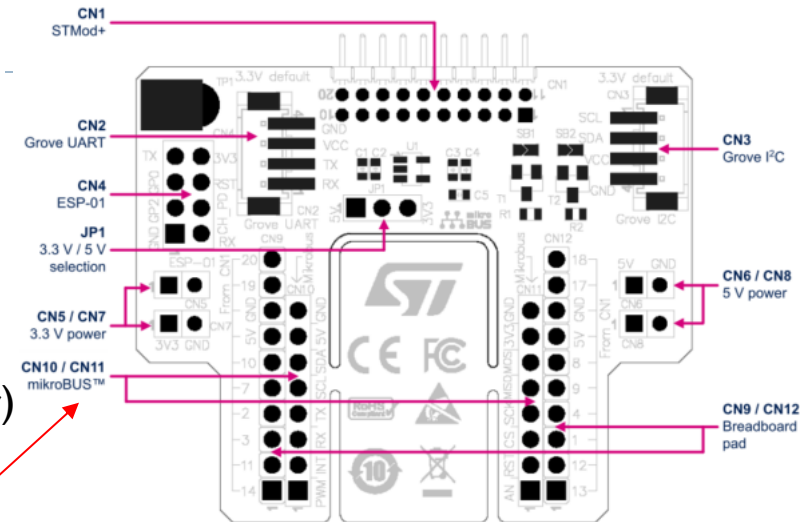


Uvod

STM32H7: Priključitev Click Board modulov

- ▶ Uporabljajo eno (ali več) od serijskih komunikacij na STMod+ plošči:

- **UART2** (Universal Asynchronous Receiver/Transmitter)
- **I2C4** (Inter-Integrated Circuit)
- **SPI2** (Serial Peripheral Interface)



1950 CLICK BOARDS™

You name it, we got it!



Skoll-I Click



Thermo 10 Click



Stepper 29 Click



Angle 11 Click

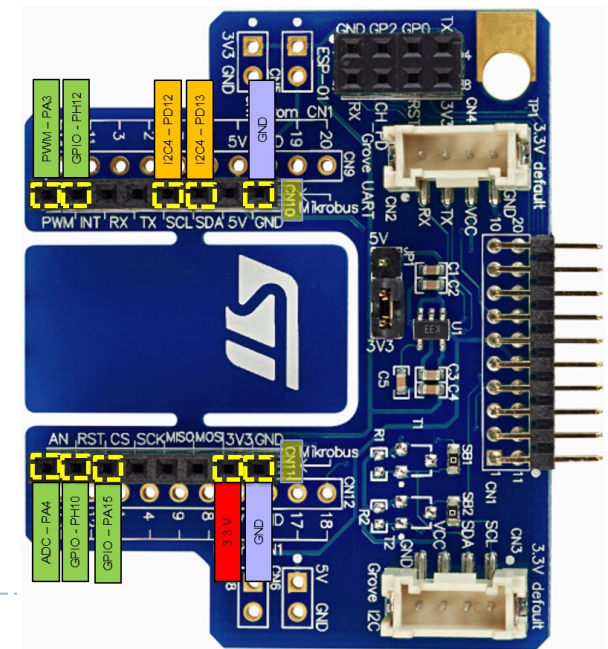


ETH WIZ 3 Click



Ultrasonic 4 Click

<https://www.mikroe.com/click-boards>



Uvod

Primer STM32F4:

1. Asinhronski serijski prenos

- **U(S)ART** (Universal (A)synchronous Receiver/Transmitter)
 - **USART1,2,3,4,5,6**
- **CAN 1,2**

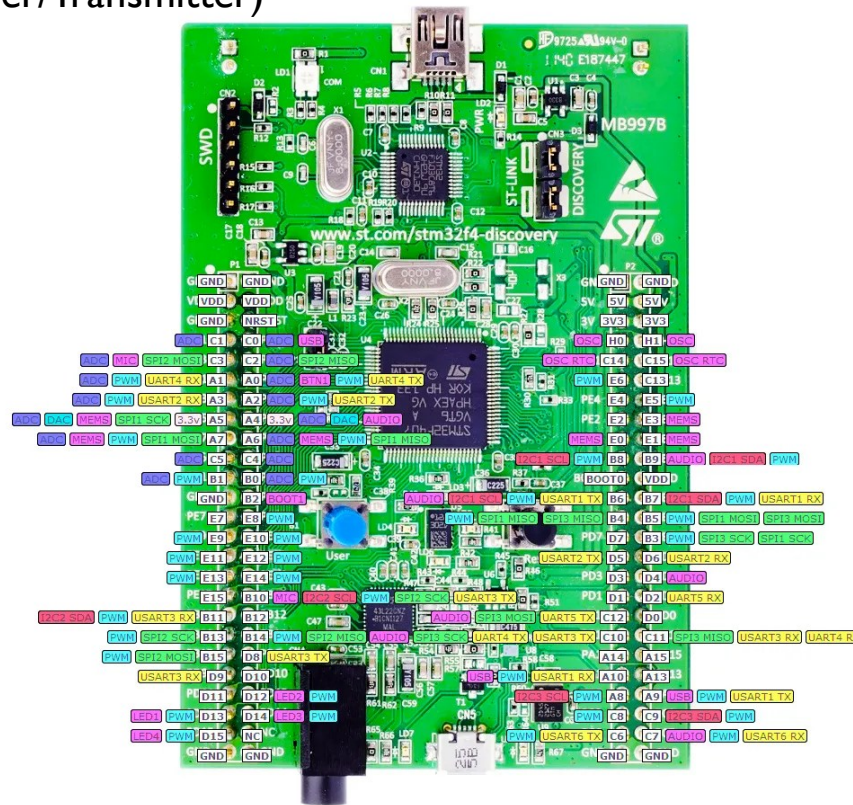
STM32F4 Discovery

2. Sinhronski serijski prenos

- **I2C** (Inter-Integrated Circuit)
 - **I2C1, I2C2, I2C3**
- **SPI** (Serial Peripheral Interface)
 - **SPI1, SPI2, SPI3**

P1

| | |
|----|----|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
| 7 | 8 |
| 9 | 10 |
| 11 | 12 |
| 13 | 14 |
| 15 | 16 |
| 17 | 18 |
| 19 | 20 |
| 21 | 22 |
| 23 | 24 |
| 25 | 26 |
| 27 | 28 |
| 29 | 30 |
| 31 | 32 |
| 33 | 34 |
| 35 | 36 |
| 37 | 38 |
| 39 | 40 |
| 41 | 42 |
| 43 | 44 |
| 45 | 46 |
| 47 | 48 |
| 49 | 50 |



P2

| | |
|----|----|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
| 7 | 8 |
| 9 | 10 |
| 11 | 12 |
| 13 | 14 |
| 15 | 16 |
| 17 | 18 |
| 19 | 20 |
| 21 | 22 |
| 23 | 24 |
| 25 | 26 |
| 27 | 28 |
| 29 | 30 |
| 31 | 32 |
| 33 | 34 |
| 35 | 36 |
| 37 | 38 |
| 39 | 40 |
| 41 | 42 |
| 43 | 44 |
| 45 | 46 |
| 47 | 48 |
| 49 | 50 |

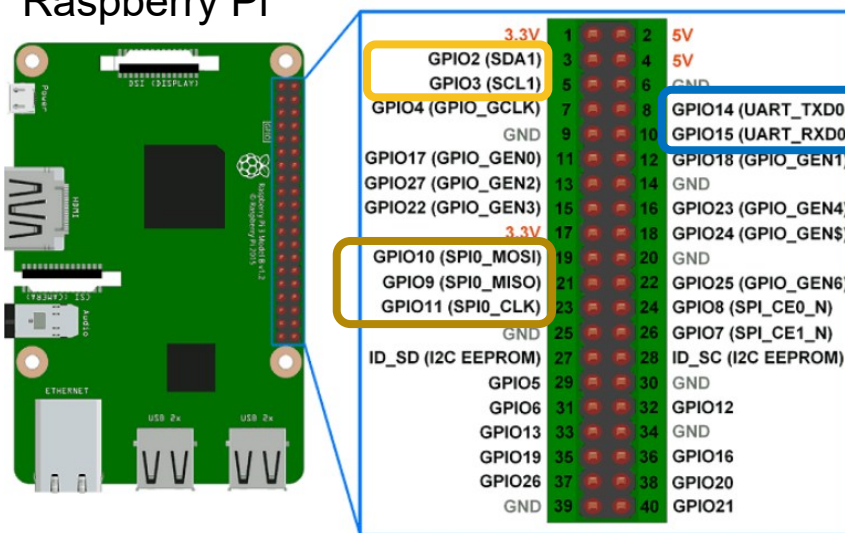
<https://microcontrollerslab.com/stm32f4-discovery-board-pinout-features-examples/>

Uvod

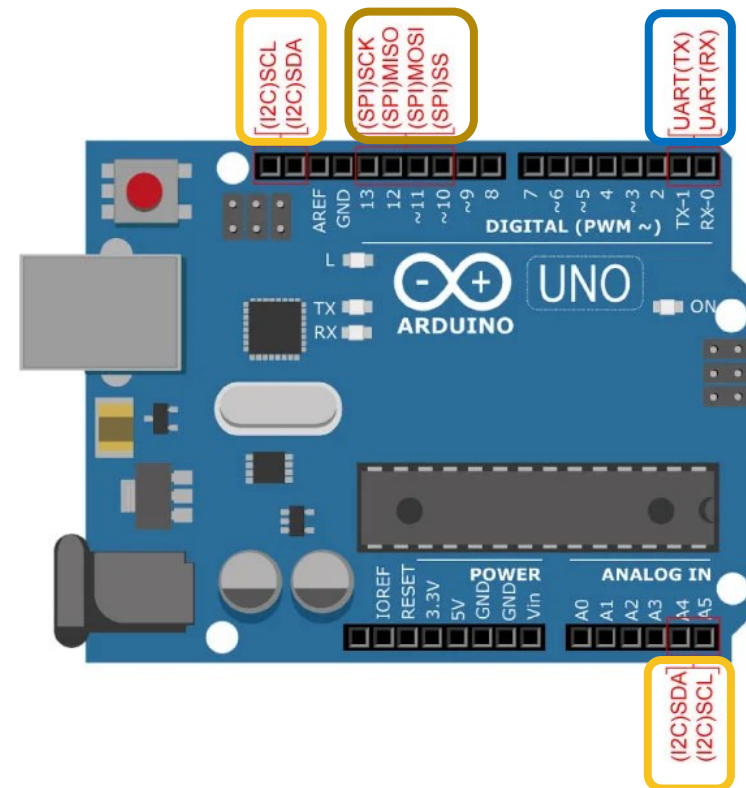
Primer RPi, Arduino:

1. Asinhronski serijski prenos
 - **UART** (Universal Asynchronous Receiver/Transmitter)
2. Sinhronski serijski prenos
 - **I2C** (Inter-Integrated Circuit)
 - **SPI** (Serial Peripheral Interface)

Raspberry Pi



Arduino UNO



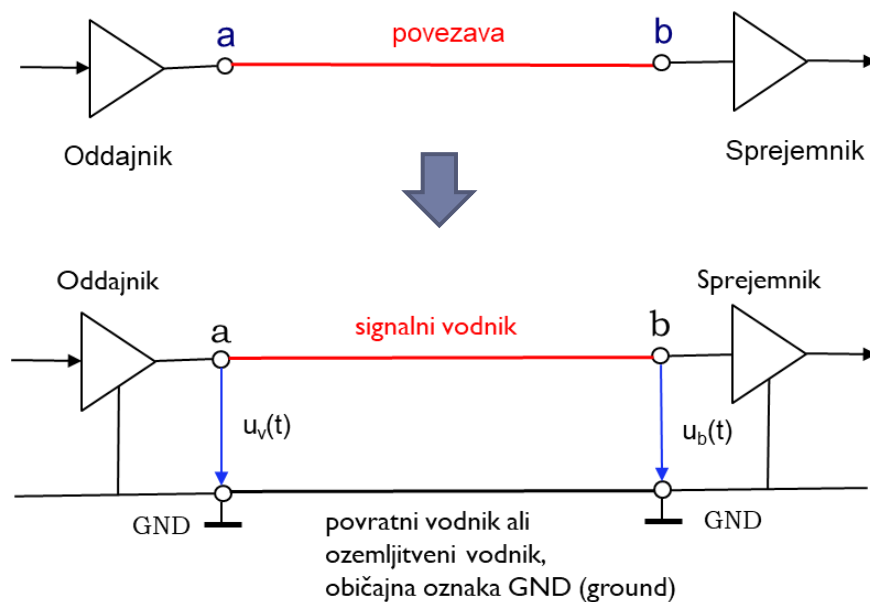
<https://www.electronicwings.com/raspberry-pi/raspberry-pi-gpio-access>

<https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi>

- ❑ **Naprave:** računalnik, mikroprocesor, mikrokrmilnik, V/I naprave



- ❑ Prenosni medij je fiksni ali žičen – **električna povezava, ali linija**
simbolna predstavitev



❑ Serijski način prenosa

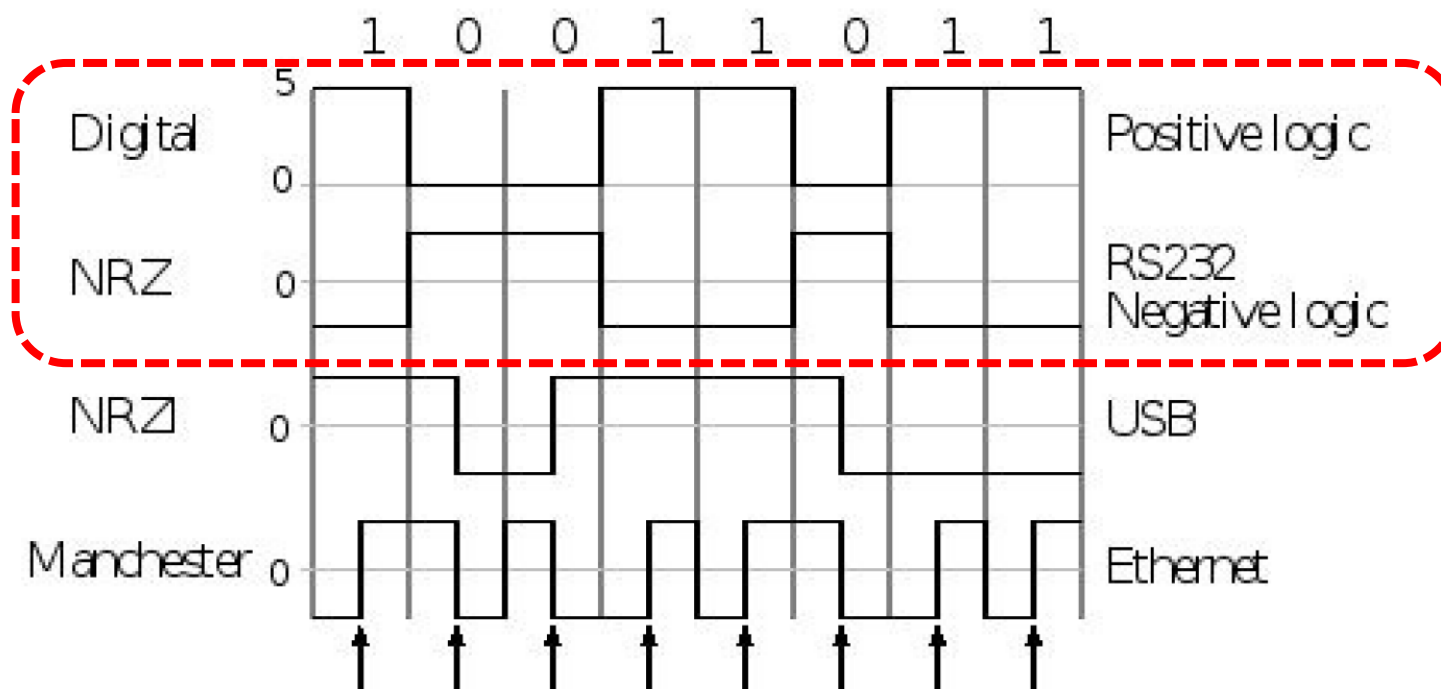
po povezavi se prenaša bit za bitom

❑ KOMUNIKACIJSKI KANAL

- Prenos podatkov poteka v **realnem času** (čas med oddajo in sprejemom podatkov je zelo kratek).
- Potrebne pretvorbe:
 - Pri oddaji podatka je potrebna **paralelno – serijska** pretvorba.
 - Pri sprejemu podatka je potrebna **serijsko – paralelna** pretvorba.
- Glede na **način sinhronizacije (obstoj skupnega urinega signala)** razlikujemo:
 - **Asinhronski** serijski prenos podatkov
 - **Sinhronski** serijski prenos podatkov
- Potrebno je izvesti tudi **kodiranje podatkov** (pretvorba podatka v signal – pogl. 8):
 - NRZ(I) (Non Return to Zero (Inverted))
 - PE (Phase Encoded)
 - RLL (Run Length Limited): 8b/10b, ...

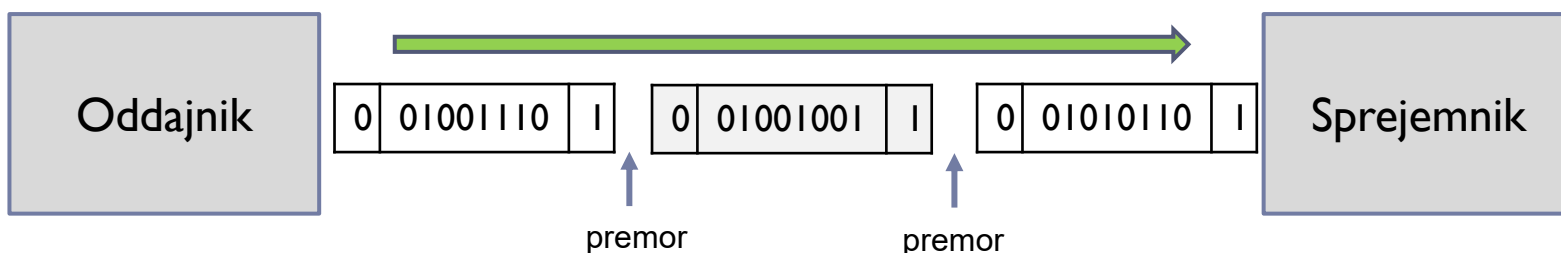
Zaenkrat se ukvarjamo z običajnim kodiranjem s pravokotnim signalom (NRZ - nivojsko kodiranje)...

- **kodiranje podatkov** (pretvorba podatka v dejanski signal – pogl. 8):
 - NRZ(I) (Non Return to Zero (Inverted))
 - PE (Phase Encoded)
 - RLL (Run Length Limited): 8b/10b, ...



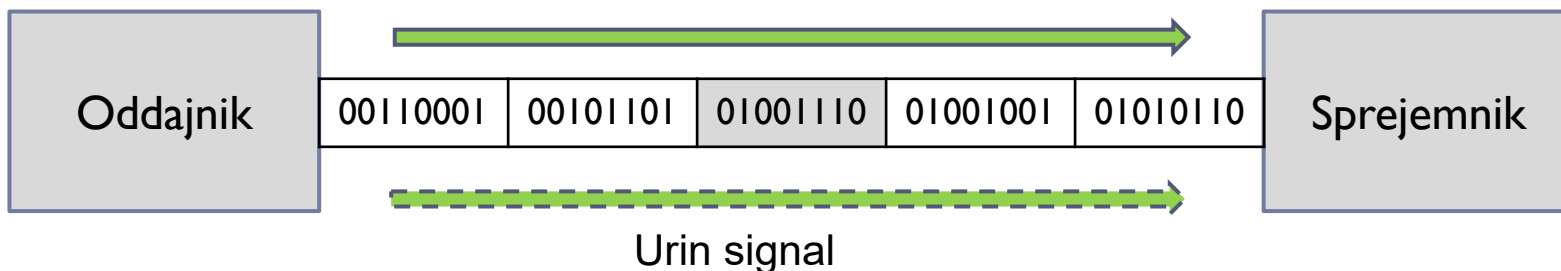
□ 4.1 Asinhronski serijski prenos

- UART (Universal Asynchronous Receiver/Transmitter)
- CANBUS



□ 4.2 Sinhronski serijski prenos

- I2C (Inter-Integrated Circuit)
- SPI (Serial Peripheral Interface)
- USART, hitre serijske povezave



4.1 Asinhronski serijski prenos

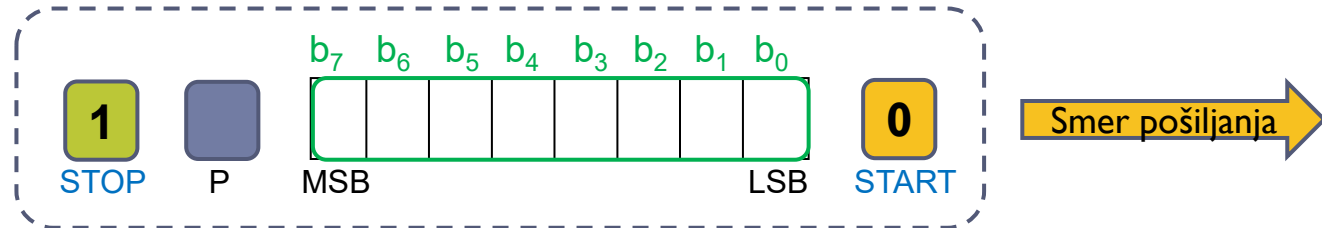
- ❑ **Enota pošiljanja je krajša (znak, sporočilo).**
- ❑ **Biti se pošiljajo** po komunikacijskem kanalu.
- ❑ Oddajnik in sprejemnik imata vsak svojo uro. **Urin signal se NE pošilja** po komunikacijskem kanalu.
- ❑ Za pravilen prenos je nujno, da sta **frekvenci oddajne in sprejemne ure enaki in sinhronizirani**
- ❑ Sinhronizacija se vzpostavi za vsak znak posebej.
- ❑ Sinhronizacija sprejemne ure se z oddajno vzpostavi
 - s prvim bitom znaka oziroma sporočila („START“) in
 - mora zagotavljati pravilen sprejem do zadnjega bita znaka („STOP“).

4.1 Asinhronski serijski prenos - UART

- ❑ Prenos podatkov je relativno **počasen**.
- ❑ Primeren je za **neenakomeren prenos znakov** s presledki.
- ❑ Med znaki je pri oddaji lahko **poljubno dolg presledek**.
- ❑ Pošiljamo lahko kakršenkoli **tekst brez protokola** (protokol je samo format znaka).
- ❑ Dodatni („režijski“) biti :
 - START in
 - STOP bit in/ali
 - parnostni (paritetni) bitso v vsakem znaku, kar predstavlja **20% - 30% redundanco** pri prenosu.

4.1 Asinhronski serijski prenos - UART

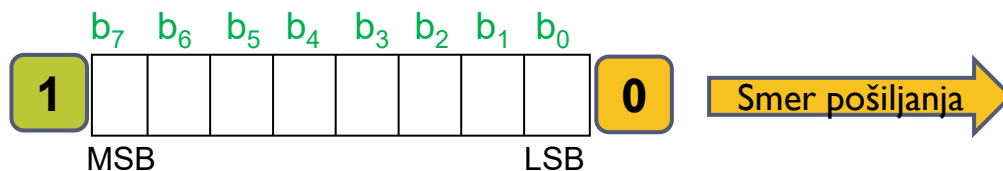
□ Znak sestavljajo:



- **Start bit** je vedno **logična 0** (nasprotno od mirovnega stanja) in določa začetek znaka:
 - služi za vzpostavitev sinhronizacije med sprejemno in oddajno uro.
- **Podatkovni biti** - število bitov se v krmilniku lahko izbere (**5 do 8 bitov**).
 - Bit z najnižjo težo se običajno pošilja prvi: **LSB** (b_0).
- **Parnostni bit - P (paritetni bit)** ni obvezen. Pri oddaji se izračuna iz podatkovnih bitov in služi za kontrolo pravilnosti sprejetega znaka. V krmilniku se lahko izbere:
 - **Soda parnost** (sodo število vseh enic podatkovnih in parnostnega bita).
 - **Liha parnost** (liho število vseh enic podatkovnih in parnostnega bita).
- **Stop bit** je vedno **logična 1** (enako mirovnemu stanju) in označuje konec znaka.
 - Mirovno stanje pred začetkom naslednjega znaka

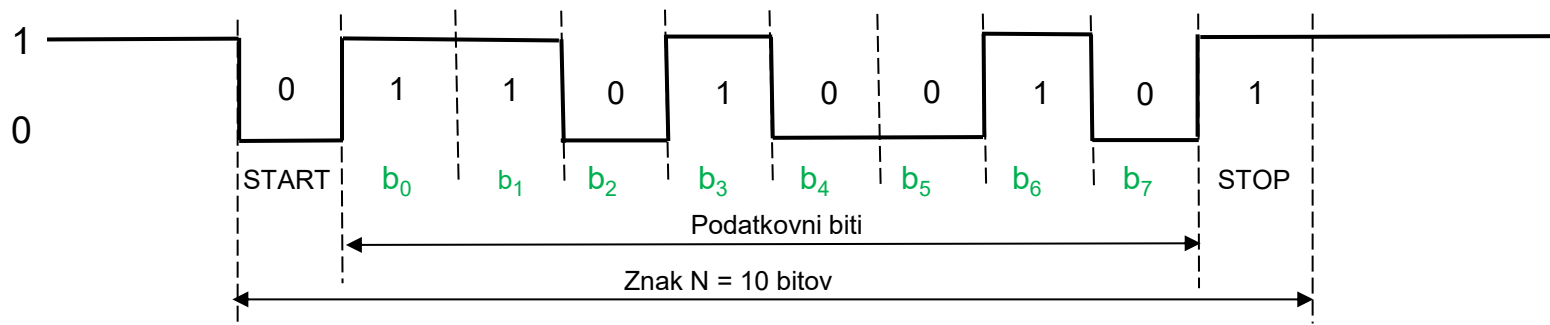
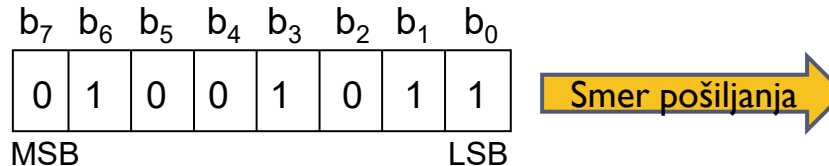
- Najbolj razširjena oblika formata serijskega asinhronskega prenosa pri osebni računalnikih (PC) je **8-N-1** in pomeni:

- 8 podatkovnih bitov,
- brez parnostnega bita,
- en stop bit,
- dolžina znaka je skupaj s start bitom enaka 10 bitov.



- Primer formata znaka ASCII "K" z enim stop bitom:

- ASCII "K" = 4B (Hex)

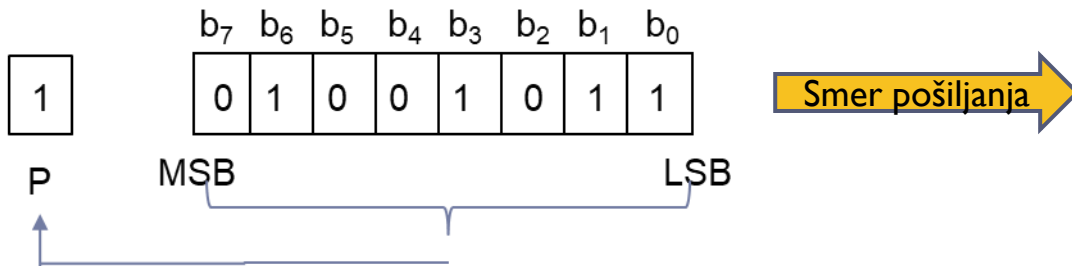


- Format znaka lahko podamo tudi takole:

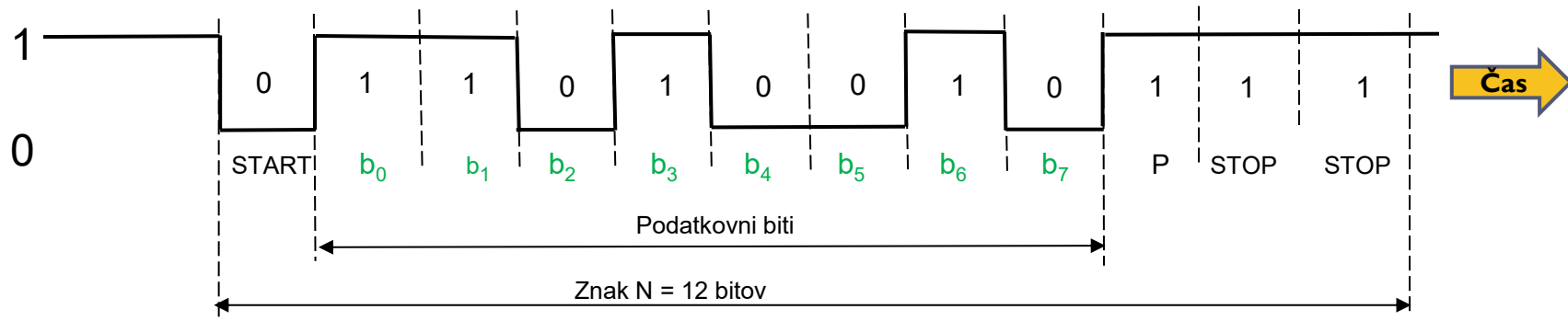


□ Primer formata znaka ASCII "K" z liho parnostjo in dvema stop bitoma:

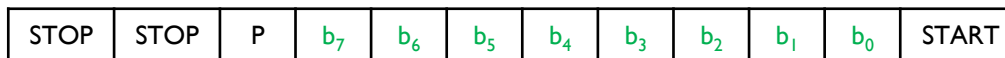
- ASCII "K" = 4B (Hex)
- P = 1



- Oblika prenosa: 8-O-2, kjer je N=12



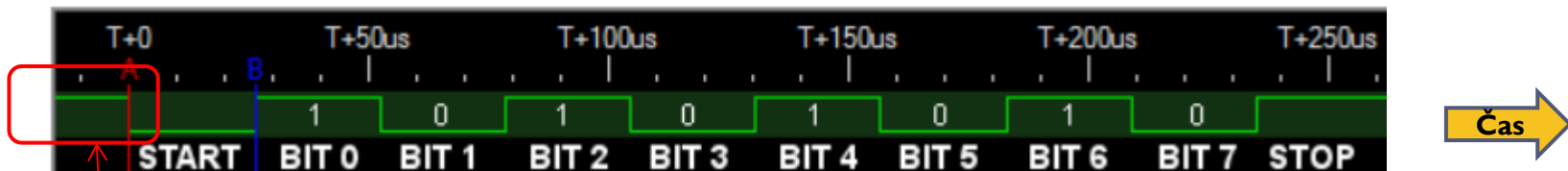
- Format znaka lahko podamo tudi takole (12 bitov):



□ Primer serijskega prenosa črke 'U'

(<http://www.robotroom.com/Asynchronous-Serial-Communication-I.html>)

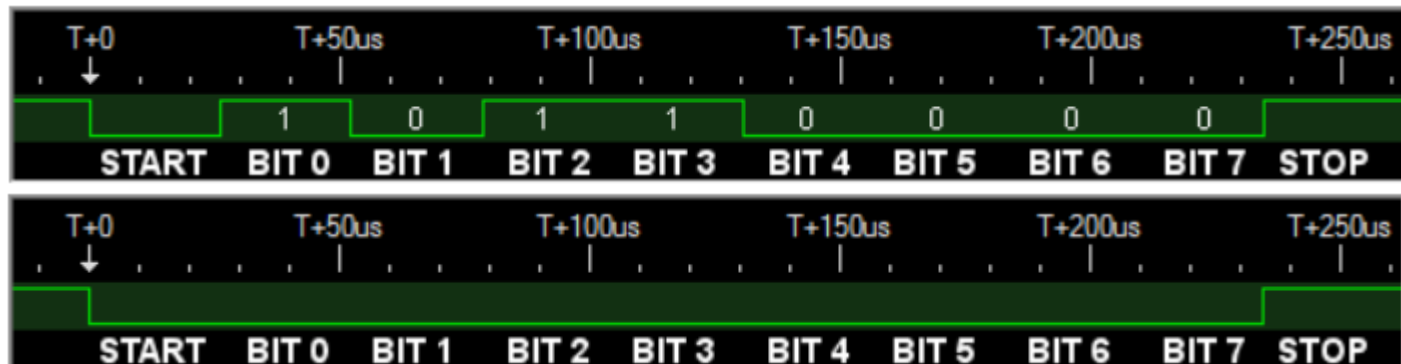
- Koda ASCII ima numerično vrednost 85, ali 55_H).
- Dvojiški 8-bitni zapis podatka je **01010101**



- Pred prvim bitom START je signal na izhodnem pinu high (5V).
- Signal na izhodnem pinu se spremeni v low (0V), kar označuje začetek prenosa (START).
- Nato sledi prenos podatkovnih bitov (BIT 0 do BIT 7) in bit STOP.

□ Drugi primeri:

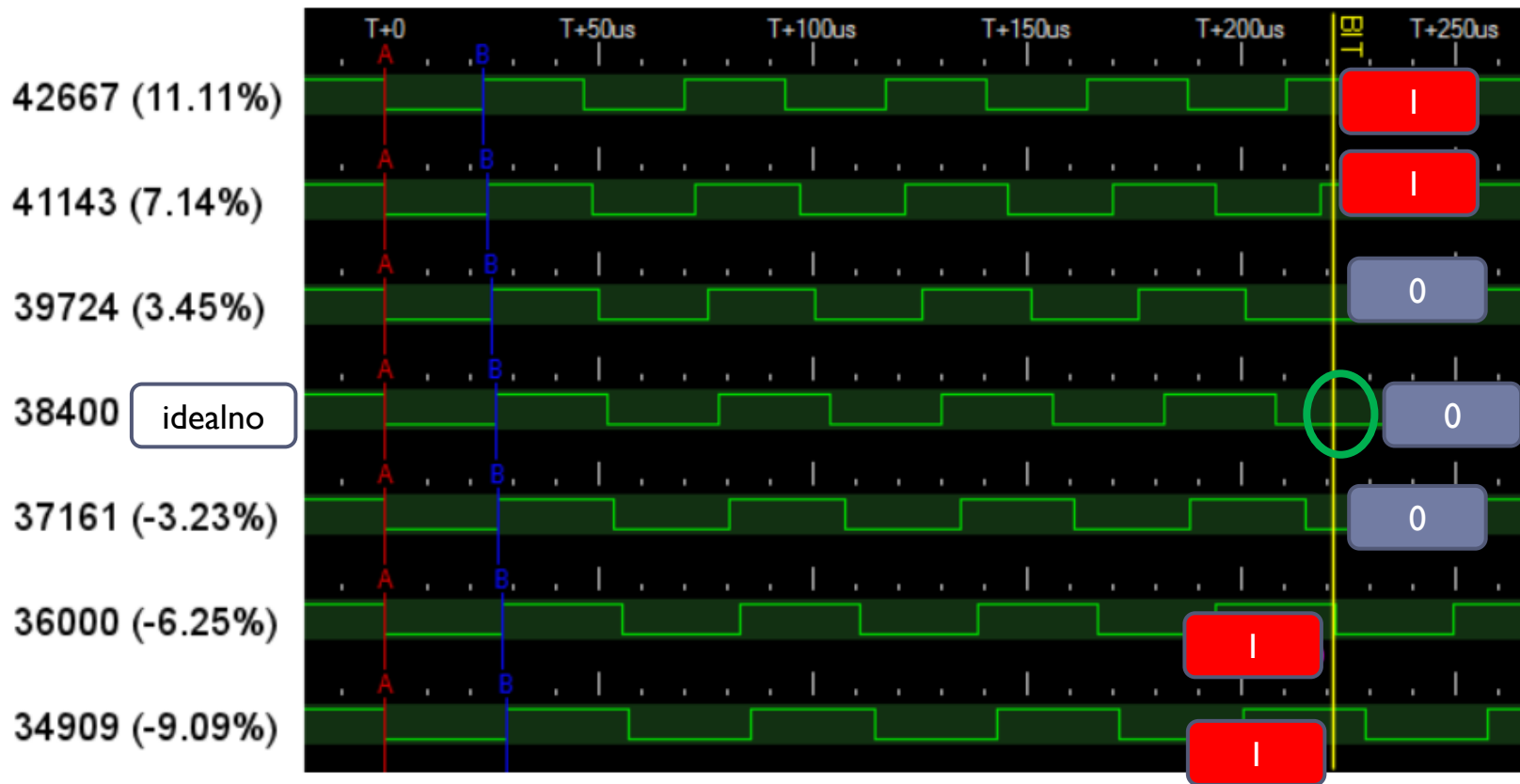
- CR (ASCII 13)
(carriage
return)
- Podatek = 0



□ Primer napačnega prenosa (bitna hitrost je 38400 b/s).

- Napaka se pri prenosu zgodi v primeru, ko je oddajnik prehitel (> 38400 b/s) ali prepočasn (< 38400 b/s).
- razlika v času med oddajno in sprejemno hitrostjo $> 5\%$.

<http://www.robotroom.com/Asynchronous-Serial-Communication-2.html>



Univerzalni asinhronski sprejemnik/oddajnik (UART)

□ UART - Universal Asynchronous Receiver/Transmitter

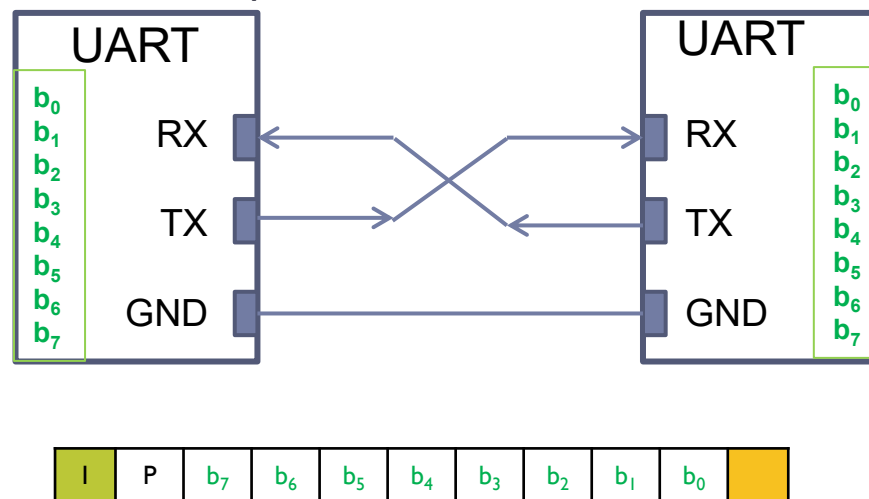
- UART kot samostojen čip v integraciji LSI (Large Scale Integration).
- UART kot enota mikrokrmilnika.
- UART ima 2 liniji za prenos podatkov med dvema napravama:
 - TX (ang. transmitter) – oddaja
 - RX (ang. receiver) – sprejem

„Universal“

- UART is programmable.

„Asynchronous“

- Sender provides no clock signal to receivers



- Oddajni UART_O pretvori paralelni podatek v serijski podatek in ga z dodatnimi biti prenaša sprejemnemu UART_S, ki pretvori sprejeti serijski podatek v paralelni podatek.

- Uporabimo ga lahko v naslednjih načinih prenosa:
 - Napravi sta povezani samo v eno smer, kot oddajnik-sprejemnik (ang. simplex)
 - Napravi izmenoma pošiljata in sprejemata podatke (ang. half-duplex)
 - Napravi istočasno pošiljata in sprejemata podatke (ang. full-duplex)

- Hitrost prenosa podatkov podajamo kot baudna hitrost
 - Najbolj običajna, standardizirana **baudna hitrost je 9600**.
 - Druge standardne baudne hitrosti so:
1200, 2400, 4800, 19200, 38400, 57600 in 115200.

- UART je cenovno ugodna komunikacijska naprava.

- USART (Universal Synchronous Asynchronous Receiver Transmitter) - univerzalni sinhronski in asinhronski sprejemnik / oddajnik, ki omogoča tako asinhronski, kot sinhronski serijski prenos podatkov.

□ Zahteve za pravilen prenos

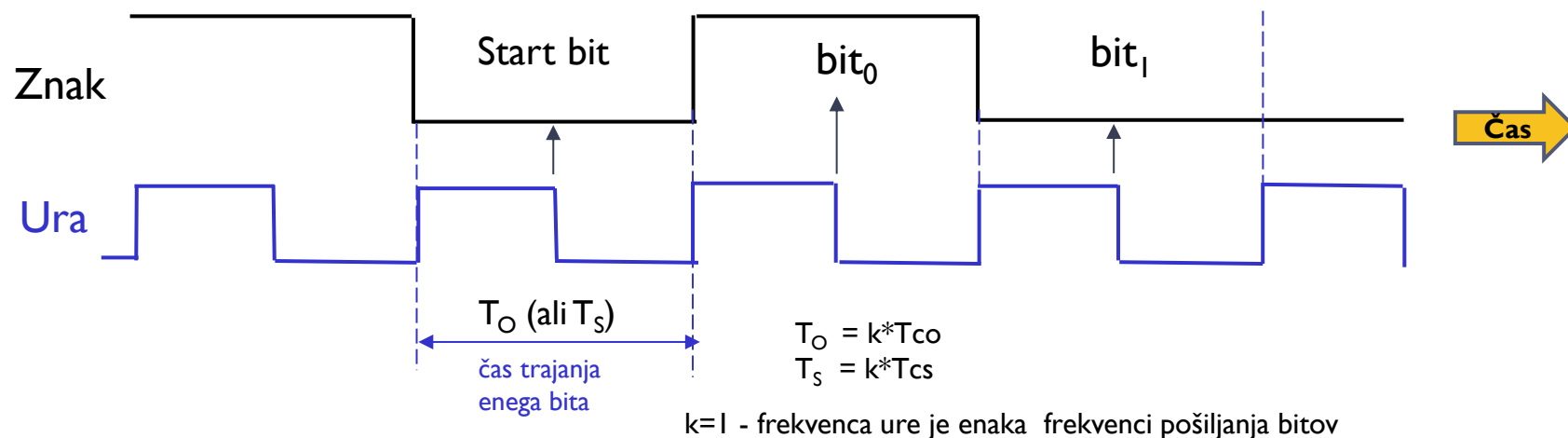
- Enak format, nastavitve na sprejemni in oddajni strani

- Število podatkovnih bitov
- Parnostni bit (da/ne), enaka parnost (soda ali liha)
- Enako število stop bitov (eden ali dva)

- Enaka frekvenca oddajne ure (f_{co}) in sprejemne ure (f_{cs}): $f_{co} = f_{cs}$

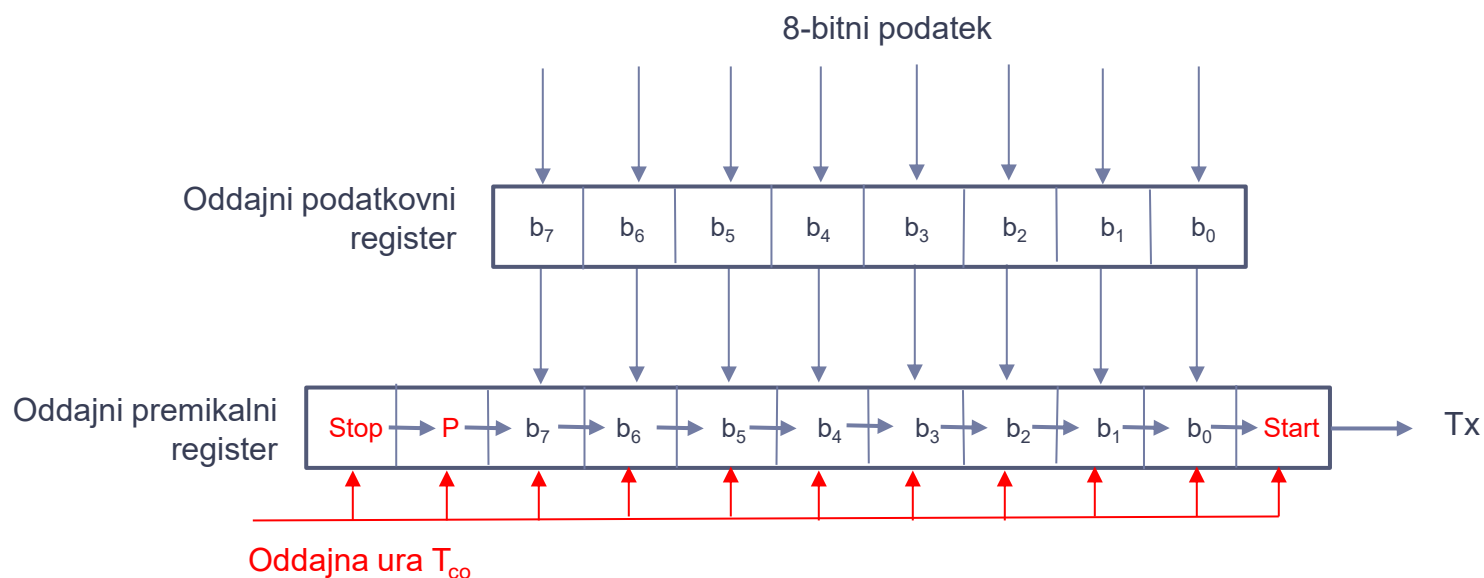
c_o – clk oddaja, c_s – clk sprejem

- Periodi sprejemne ure (T_{cs}) in oddajne ure (T_{co}) se lahko razlikujeta za največ 4 - 5%.



□ UART – oddaja

- Paralelno serijska pretvorba
- Uokvirjanje znaka (ang. framing): start bit, kontrolni parnostni bit, stop bit)
- Oddaja z izbrano baudno hitrostjo



UART - oddaja

Oddajna ura - Tco 4

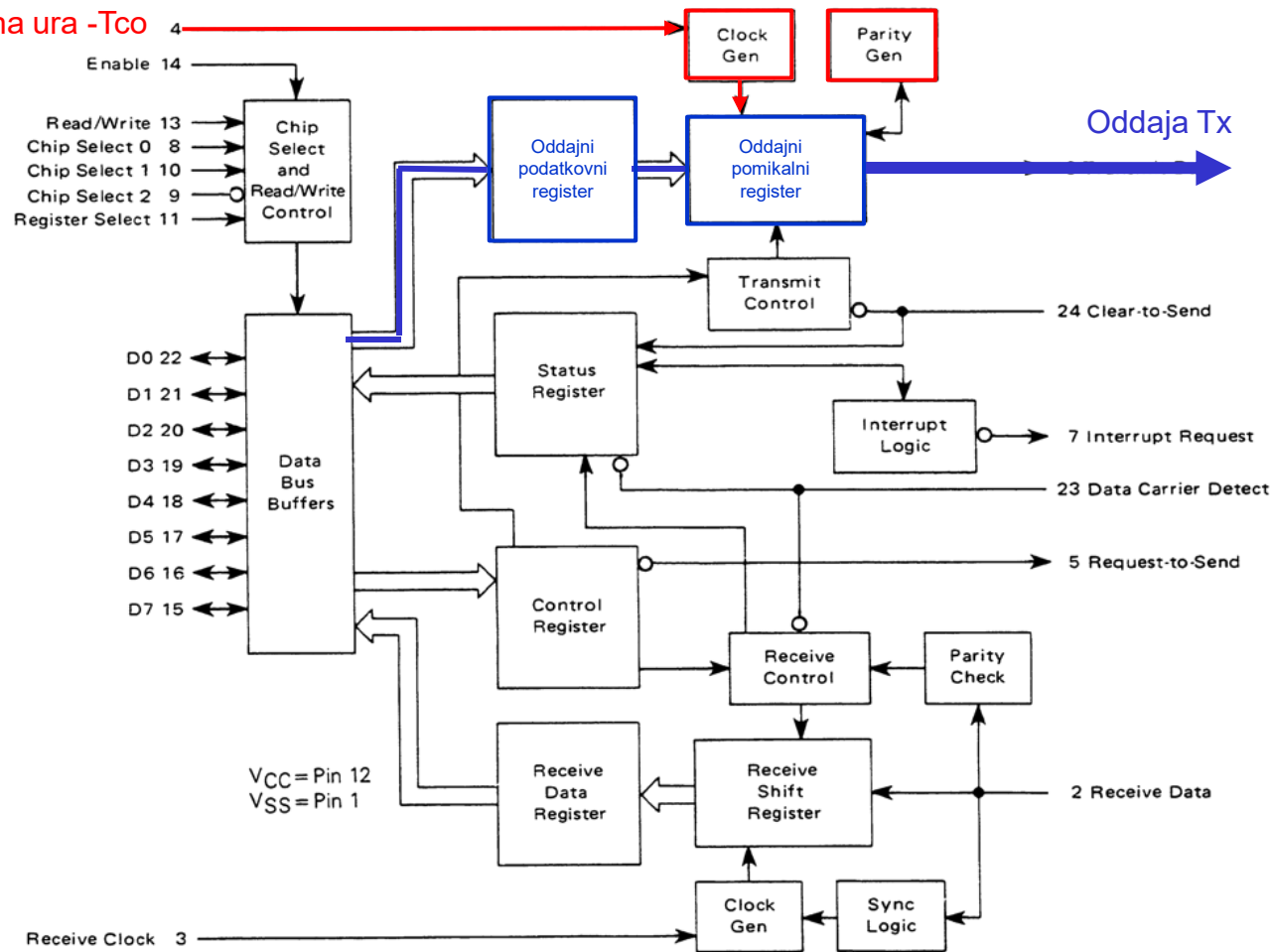
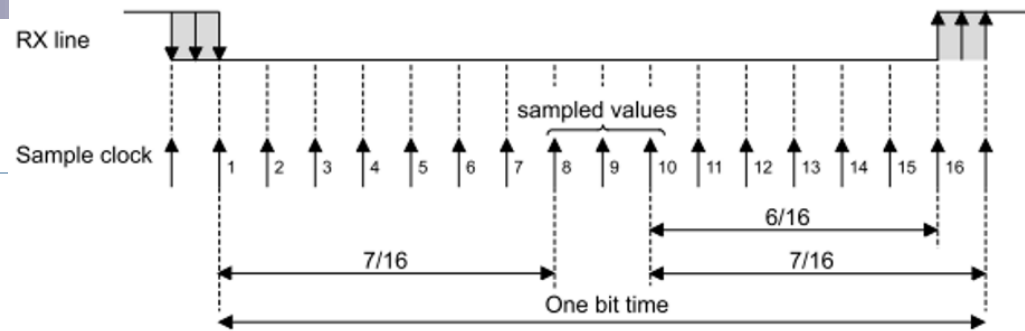


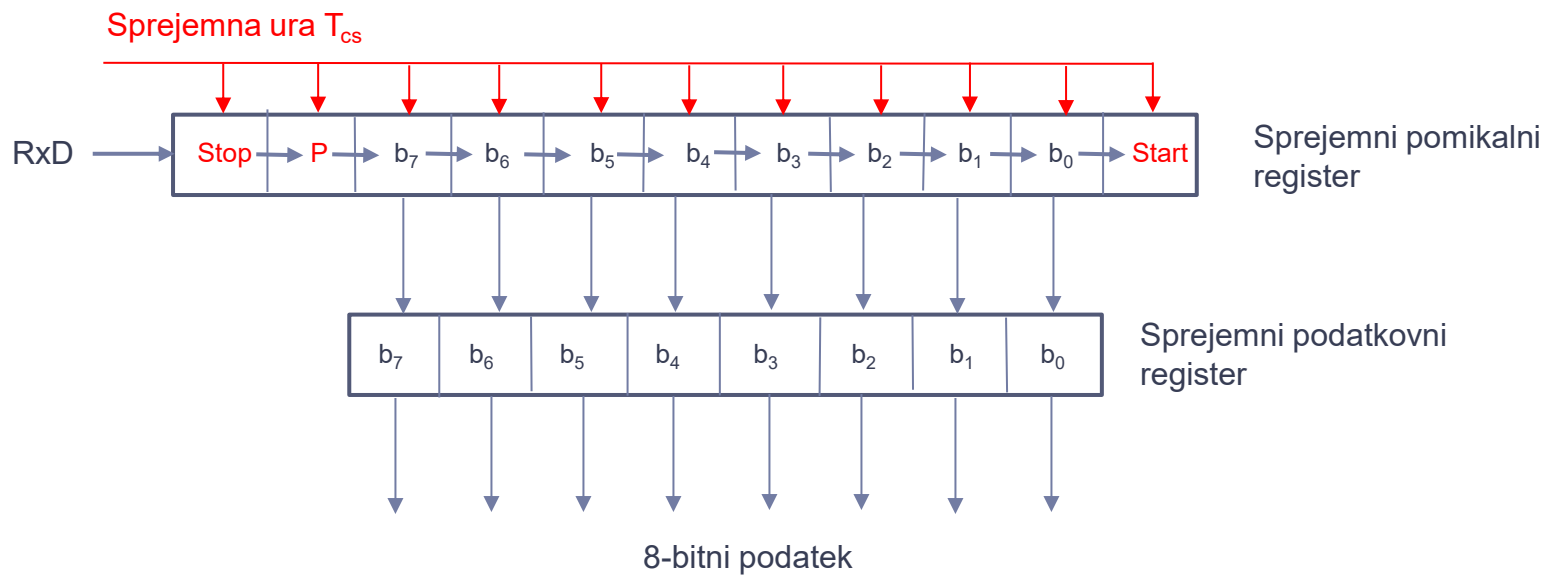
Figure 301. Data sampling when oversampling by 16



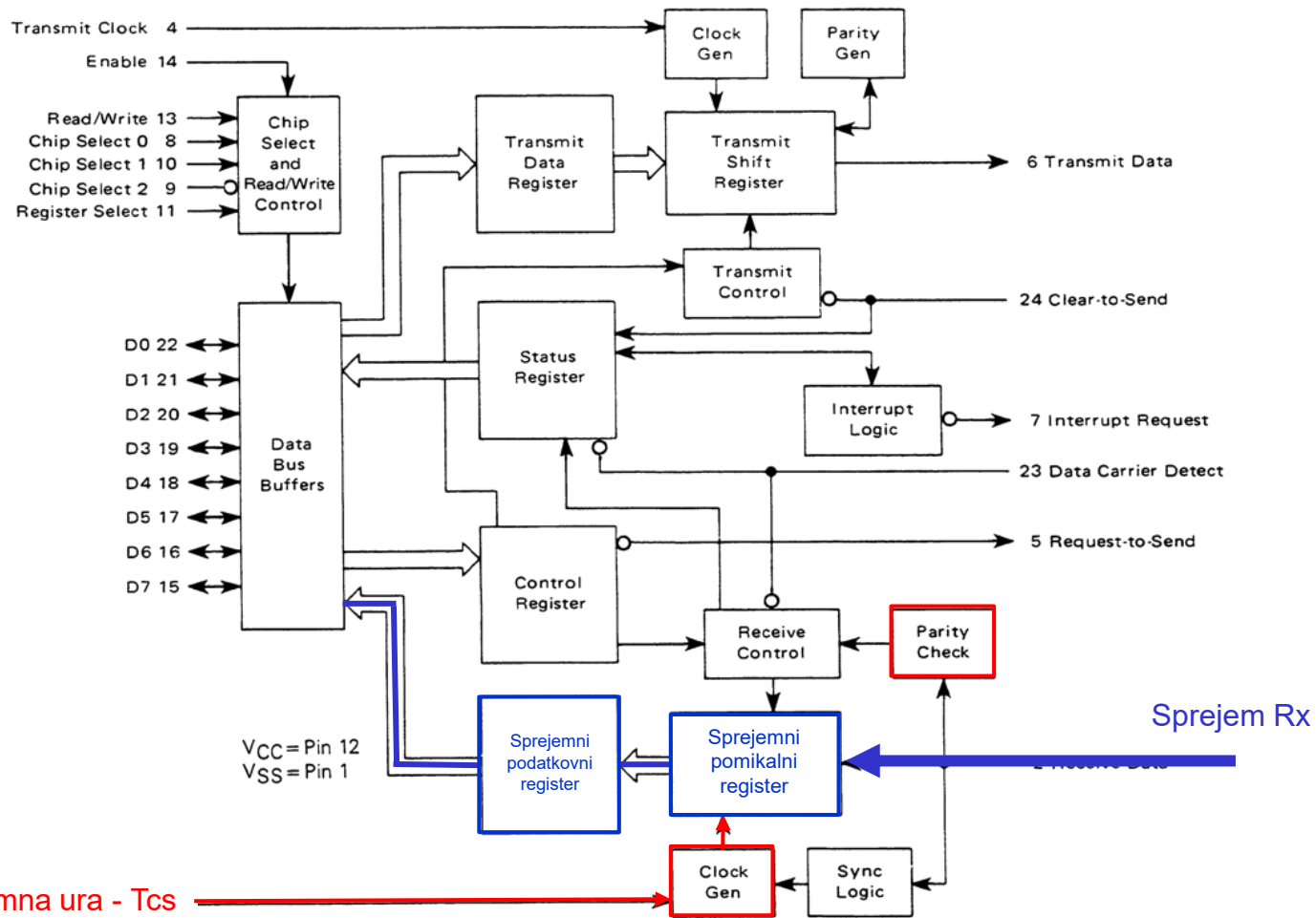
Sprejemnik običajno bere stanje 8 ali 16x pogostejše (angl. Oversampling)

UART – sprejem

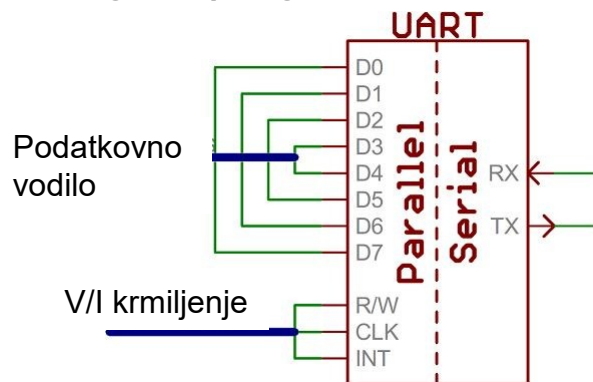
- Serijsko paralelna pretvorba
- Kontrola pravilnosti (parnostni bit)
- Odstranitev okvirja (start, stop bit in parnostni bit)



UART – sprejem



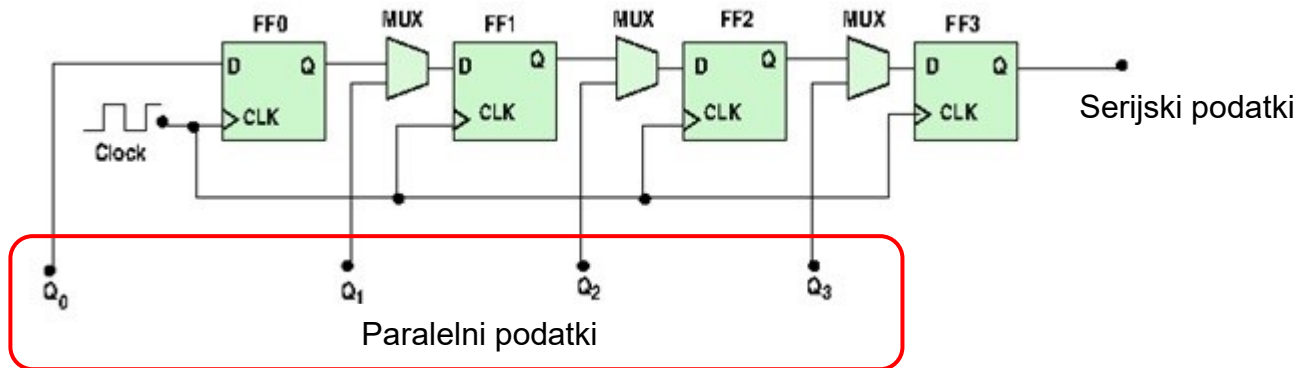
- UART je mogoče programirati, tako da določimo parametre prenosa:



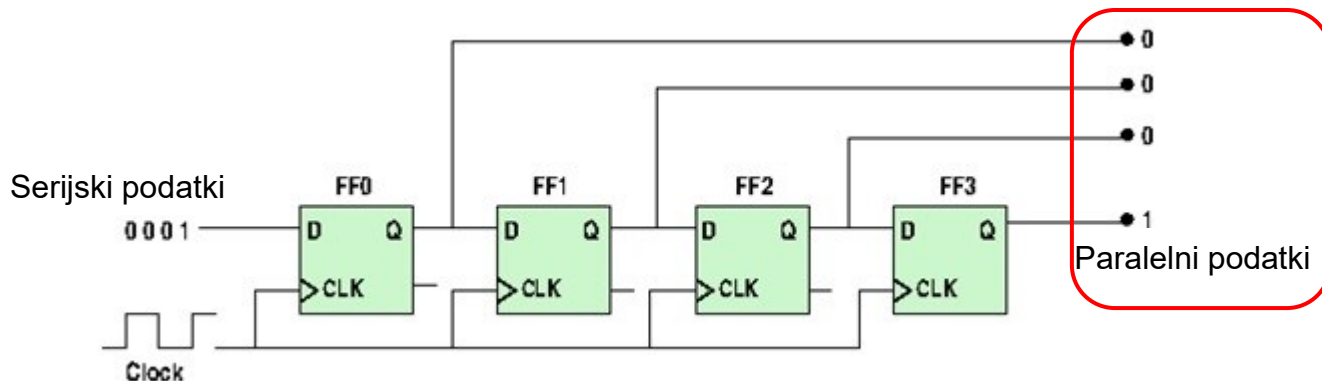
- Baudno hitrost (baud rate) – baudna hitrost = bitna hitrost
- Število podatkovnih bitov
- Parnost (liha, soda, brez)
- Število stop bitov
- Stanje na kontrolnih izhodih
- Prekinitev ob sprejemu znaka
- Prekinitev po oddaji znaka
- Prenos z dvojnimi izravnanimi ali s FIFO vmesnikom

Pomikalni registri

□ Paralelno-serijska pretvorba

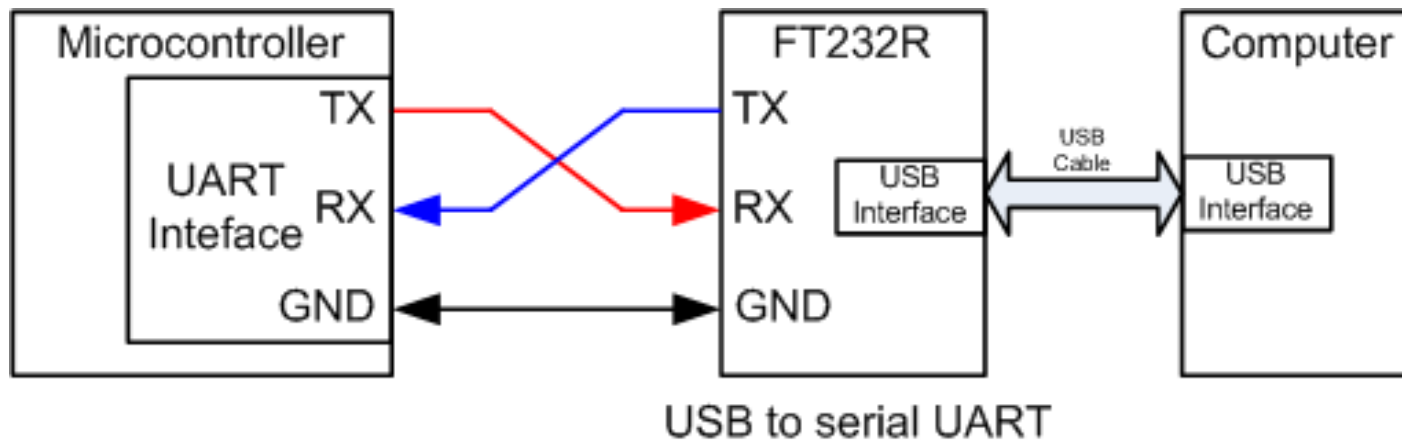
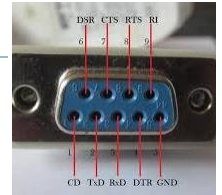


□ Serijsko-paralelna pretvorba



Običajna povezava UART <-> PC

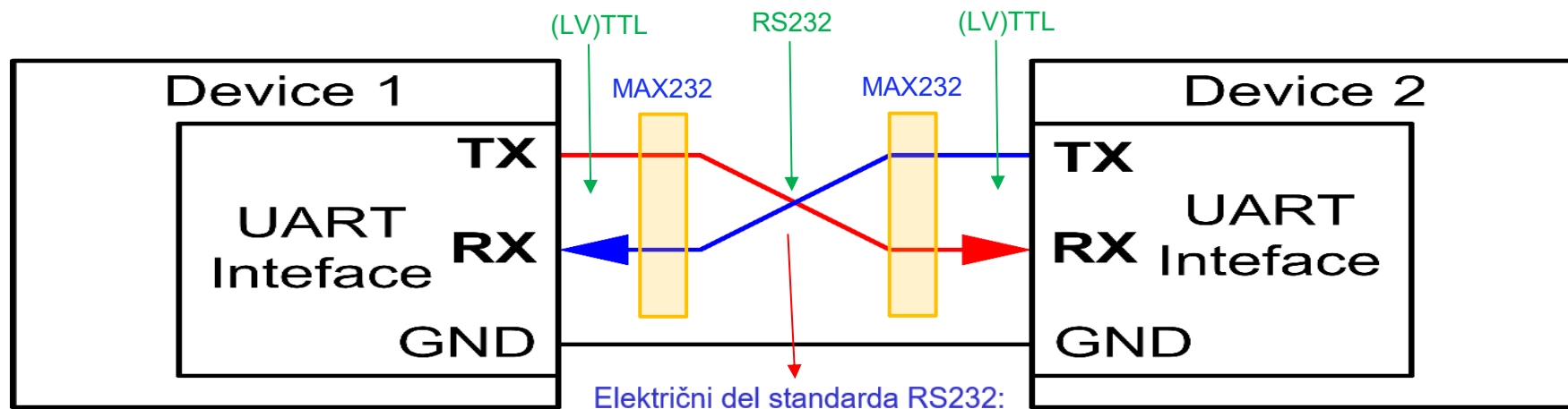
- Primer povezave na PC (običajno nima UART (RS232, COM) priključka):
 - FT232R pretvarja med UART RS232 in USB vmesnikom (PC prenosniki)



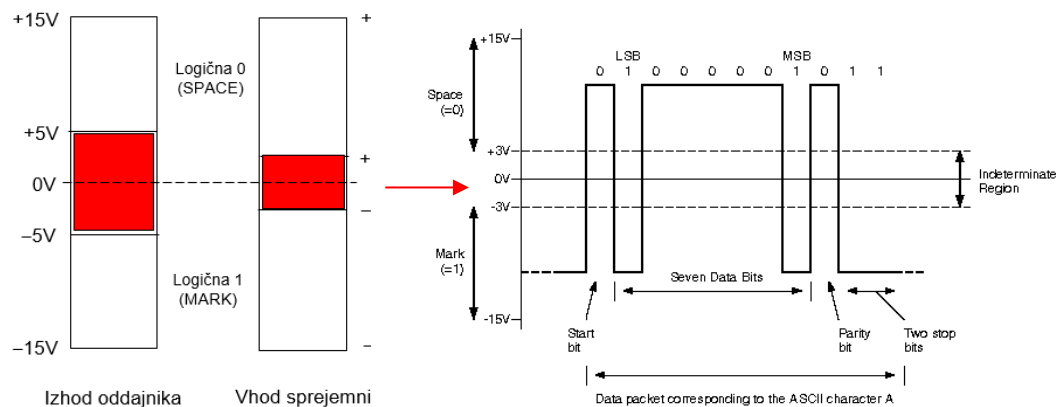
UART - električno

□ Napetostni nivoji so lahko (odvisno od pretvornika):

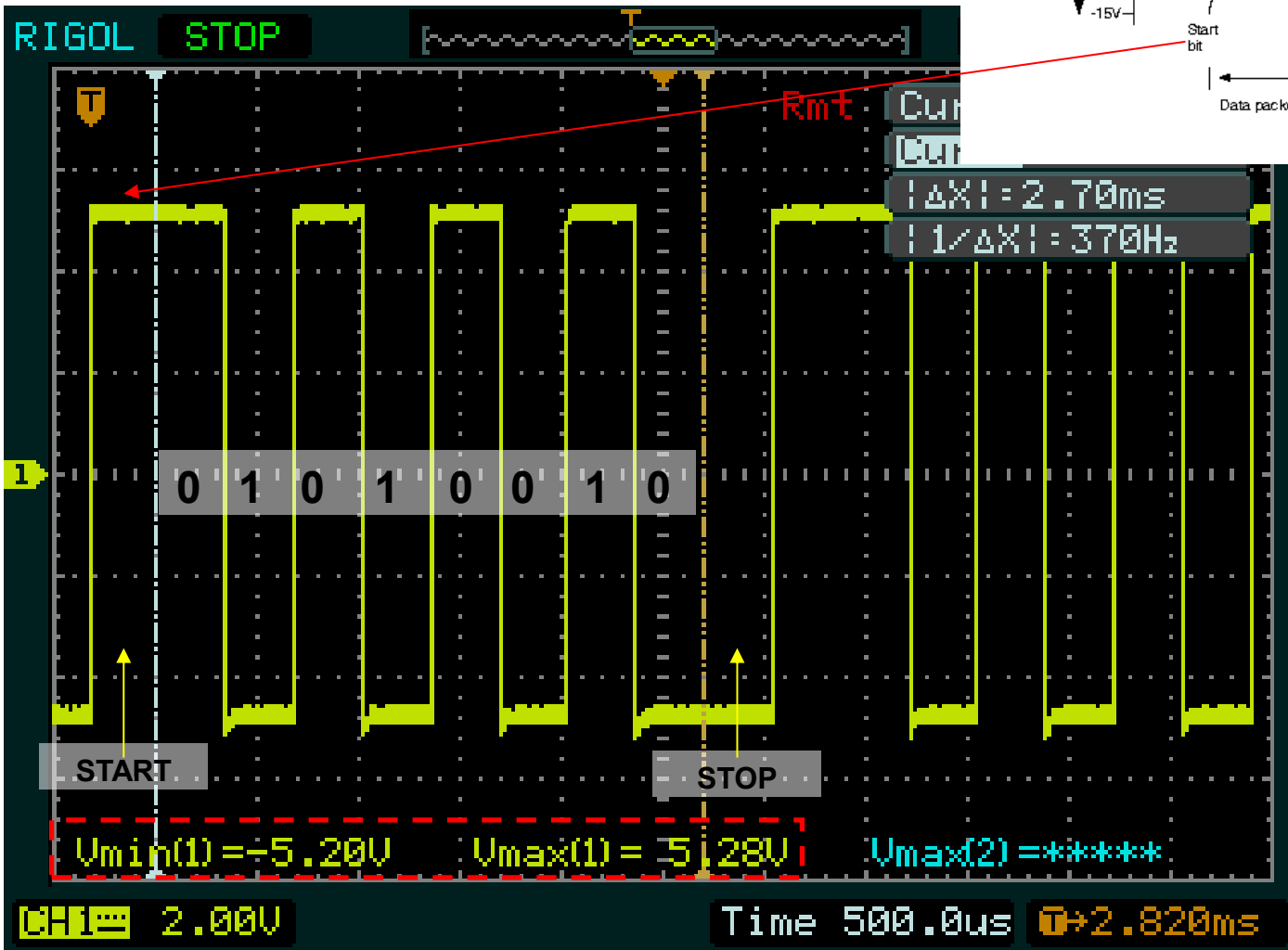
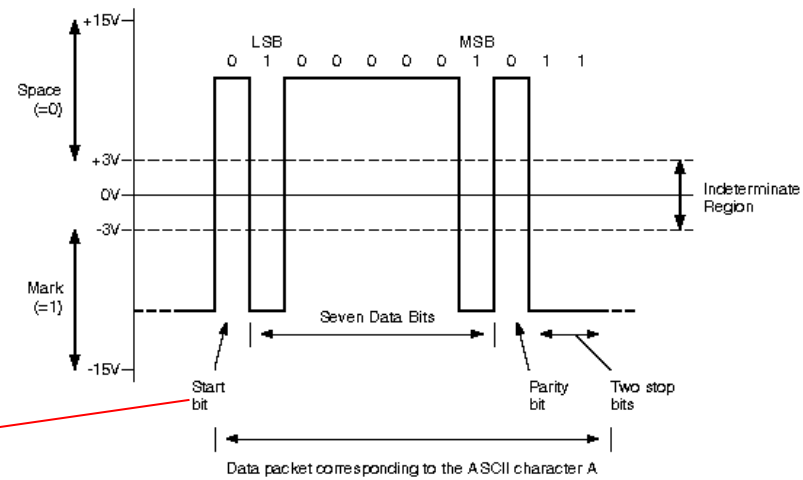
- (LV)TTL
- RS232 (pretvornik „MAX232“), 422, 485 (pretvornik „MAX485“)



□ Napetostna in logična nivoja

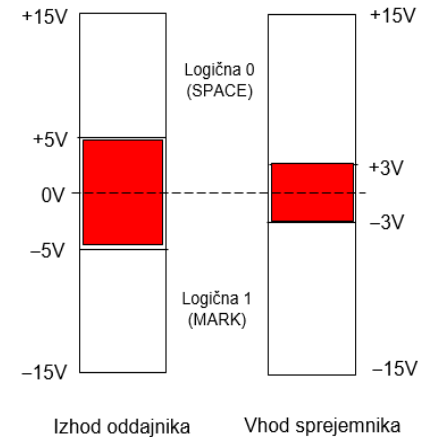


UART – osciloskop



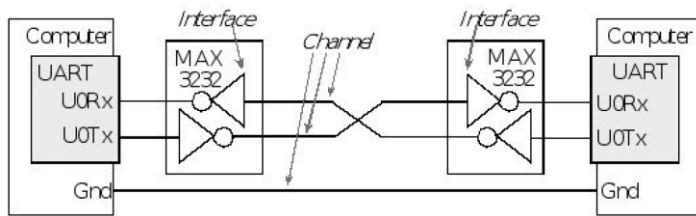
Električni del standarda RS232:

□ Napetostna in logična nivoja

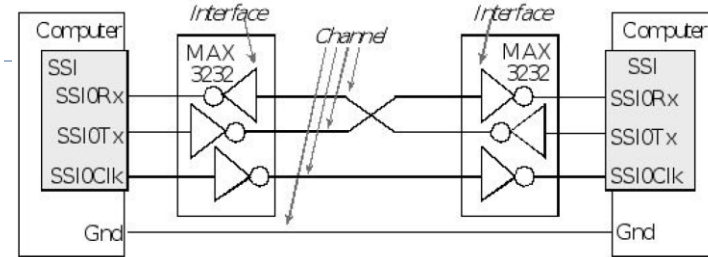


UART in različne komunikacije

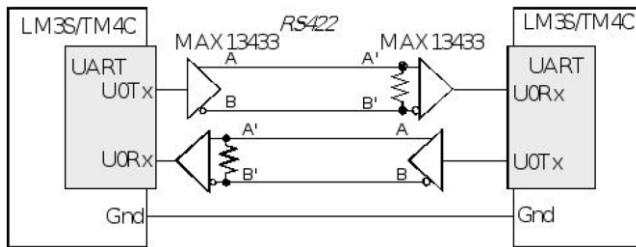
UART (RS232)



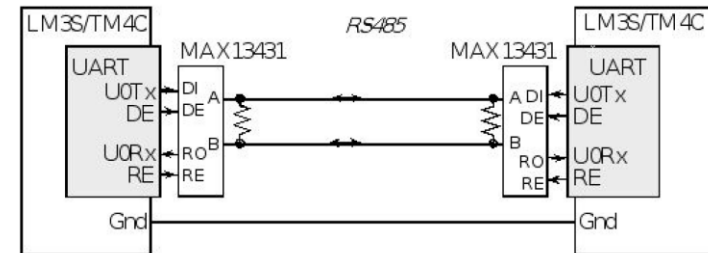
USART (RS232)



UART (RS422)



UART (RS485)



| | | +3.3V logic | +5V logic | RS232 level | RS422 level |
|-------|-------|-------------|-----------|-------------|--|
| True | Mark | +3V | +5V | TxD = -5.5V | (TxD ⁺ - TxD ⁻) = -3V |
| False | Space | +0.1V | +0.1V | TxD = +5.5V | (TxD ⁺ - TxD ⁻) = +3V |

Table 7.1. Typical voltage levels for the digital logic, RS232 and RS422 protocols.

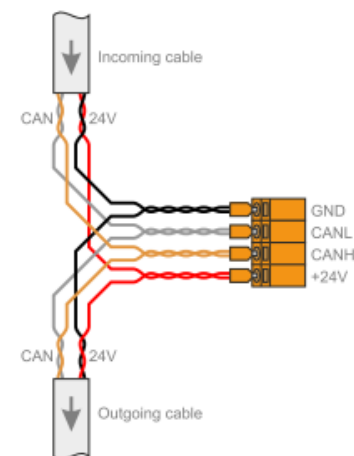
| Specification | RS232D | RS423A | RS422 | RS485 |
|-------------------------|--------------|--------------|--------------|--------------|
| Mode of operation | single-ended | single-ended | differential | differential |
| Drivers on one line | 1 | 1 | 1 | 32 |
| Receivers on one line | 1 | 10 | 10 | 32 |
| Maximum distance (feet) | 50 | 4,000 | 4,000 | 4,000 |
| Maximum data rate | 20 kb/s | 100 kb/s | 10 Mb/s | 10 Mb/s |

UART - Standardi

| Standard | Voltage signal | Max distance | Max speed | Number of devices supported per port |
|---------------|--|--------------|-----------|--------------------------------------|
| RS-232 | Single end (logic 1: +5 to +15V, logic 0: -5 to -15V) | 100 feet | 115Kbit/s | 1 master, 1 receiver |
| RS-422 | Differential (-6V to +6V) | 4000 feet | 10Mbit/s | 1 master, 10 receivers |
| RS-485 | Differential (-7V to +12V) | 4000 feet | 10Mbit/s | 32 masters, 32 receivers |

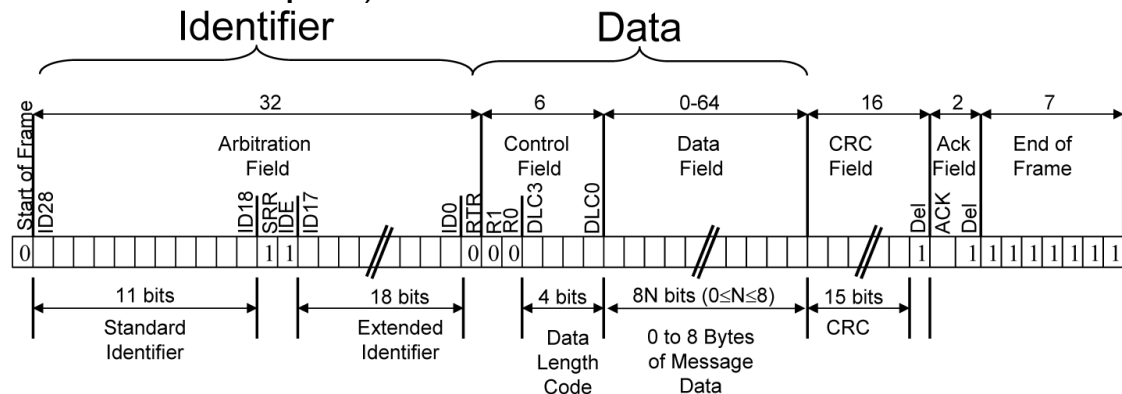
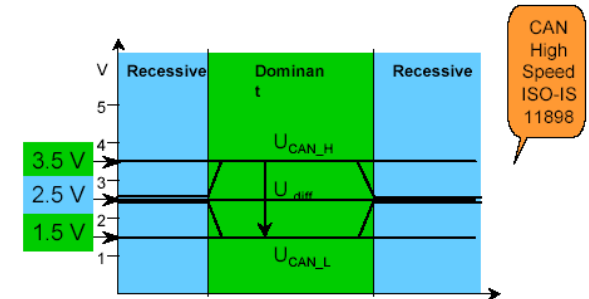
CANBus na kratko

- ▶ **CAN**bus – **C**ontroller **A**rea **N**etwork bus
 - ▶ serijsko vodilo za komunikacijo med vgrajenimi mikrokontrolerji
- ▶ **CAN**bus na kratko :
 - ▶ dve žici (CAN_H, CAN_L) + napajanje,
 - ▶ diferencialni prenos signala
 - ▶ odpornost na šum.
 - ▶ max 1 Mbit/s, 40m,
 - ▶ sporočila do 8 bajtov (latenca)
 - ▶ CAN-FD tudi daljša sporočila
- ▶ **CAN-FD** standard, ISO 11898-2:2016
 - ▶ 2Mbps, 5Mbps



CANbus na kratko

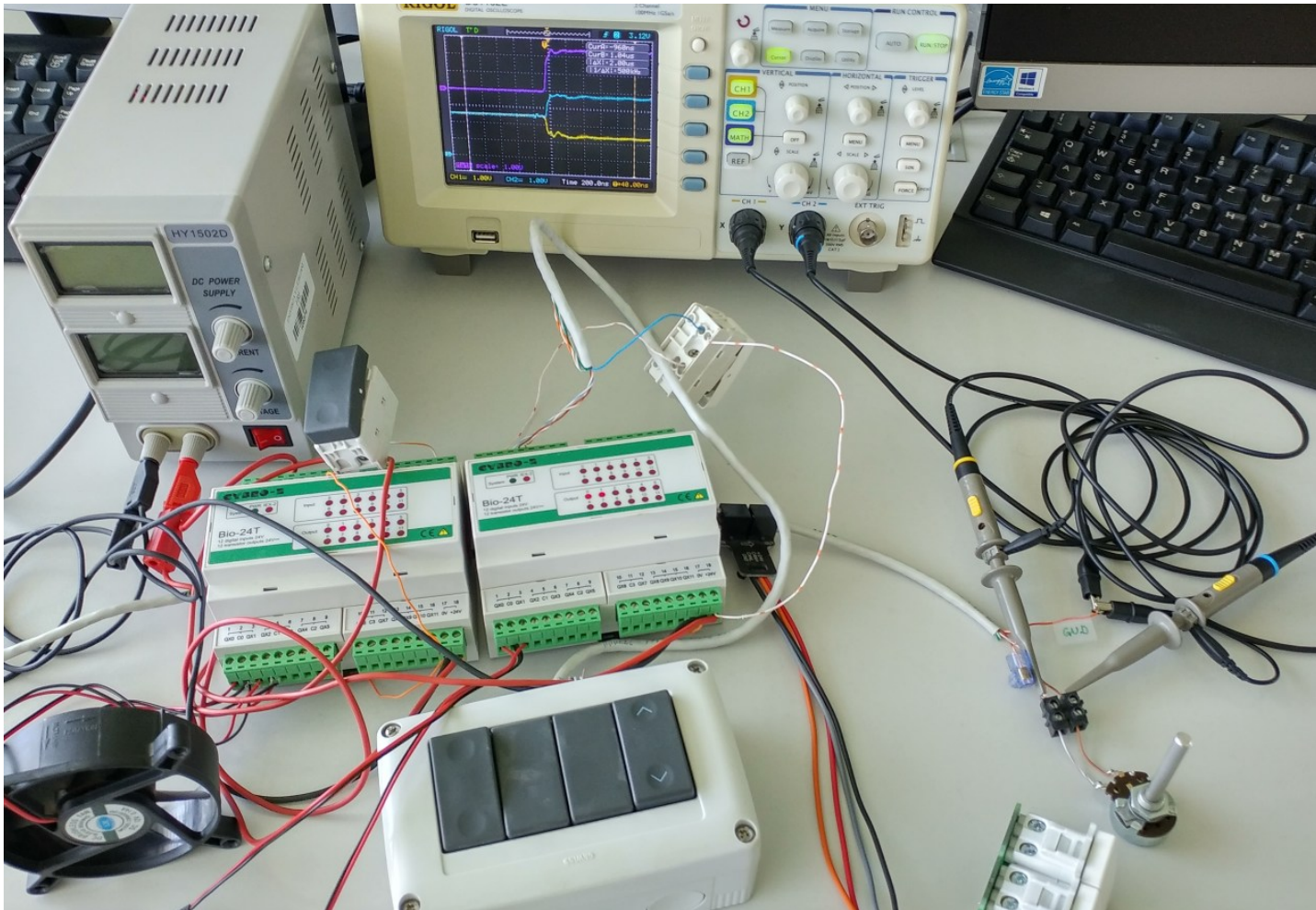
- ▶ **Diferencialni prenos** običajno na parici –
 - ▶ Non-Return To Zero (NRZ) in
 - ▶ bit-stuffing.
- ▶ Wired –AND povezava: vozlišče z logično 0 prevlada
 - ▶ 0 .. „dominant“, 1 .. „recessive“)
- ▶ Prenos podatkov
 - ▶ Protokol – sporočilno naravnan
 - ▶ Detekcija napake
 - ▶ Nivo Bitov (branje, „bit stuffing“)
 - ▶ Nivo sporočila (CRC, okvir, ACK napake)



CANbus na kratko

- ▶ Komunikacijski protokol:
 - ▶ **CSMA/CD**
 - ▶ **Carrier Sense Multiple Access/Collision Detection (Avoidance)**
 - ▶ z **NDA**
 - ▶ **Non-Destructive Arbitration**
- ▶ Sporočilno orientiran :
 - ▶ ni naslova, vsako sporočilo ima svojo **ID številko** (prioriteta, vsebina) – mora biti unikatna
 - ▶ potrjevanje (ACK)
 - ▶ se zazna potrditev vsaj enega sprejemalca

Lab. vaja - meritve



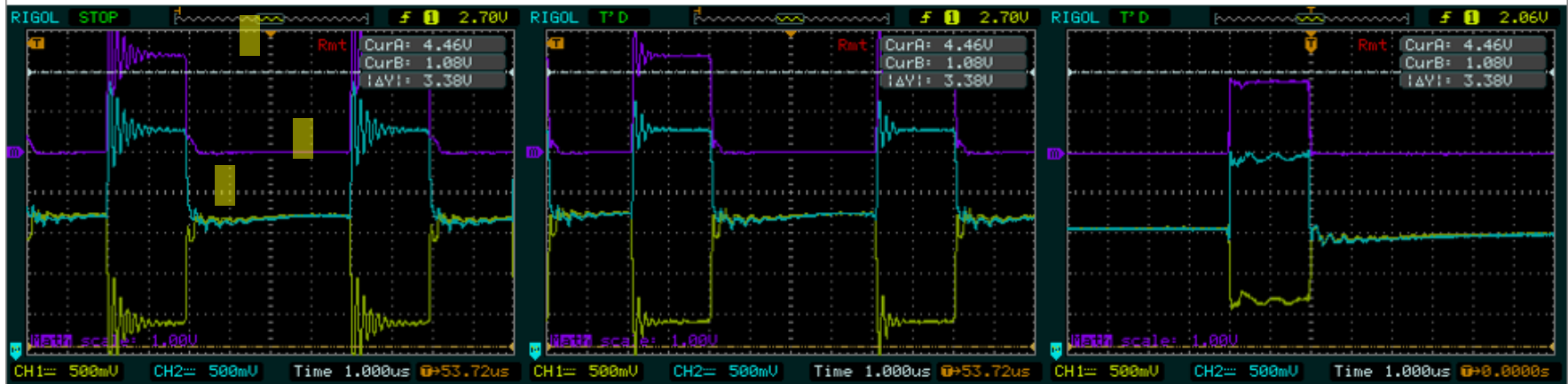
Lab. vaja – primerjava hitrosti

500kb/s:

Odprte sponke

500ohm

107ohm

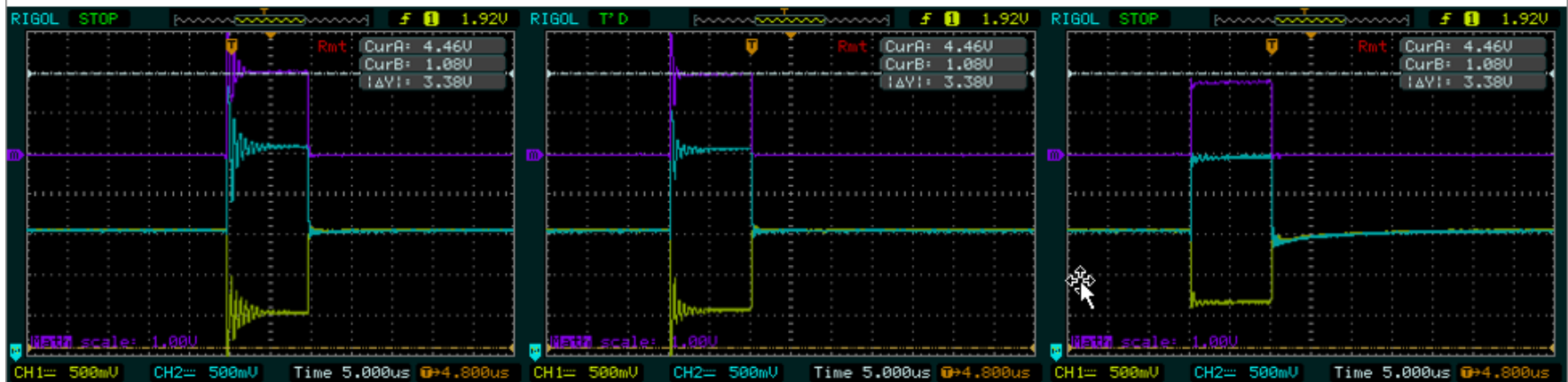


100kb/s:

Odprte sponke

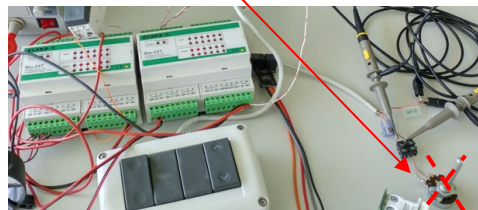
500ohm

107ohm

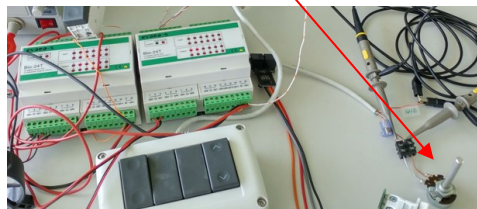


Lab. vaja – CANBUS (meritve)

Nezaključena linija

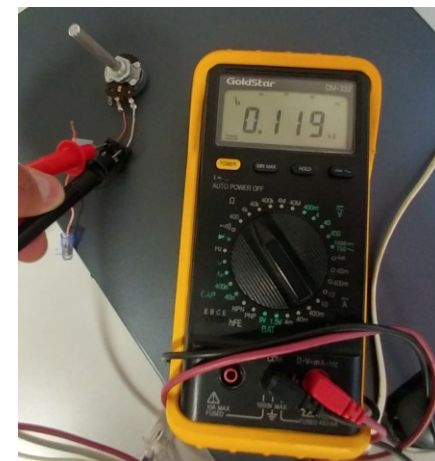


Zaključena linija



Ročna meritev

R_0

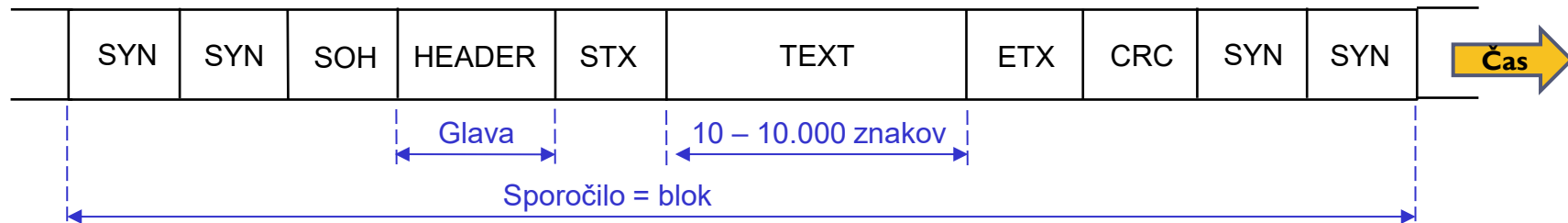


4.2 Sinhronski serijski prenos

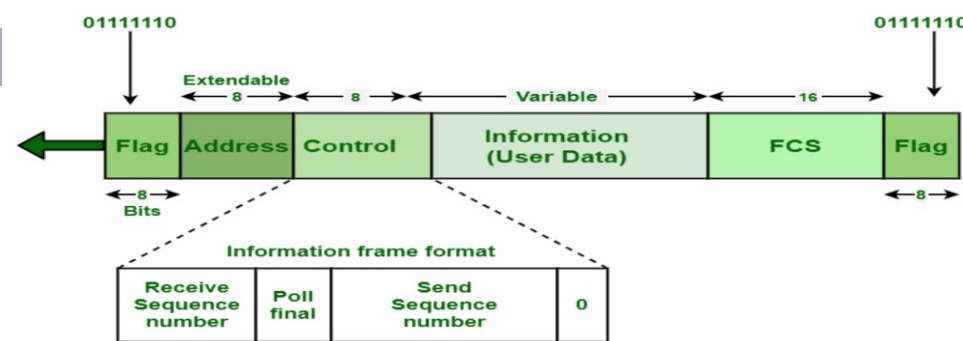
- ❑ Razvit je bil za namensko komunikacijo med računalniki.
- ❑ Enota pošiljanja ni več posamezen znak, temveč **sporočilo ali blok**.
- ❑ Sinhronizacija poteka na nivoju celotnega sporočila ali bloka.
 - ❑ se doseže na enega od dveh načinov (glede na hitrost):
 - ❑ počasnejše : dodana je povezava za **prenos urinega signala** (ang. Clock) – nižje hitrosti
 - ❑ hitre: brez „Clock“ povezave, oddajna in sprejemna ura se sinhronizirata iz prehodov v podatkovni povezavi
- ❑ Znotraj sporočila ali bloka ni presledkov, niz bitov je neprekinjen. Redundanca se v primerjavi z asinhronskim prenosom zmanjša in je samo nekaj procentov (%).
- ❑ Potrebno je pravilo za organizacijo informacij v bloku ⇒ **PROTOKOL**
- ❑ V uporabi sta dve vrsti protokolov:
 - Znakovno ali bajtno orientirani protokoli (starejši)
 - Bitno orientirani protokoli (novejši)

□ Znakovno ali bajtno orientirani protokoli

- BSC (Binary Synchronous Communication) je IBM-ov znakovno orientirani protokol, 1967. Dolgo časa je bil najbolj razširjen protokol.
- Uporaba abecede z definiranimi **posebnimi kontrolnimi znaki**.
 - STX – Start of text
 - SYN znak - vzpostavitev sinhronizacije med oddajnikom in sprejemnikom
 - ...



- Paritetni bit za kontrolo pravilnosti ne zadošča več.
- Oddajnik pri oddaji iz bitov besedila izračuna CRC (Cyclic Redundancy Check) bite (eden ali dva bajta) in jih doda v sporočilo.
 - Če sprejemnik s pomočjo CRC bitov ugotovi, da je bilo sporočilo pravilno sprejeto, pošlje znak za potrditev ACK (ang. Acknowledgemnt).
 - Če pa ugotovi, da je bilo sporočilo napačno sprejeto, pošlje nazaj znak NACK (ang. Negative Acknowledgemnt) in oddajnik prenos ponovi.



□ Bitno orientirani protokoli

- Sporočilo ali blok ne sestavljajo več znaki, temveč **pojujno zaporedje bitov**.
- Najbolj razširjena protokola sta
 - SDLC - Synchronous Data Link Control IBM 1970
 - HDLC – High-Level Data Link control ISO 1979, ki ima za osnovo SDLC
- HDLC je pravzaprav standardizirana skupina protokolov, vsem pa je skupno pravilo, da v zaporedju bitov ne sme biti več kot 5 zaporednih enic (1).
- Na ta način ločimo vsebino sporočila od bajtov, ki določajo začetek in konec sporočila – zaporedju šestih enic sledi ničla (Flag bajt $\equiv 01111110$).
- Oddajnik v sporočilu po vsakih petih zaporednih enkah vstavi 0 (ang. bit stuffing), sprejemnik pa te ničle izloča.
 - Pri NRZI kodiranju (0 – sprememba; 1 – brez spremembe) je običajno zagotovljena najmanj ena sprememba nivoja na vsakih 6 bitov \Rightarrow sinhronizacija.
 - Novejši protokoli pa za uspešno sinhronizacijo večinoma uporabljajo kodiranja 8b/10b, ...

USART: Primer STM32F4 Discovery

□ Sinhronski serijski prenos

Figure 307. USART example of synchronous transmission

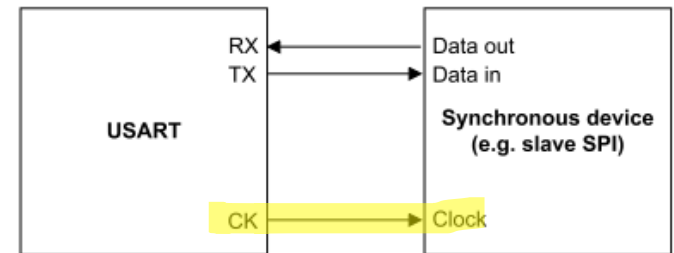


Figure 308. USART data clock timing diagram (M=0)

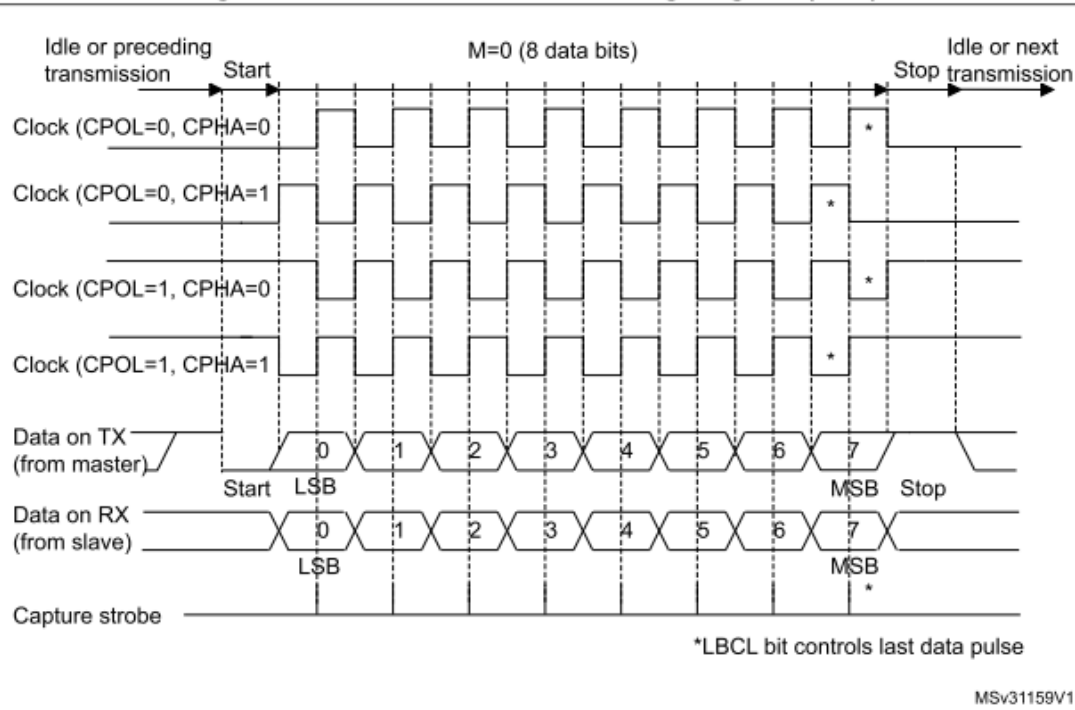
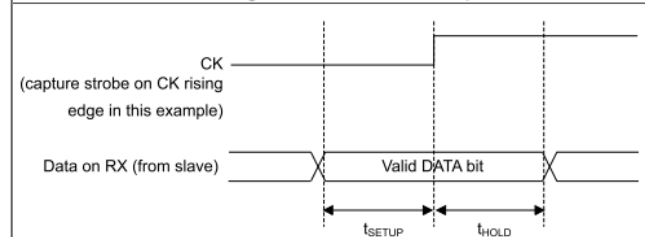
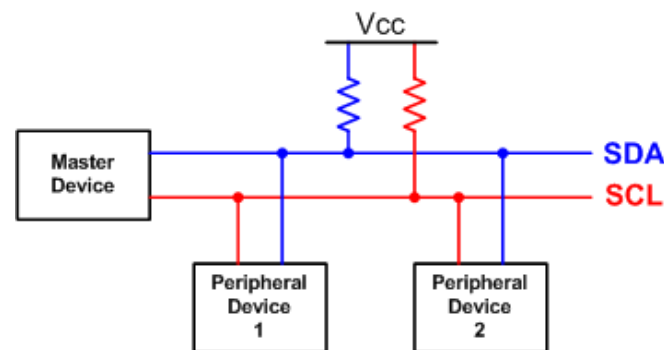


Figure 310. RX data setup/hold time



4.2.1 Vodilo I2C, IIC, I²C (Inter-Integrated Circuit)

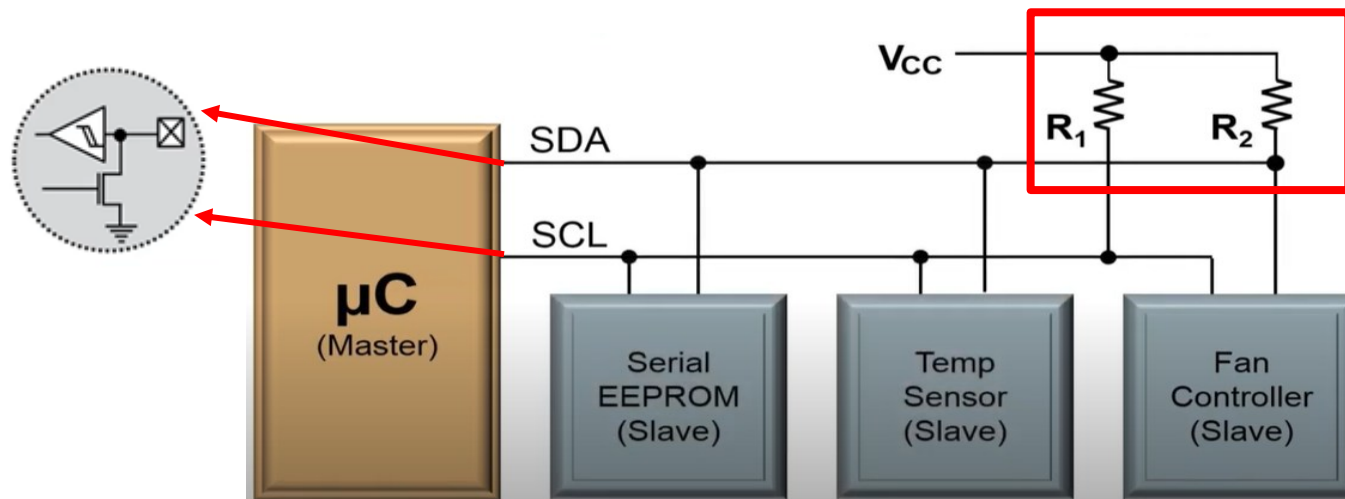
- ❑ **1982** razvito kot interno vodilo za povezavo med čipi proizvajalca **Philips (NXP)**
- ❑ 2 liniji za prenos podatkov med dvema napravama (Master in Slave):
 - **SDA (serial data)** – linija za pošiljanje in sprejemanje podatkov
 - **SCL (serial clock)** - urin signal, ki ga vedno pošilja glavna naprava
- ▶ 3 hitrosti :
 - ▶ Standard: do 100 kbit/s
 - ▶ Fast: do 400 kbit/s
 - ▶ High: do 3.4 Mbit/s
- ▶ Prilagoditev vrednosti pull-up uporov
 - ▶ Višja hitrost, manjša upornost



Video: Microchip I2C: <https://www.youtube.com/watch?v=qTLRRg6Mee0>

□ Primer povezave mikrokrmilnika (Master) s tremi V/I napravami (Slave)

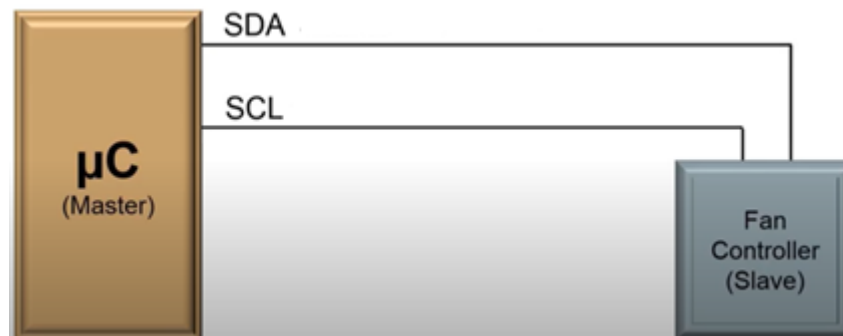
- Vsaka naprava Slave ima svoj naslov
- Za pravilno delovanje sta pri povezavi naprav na vodilo I2C upora R_1 in R_2 ('pull-up').



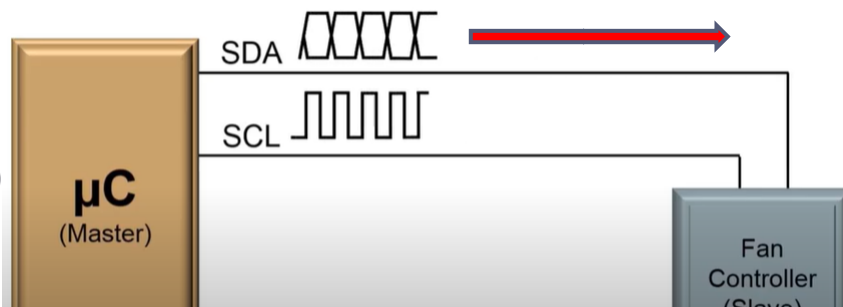
- Protokol I2C deluje na predpostavki, da sta liniji SDA in SCL 'open drain' ali 'open collector'
 - Katerakoli naprava zagotovi nizek nivo (low) na linijah, ne more pa visokega (high).
 - Vsaka linija ima zato dodan upor, da privzeto ohranja visok nivo.
 - To omogoča, da ne pride do kratkega stika, če ena naprava poskuša dvigniti linijo na visok nivo, druga pa na nizek nivo.

□ Povezava dveh naprav:

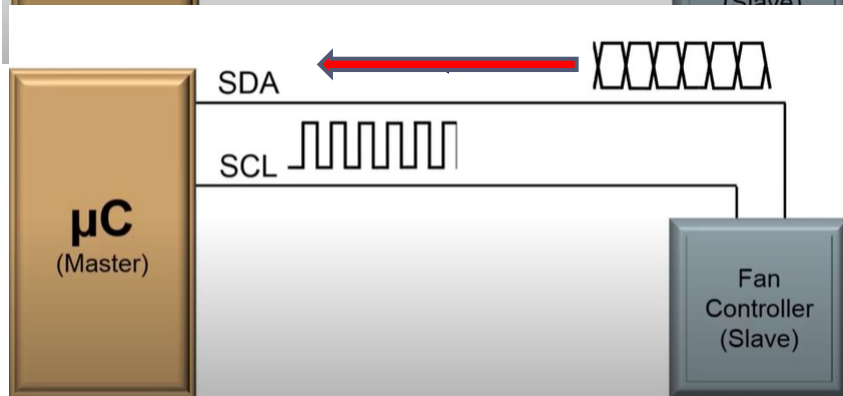
- Mikrokrmilnik (Master)
- Krmiljenje ventilatorja (Slave)



- Master pošilja urin signal (SCL)
- Master zapisuje na napravo Slave (SDA)

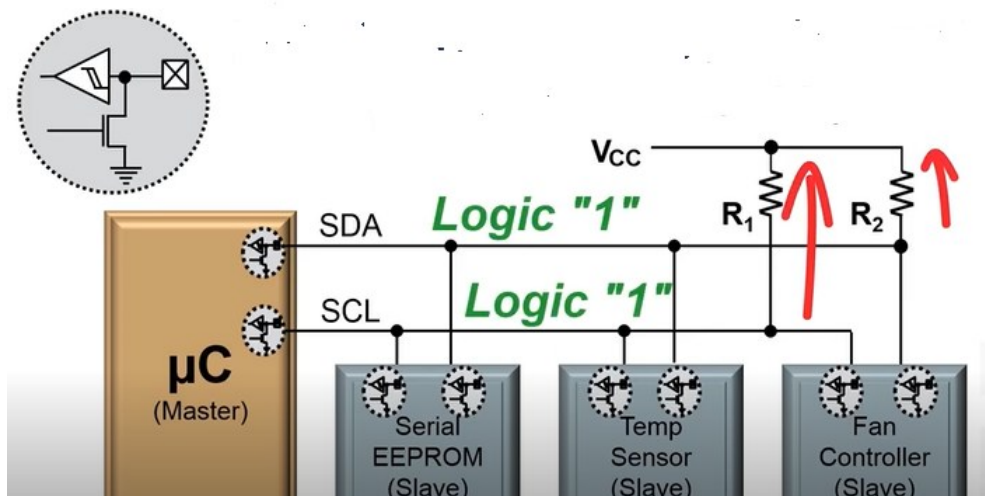


- Master pošilja urin signal (SCL)
- Master **bere iz** naprave Slave

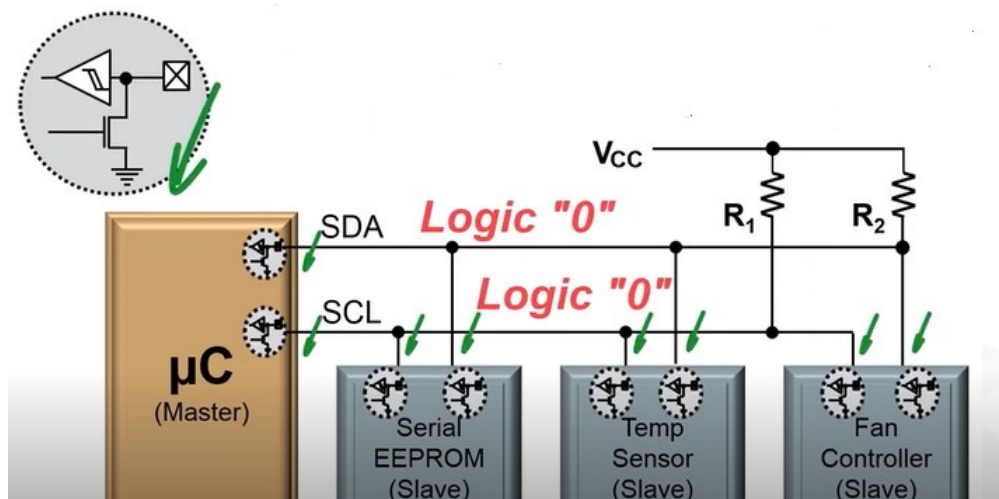


□ Logični nivoji:

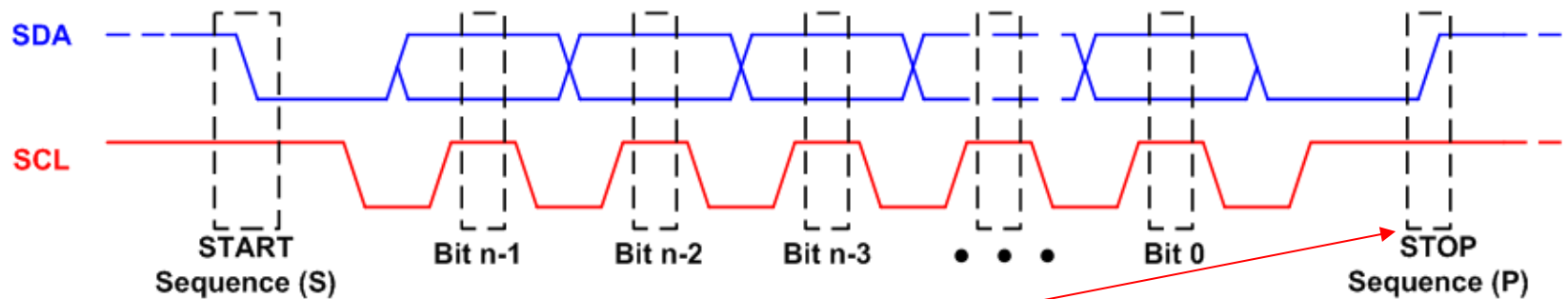
- Logična 1



- Logična 0



□ Signali na linijah



- Dodatna bita (vedno generira master) :
 - pogoj **start** – linija SDA preklopi iz 1 v 0 preden linija SCL preklopi iz 1 v 0
 - pogoj **stop** - linija SDA preklopi iz 0 v 1 potem ko linija SCL preklopi iz 0 v 1
- SCL
 - generira master (sinhronizacija pri več masterjih)
 - Slave lahko upočasnjuje („clock stretching“)
- SDA
 - se lahko spremeni le, ko je SCL v nizkem stanju
 - mora biti konstanten, ko SCL v visokem stanju
 - arbitraža pri več masterjih

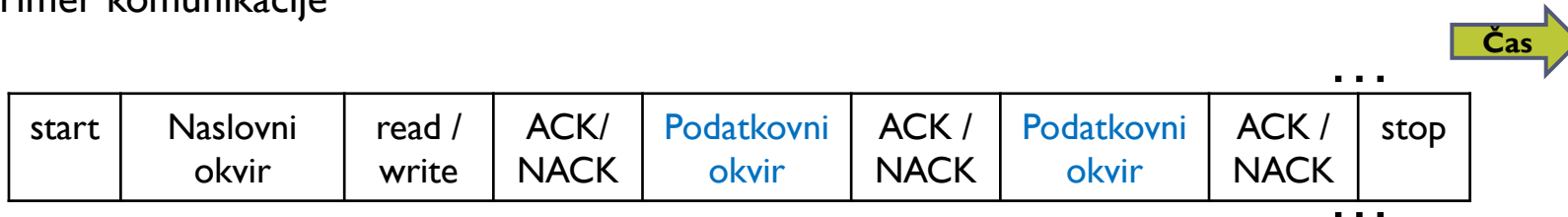
4.2.1 Vodilo I2C, IIC, I²C (Inter-Integrated Circuit)

- ❑ Sinhronski serijski prenos, dvosmerni izmenični (ang. half duplex)
- ❑ Open drain izhodi, pull-up upori -> „wired-and“ vezava ->
- ❑ Multi master z arbitražo in sinhronizacijo ure

Podatki se vedno pošiljajo v **sporočilih** (ang. messages), ki so razdeljena v **okvirje** (ang. frames) podatkov in dodatne bite:

- **Naslovni okvir** (ang. address frame) – naslov naprave
- **Podatkovni okvir** (ang. data frame) – podatki, ki se prenašajo.

- ❑ Primer komunikacije



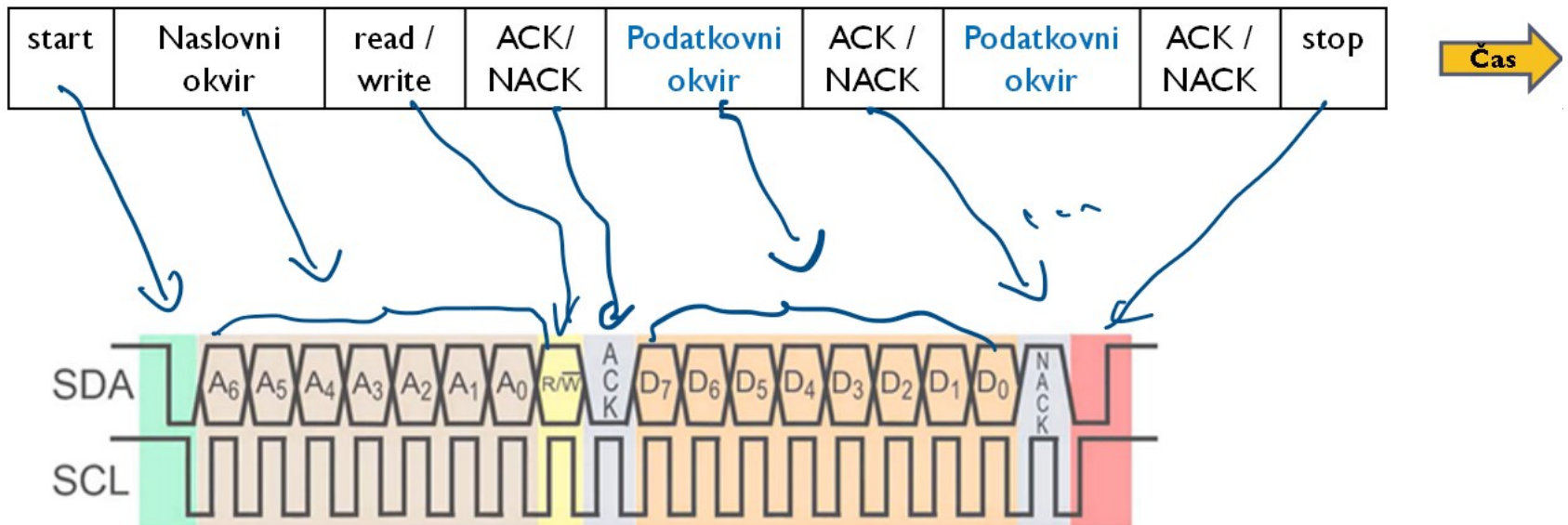
4.2.1 Vodilo I2C, IIC, I²C (Inter-Integrated Circuit)

- Podatki se vedno pošiljajo v **sporočilih** (ang. messages), ki so razdeljena v **okvirje** (ang. frames) podatkov in dodatne bite:
 - Naslovni okvir** (ang. address frame) – naslov naprave
 - Podatkovni okvir** (ang. data frame) – podatki, ki se prenašajo.

Primer komunikacije



- Dodatni biti:
 - read/write** – en bit določa prenos iz 'master' v 'slave' napravo (0) ali 'master' zahteva podatek iz 'slave' naprave (1).
 - ACK/NACK** – vsak okvir sporočila ima bit 'acknowledge/noacknowledge'. Če je bil naslovni ali podatkovni okvir uspešno prejet je pošiljatelju vrnjen bit ACK, sicer NACK.



Dodatni biti:

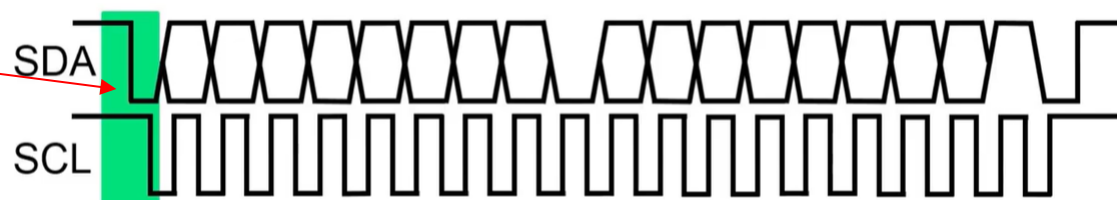
- ▶ **read/write** – en bit določa prenos iz ‘master’ v ‘slave’ napravo (0) ali ‘master’ zahteva podatek iz ‘slave’ naprave (1).
- ▶ **ACK/NACK** – vsak okvir sporočila ima bit ‘acknowledge/noacknowledge’. Če je bil naslovni ali podatkovni okvir uspešno prejet je pošiljatelju vrnjen bit ACK, sicer NACK.
- pogoj **start** – linija SDA preklopi iz 1 v 0 preden linija SCL preklopi iz 1 v 0
- pogoj **stop** - linija SDA preklopi iz 0 v 1 potem ko linija SCL preklopi iz 0 v 1

Tipičen potek I2C komunikacije :

I2C signal v celoti:

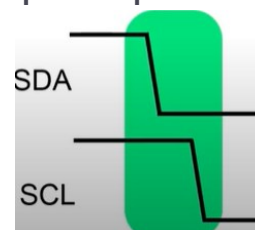


□ **START**

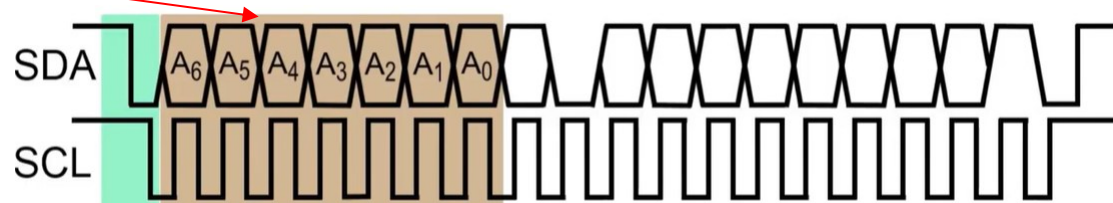


- Linija SDA preklopi iz 1 v 0 predno linija SCL preklopi iz 1 v 0

Idle – SDA = SCL - High
Master SDA – High v Low
Master SCL – drži High



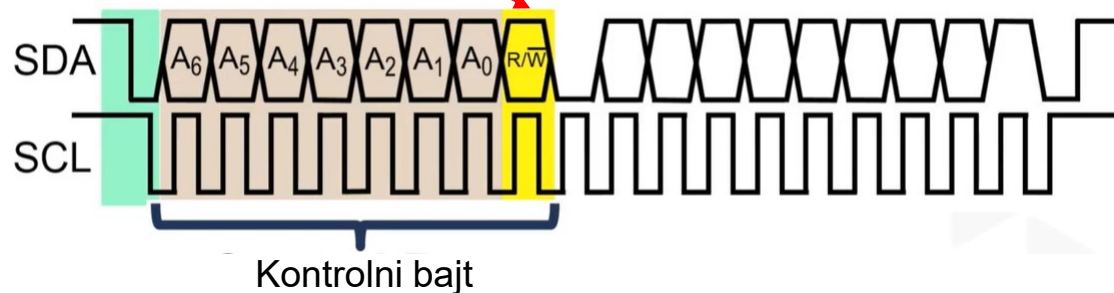
□ **Naslov naprave** Slave je 7-biten (najpogostejša dolžina)



Tipičen potek I2C komunikacije :

□ Branje/pisanje – R/\bar{W} (kontrolni bajt: $A_6 \dots A_0 R/\bar{W}$)

- $R/\bar{W} = \text{High}$ (Master zahteva branje podatkov iz Slave)
- $R/\bar{W} = \text{Low}$ (Master bo pisal na Slave)



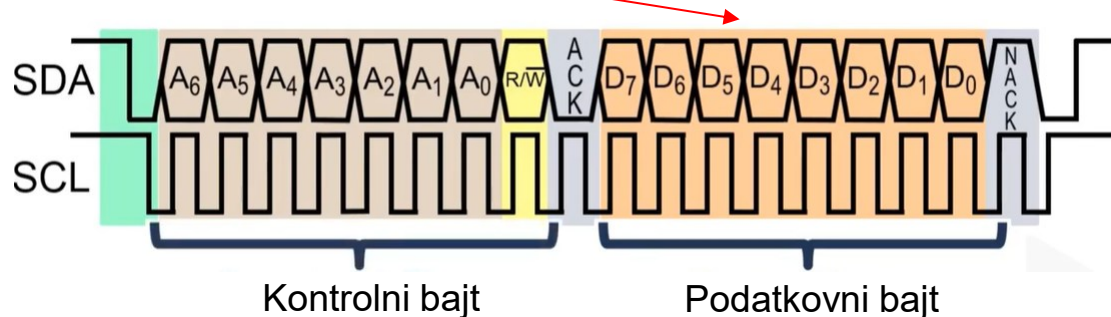
- kontrolni bajt: $A_6 \dots A_0 R/\bar{W}$

□ Potrditev (Acknowledge) – pojavi se na vsakem 9 urinem ciklu

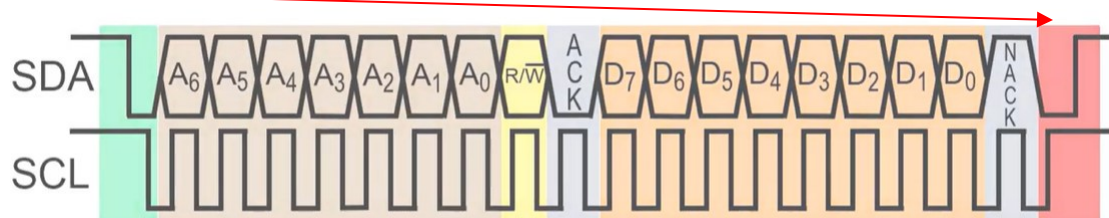


Tipičen potek I2C komunikacije :

- Podatkovni bajt ($D_7 \dots D_0$)



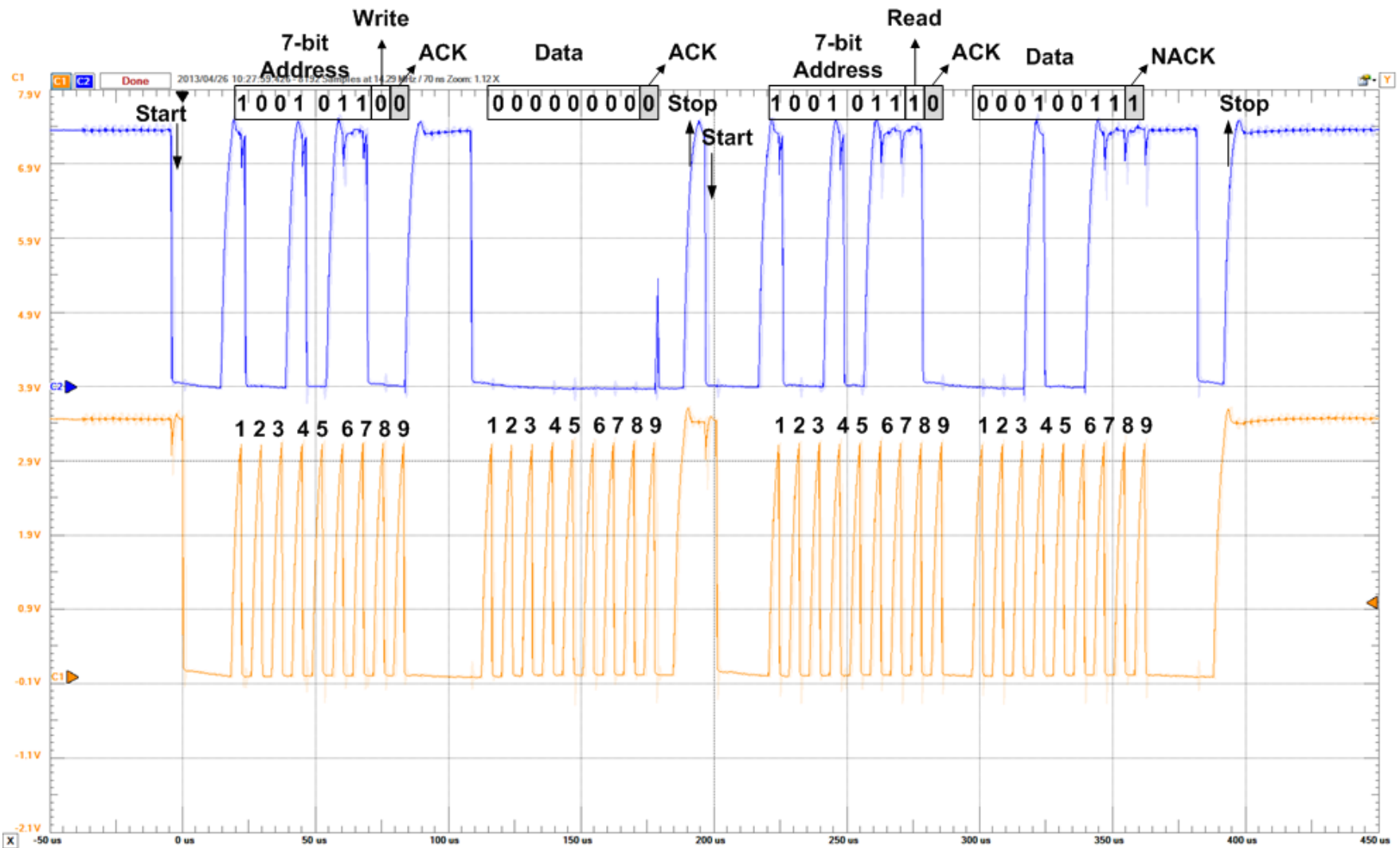
- STOP



- Linija SDA preklopi iz 0 v 1 potem ko linija SCL preklopi iz 0 v 1
Master SDA – Low v High
Master SCL – drži High

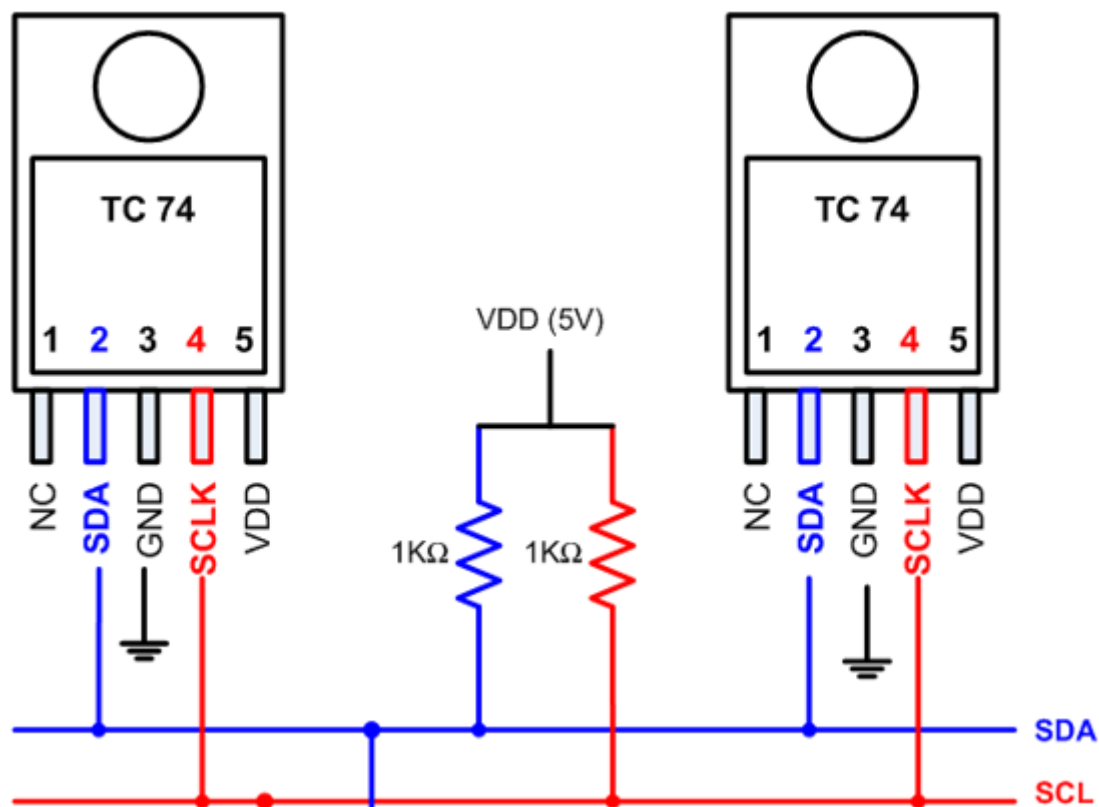


I2C – električno (osciloskop)



Primeri I2C povezav:

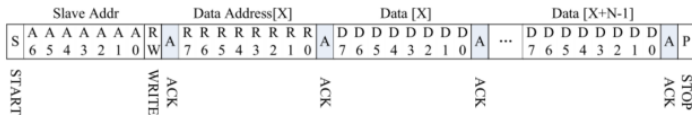
□ Digitalno temperaturno tipalo



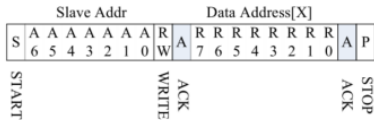
Primer I2C komunikacije STM32H7 - Touch

1.2 I²C Read/Write Interface description

Write N bytes to I2C slave



Set Data Address



Read X bytes from I²C Slave

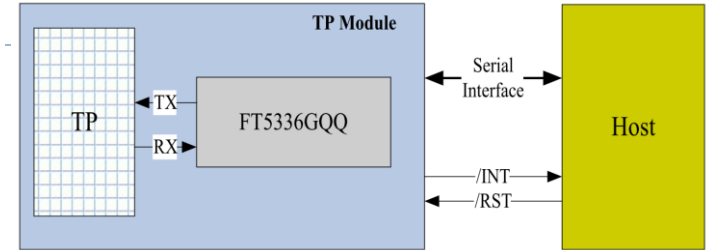
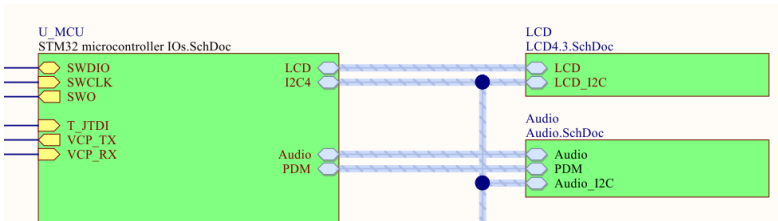


Figure 2-3 Host Interface Diagram

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Touch_Demo

8-bitni naslovi in registri

Work Mode Register Map

| Address | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Host Access | |
|---------|---------------|---------------------------------------|------|------|--|-----------------------------|------|------|------|-------------|---|
| 00h | DEVIDE_MODE | Device Mode[2:0] | | | | | | | | RW | |
| 01h | GEST_ID | Gesture ID[7:0] | | | | | | | | | R |
| 02h | TD_STATUS | | | | | Number of touch points[3:0] | | | | R | |
| 03h | TOUCH1_XH | 1 st Event Flag | | | 1 st Touch X Position[11:8] | | | | | R | |
| 04h | TOUCH1_XL | 1 st Touch X Position[7:0] | | | | | | | | | R |
| 05h | TOUCH1_YH | 1 st Touch ID[3:0] | | | 1 st Touch Y Position[11:8] | | | | | | R |
| 06h | TOUCH1_YL | 1 st Touch Y Position[7:0] | | | | | | | | | R |
| A8h | ID_G_FT5201ID | CTPM Vendor ID | | | | | | | | | R |

Primer I2C komunikacije STM32H7 – Touch (programsko)

```
// Reading from address 0x38 register Vendor's Chip ID (addr. 0xA8) default value should be 0x51=81
```

```
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0xA8, I2C_MEMADD_SIZE_8BIT, &VendorID, 1, HAL_MAX_DELAY);

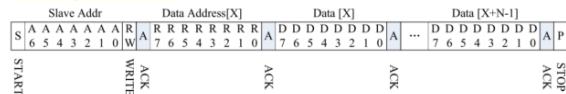
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x00, I2C_MEMADD_SIZE_8BIT, &DeviceMode, 1, HAL_MAX_DELAY);
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x01, I2C_MEMADD_SIZE_8BIT, &Gesture, 1, HAL_MAX_DELAY);
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x02, I2C_MEMADD_SIZE_8BIT, &Status, 1, HAL_MAX_DELAY);

retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x03, I2C_MEMADD_SIZE_8BIT, &dataBuffer, 5, HAL_MAX_DELAY);
if (Status != 0) {
    TouchX = ((dataBuffer[0] & 0b1111) << 8) + dataBuffer[1];
    TouchY = ((dataBuffer[2] & 0b1111) << 8) + dataBuffer[3];
} else {
    TouchX = 0;
    TouchY = 0;
}
```

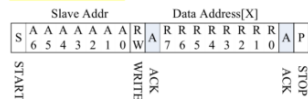
8-bitni naslovi in registri

1.2 I²C Read/Write Interface description

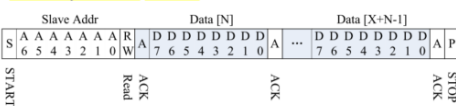
Write N bytes to I2C slave



Set Data Address



Read X bytes from I²C Slave



Work Mode Register Map

| Address | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Host Access |
|---------|---------------|---------------------------------------|------|--|--|------|------|------|------|-------------|
| 00h | DEVIDE_MODE | Device Mode[2:0] | | | | | | | | RW |
| 01h | GEST_ID | Gesture ID[7:0] | | | | | | | | R |
| 02h | TD_STATUS | | | | Number of touch points[3:0] | | | | | R |
| 03h | TOUCH1_XH | 1 st Event Flag | | 1 st Touch X Position[11:8] | | | | | | R |
| 04h | TOUCH1_XL | 1 st Touch X Position[7:0] | | | | | | | | R |
| 05h | TOUCH1_YH | 1 st Touch ID[3:0] | | | 1 st Touch Y Position[11:8] | | | | | R |
| 06h | TOUCH1_YL | 1 st Touch Y Position[7:0] | | | | | | | | R |
| A8h | ID_G_FT520IID | CTPM Vendor ID | | | | | | | | R |

https://github.com/LAPSYLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Touch_Demo

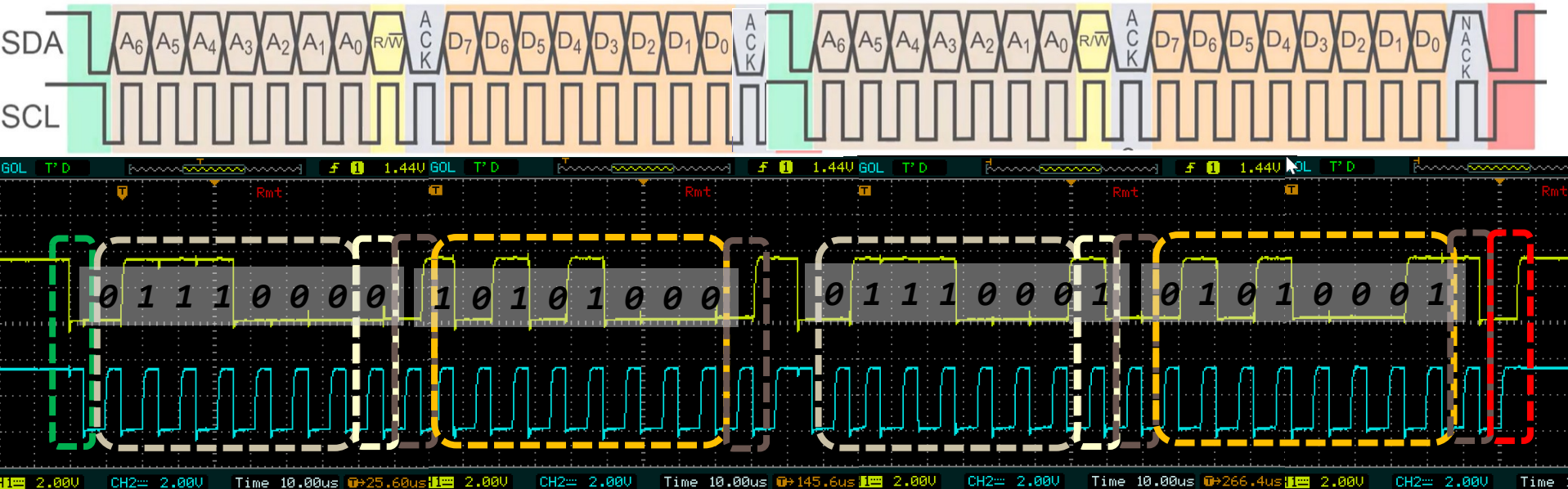
Primer I2C komunikacije

STM32H7 - Touch

I2C branje

```
// Reading from address 0x38 register Vendor's Chip ID (addr. 0xA8) default value should be 0x51=81
```

```
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0xA8, I2C_MEMADD_SIZE_8BIT,&dataBuffer[5], 1, HAL_MAX_DELAY);
```



Set Data Address

| Slave Addr | | | | | | | | Data Address[X] | | | | | | | | | | |
|------------|---|---|---|---|---|---|-------|-----------------|---|---|---|---|---|---|---|---|-----|------|
| S | A | A | A | A | A | A | A | R | R | R | R | R | R | R | R | | | |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 | W | A | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | A | P |
| START | | | | | | | WRITE | | | | | | | | | | ACK | STOP |

Read X byte I2C Slave

| Slave Addr | | | | | | | | Data [N] | | | | | | | | Data [X+N-1] | | | | | | | | | | | | |
|------------|---|---|---|---|---|---|------|----------|---|---|---|---|---|---|---|--------------|-----|-----|---|---|---|---|---|---|---|---|---|------|
| S | A | A | A | A | A | A | A | R | A | D | D | D | D | D | D | D | D | ... | D | D | D | D | D | D | D | | | |
| 6 | 5 | 4 | 3 | 2 | 1 | 0 | W | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | A | ... | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | A | P | |
| START | | | | | | | Read | | | | | | | | | | ACK | | | | | | | | | | | STOP |

https://github.com/LAPSYLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_Demo

Primer I2C komunikacije STM32H7 - Audio

Multi-channel Audio Hub CODEC for Smartphones

The sequence of signals associated with a single register write operation is illustrated in Figure 72.

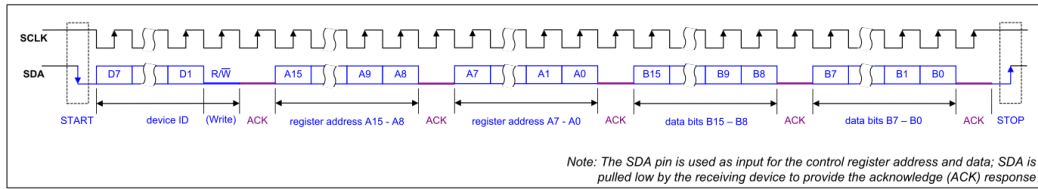


Figure 72 Control Interface 2-wire (I2C) Register Write

The sequence of signals associated with a single register read operation is illustrated in Figure 73.

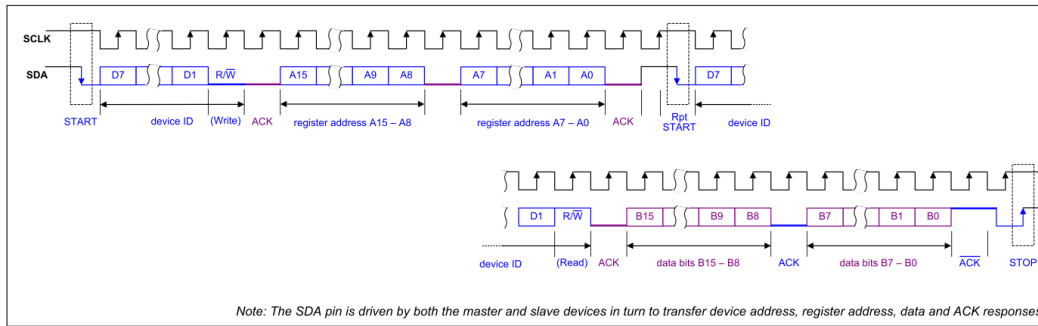
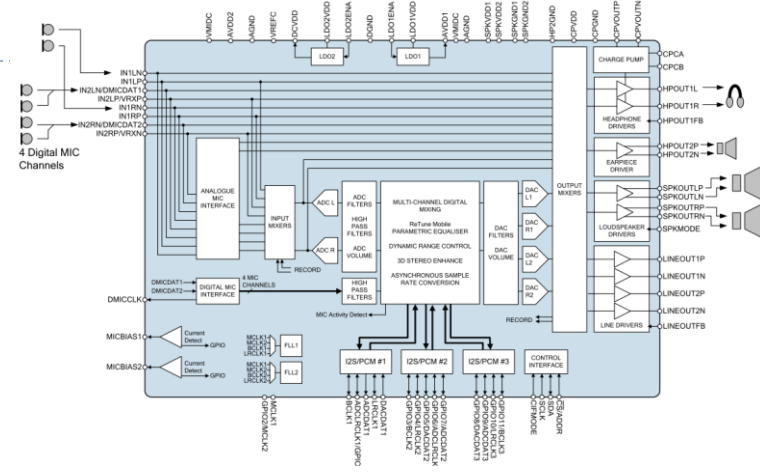


Figure 73 Control Interface 2-wire (I2C) Register Read



https://github.com/LAPSYLAB/STM32H7_Discovery_VIN_Projcts/tree/main/STM32H750B-DK_I2C_Basic_Demo

16-bitni naslovi in registri

REGISTER MAP

The WM8994 control registers are listed below. Note that only the register addresses described here should be accessed; writing to other addresses may result in undefined behaviour. Register bits that are not documented should not be changed from the default values.

| REG | NAME | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | DEFAULT |
|---------|----------------------|-----------------|-----------|--------------|--------------|-------------|----|--------------|--------------|----------|----------|-----------|-----------|---|----------------|---|----------|---------|
| R0 (0h) | Software Reset | SW_RESET [15:0] | | | | | | | | | | | | | | | | 0000h |
| R1 (1h) | Power Management (1) | 0 | 0 | SPKO_UTL_ENA | SPKO_UTL_ENA | HPOU_T2_ENA | 0 | HPOU_T1L_ENA | HPOU_T1R_ENA | 0 | 0 | MICB2_ENA | MICB1_ENA | 0 | VMID_SEL [1:0] | | BIAS_ENA | 0000h |
| R2 (2h) | Power Management (2) | 0 | TSHUT_ENA | TSHUT_OPDIS | 0 | OPCLK_ENA | 0 | MIXINL_ENA | MIXINR_ENA | IN2L_ENA | IN1L_ENA | IN2R_ENA | IN1R_ENA | 0 | 0 | 0 | 0 | 6000h |

Primer I2C komunikacije STM32H7 - Audio

Multi-channel Audio Hub CODEC for Smartphones

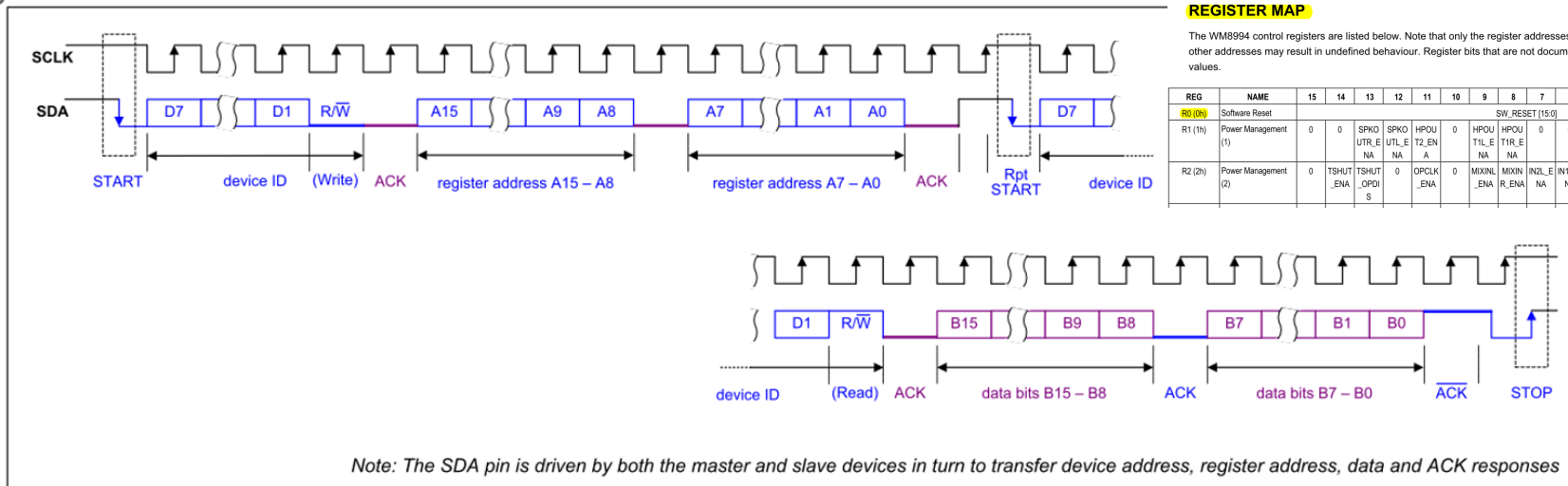
16-bitni naslovi in registri

```
// Reading from device address 0x1a register R0 (addr. 0x00) default value should be 0x8994
dataBuffer[0] = 0; dataBuffer[1] = 0x00;
retval = HAL_I2C_Master_Transmit(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);
```

```
retval = HAL_I2C_Master_Receive(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);
```

```
snprintf(SendBuffer, BUFSIZE, "Hello World [%d]: Key:%d Reg.value1:0x%\n\r", Counter++, KeyState,
dataBuffer[0]*256+dataBuffer[1]);
```

```
HAL_UART_Transmit(&huart3, SendBuffer, strlen(SendBuffer), 100);
```



Note: The SDA pin is driven by both the master and slave devices in turn to transfer device address, register address, data and ACK responses

Figure 73 Control Interface 2-wire (I2C) Register Read

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_Demo

Primer I2C komunikacije – STM32F4

```

/* USER CODE BEGIN PV */
HAL_StatusTypeDef retval;

uint8_t ChipID;

/* USER CODE END PV */
MX_I2C1_Init();

/* USER CODE BEGIN 2 */

```

```

// Set Reset Line to 1 (switch device on)

```

```

HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);

```

```

HAL_Delay(1000); // recomended by datasheet

```

```

// From Device with address=0x94, Read register with address 0x01 and put value in ChipID

```

```

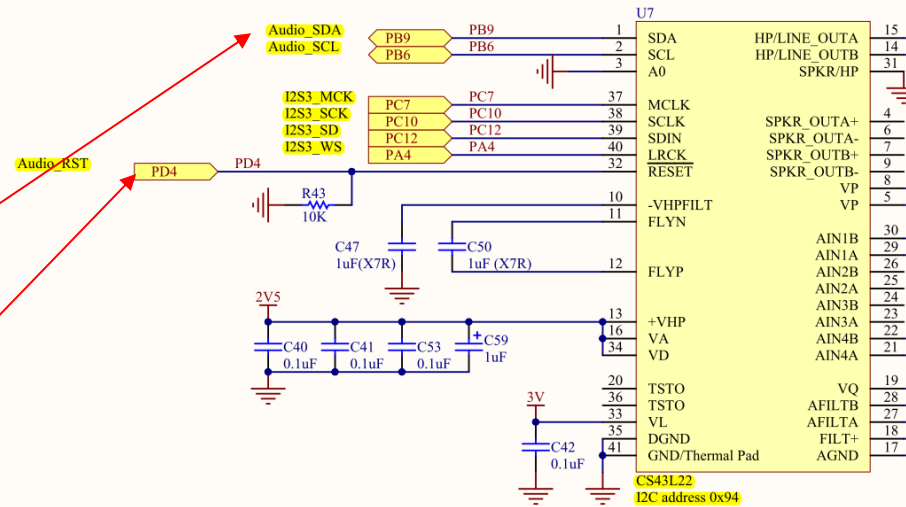
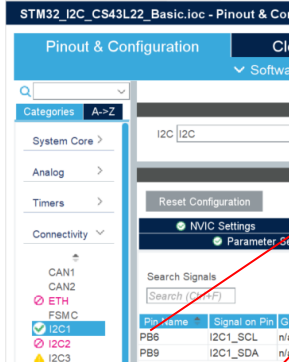
// DevAddress_0x94, tMemAddress=0x01, MemAddSize=8b, *pData,Size, Timeout);

```

```

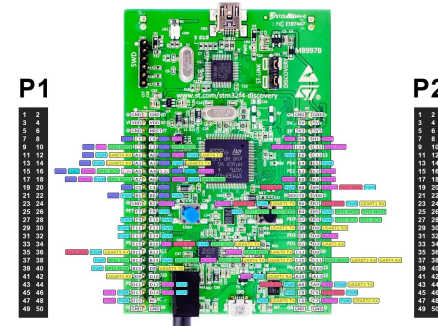
retval = HAL_I2C_Mem_Read(&hi2c1, 0x94, 0x01, I2C_MEMADD_SIZE_8BIT, &ChipID, 1, 1000);

```



CS43L22

Low Power, Stereo DAC w/Headphone & Speaker Amps



7.1 Chip I.D. and Revision Register (Address 01h) (Read Only)

| | | | | | | | |
|---------|---------|---------|---------|---------|--------|--------|--------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHIPID4 | CHIPID3 | CHIPID2 | CHIPID1 | CHIPID0 | REVID2 | REVID1 | REVID0 |

7.1.1 Chip I.D. (Read Only)

I.D. code for the CS43L22.

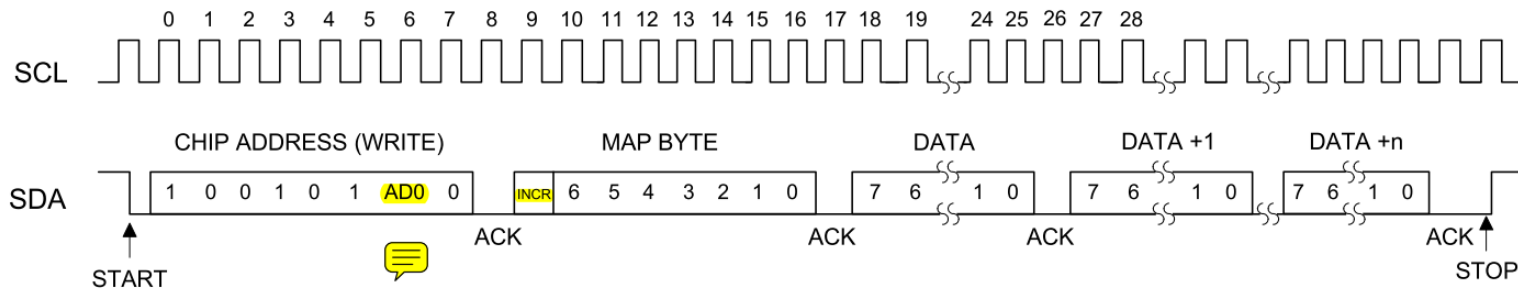
| | |
|-------------|---------|
| CHIPID[4:0] | Device |
| 11100 | CS43L22 |

7.1.2 Chip Revision (Read Only)

CS43L22 revision level.

| | |
|------------|----------------|
| REVID[2:0] | Revision Level |
| 000 | A0 |
| 001 | A1 |
| 010 | B0 |
| 011 | B1 |

The upper 6 bits of the address field are fixed at 100101. To communicate with the CS43L22, the chip address field, which is the first byte sent to the CS43L22, should match 100101 followed by the setting of the AD0 pin. The eighth bit of the address is the R/W bit. If the operation is a write, the next byte is the Memory Address Pointer (MAP), which selects the register to be read or written. If the operation is a read, the contents of the register pointed to by the MAP will be output. Setting the auto-increment bit in MAP allows successive reads or writes of consecutive registers. Each byte is separated by an acknowledge bit. The ACK bit is output from the CS43L22 after each input byte is read and is input to the CS43L22 from the microcontroller after each transmitted byte.



rozman 14. 03. 2023, 1..

AD0 -> GND Addr=0x94

Figure 16. Control Port Timing, I²C Write

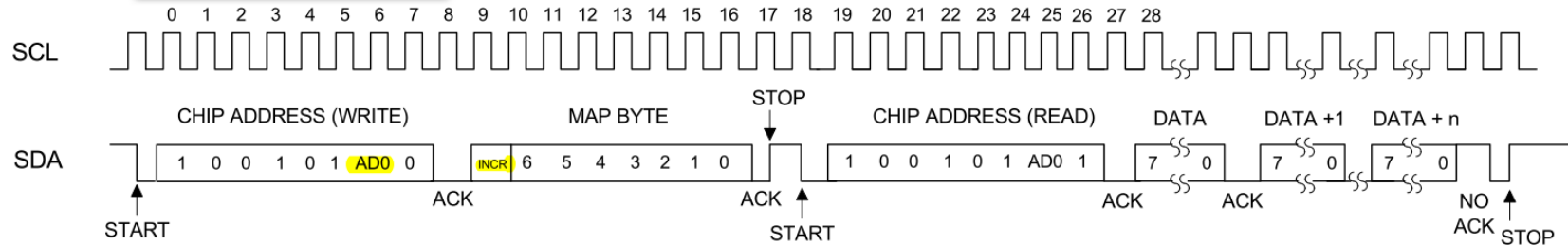
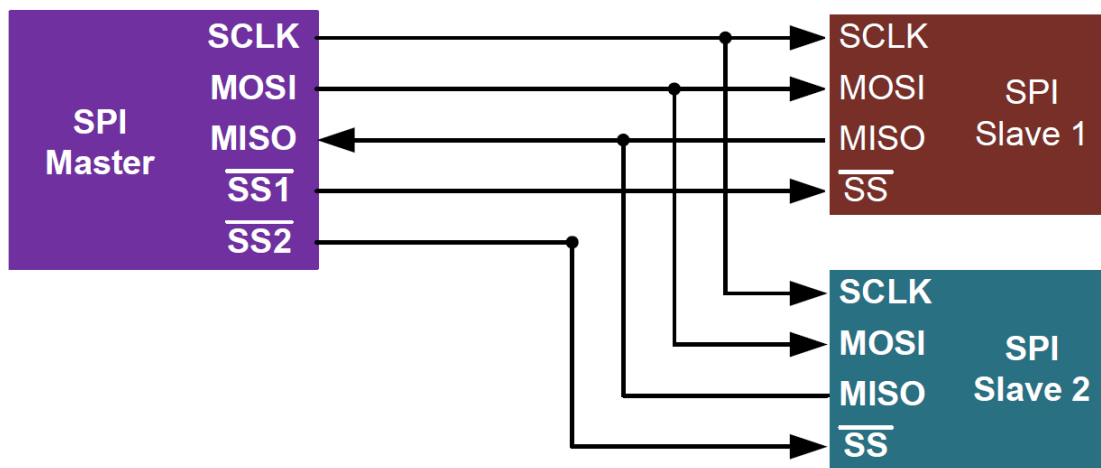


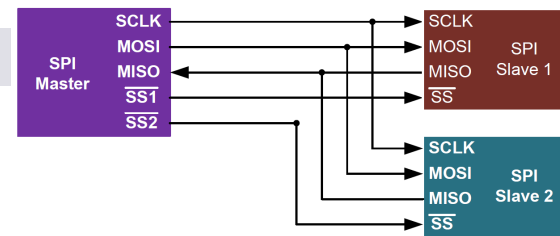
Figure 17. Control Port Timing, I²C Read

4.2.2 Vodilo SPI (Serial Peripheral Interface)

- ❑ 1985: pojavi se potreba **po višjih hitrostih**, kot jih ponuja I2C
- ❑ SPI vmesnik ima **3 ali 4 linije, 2 za prenos podatkov**:
 - **MISO (Master In Slave Out)** – naprava Master sprejme podatke od naprave slave
 - **MOSI (Master Out Slave In)** - naprava master pošlje podatke napravi slave
 - **SCK (Serial Clock)** – urin signal, ki ga pošlje naprava master
 - **\overline{SS} - Slave (Chip) select** (omogoča izbiro naprave slave, kadar jih je več povezanih na master)



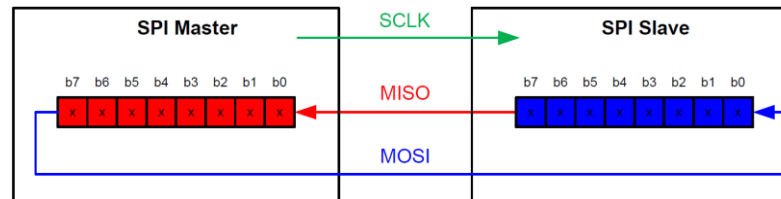
Video: Microchip: <https://www.youtube.com/watch?v=NyxQkGXbG6I>



❑ Prenos podatkov je omogočen z načinom povezave **točka-v-točko** (ang. **point-to-point**)

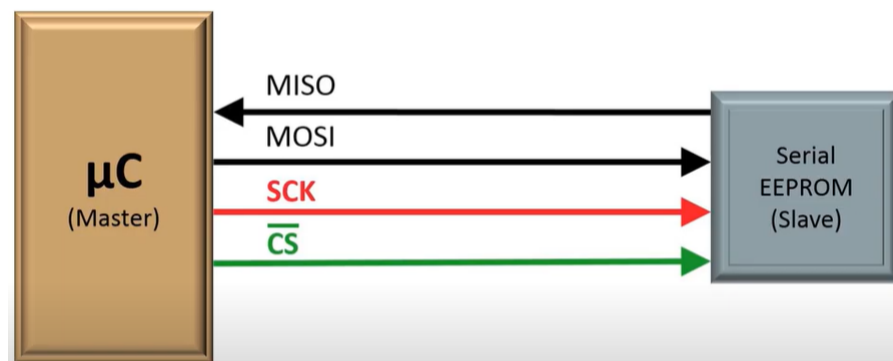
- Naprava **master pošilja napravi slave** bit za bitom po liniji MOSI, običajno najprej MSB bit.
- Naprava **slave pošilja napravi master** bit za bitom po liniji MISO, običajno najprej LSB bit.

❑ Prenos podatkov poteka istočasno **v obe smeri** (ang. **full-duplex**)



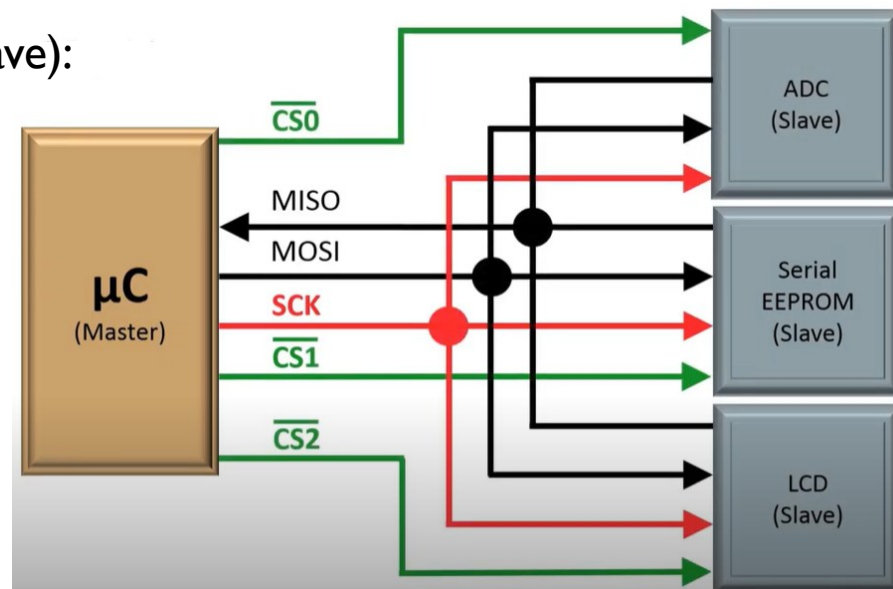
□ Vodilo SPI je vmesnik med dvema napravama, ki ima 4 povezave

- Mikrokrmilnik je glavna naprava (Master)
- EEPROM je delovna naprava (Slave)



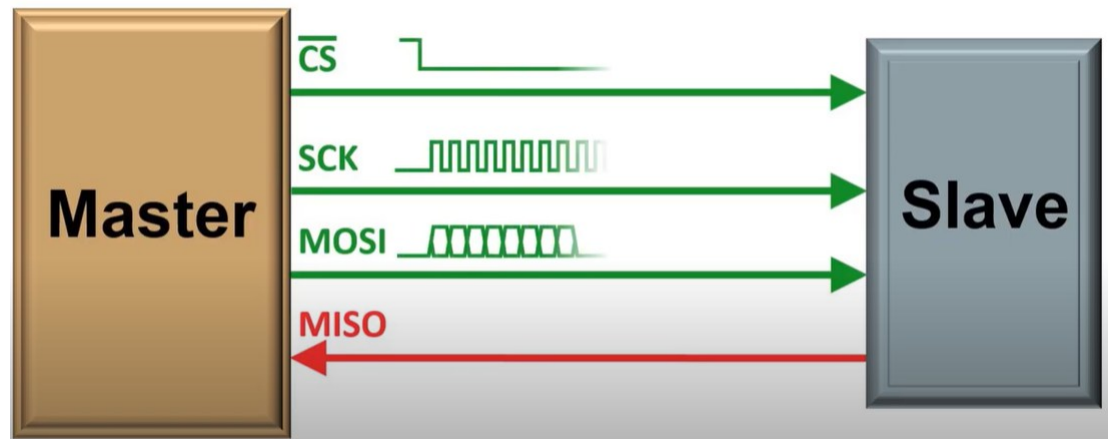
□ 4 naprave na vodilu SPI (Master + 3x Slave):

- Mikrokrmilnik (Master)
- ADC (Slave)
- EEPROM (Slave)
- LCD (Slave)
- Vsaka naprava Slave ima svoj signal $\overline{\text{CS}}$

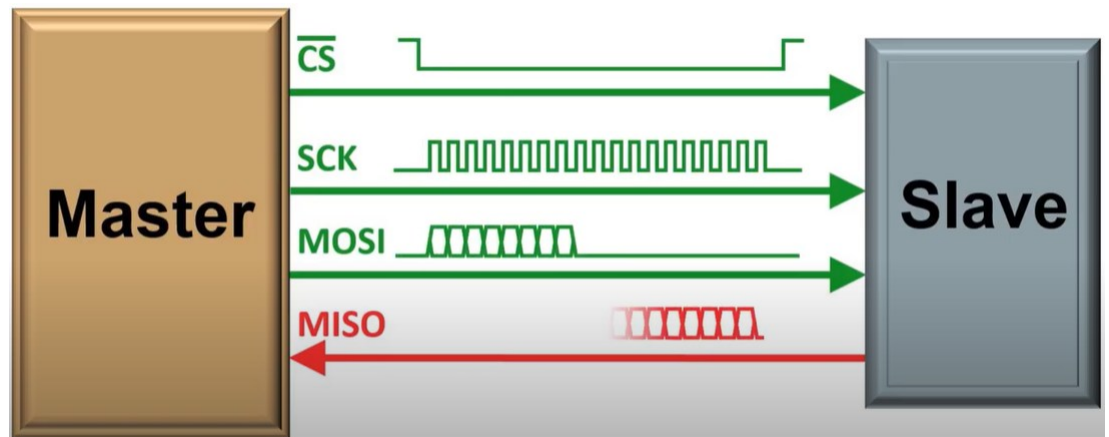


□ Signali SPI pri prenosu podatkov

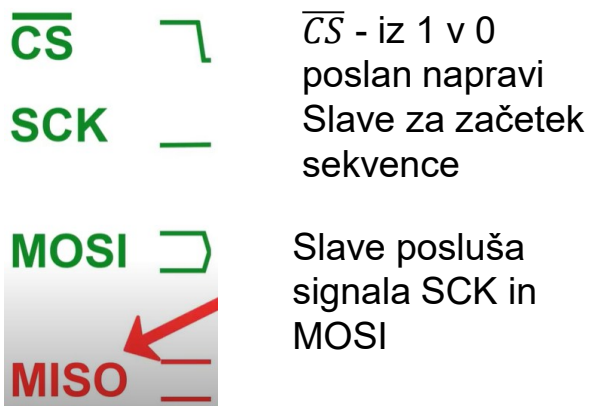
- Master generira signale \overline{CS} , SCK, MOSI



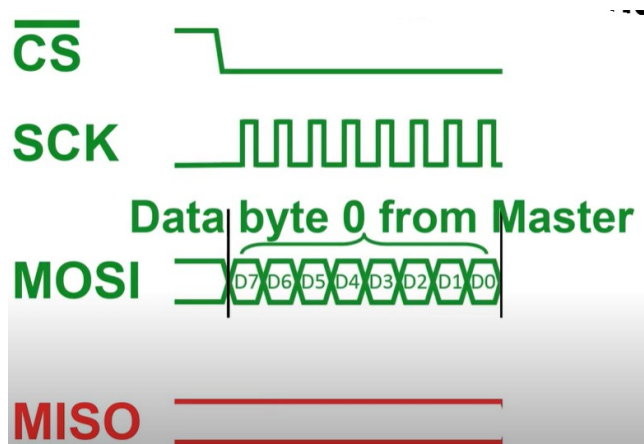
- Slave generira signal MISO



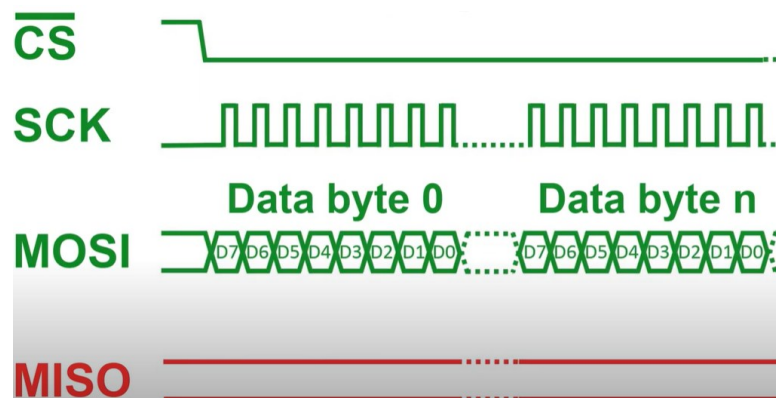
□ Serijski prenos podatkov iz naprave Master v Slave



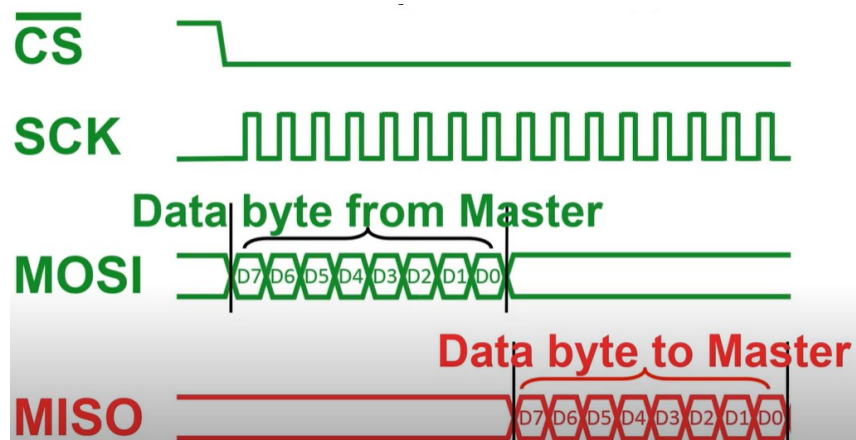
- Master prenese **bajt 0** po liniji MOSI



oz n bajtov se prenese

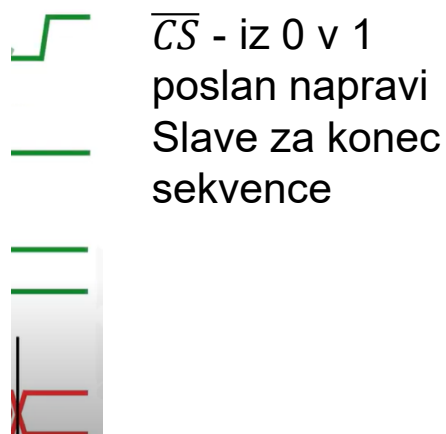


- Prenos podatkov Master to Slave in Slave to Master



- Konec prenosa:

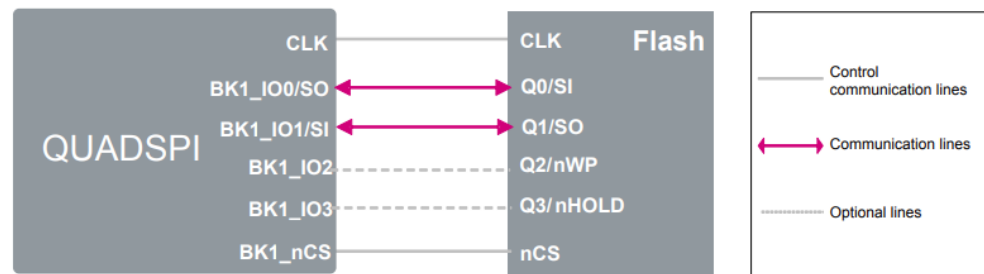
Chip Select gre iz Low v High



Hitrost prenosa podatkov je odvisna of frekvence urinega signala

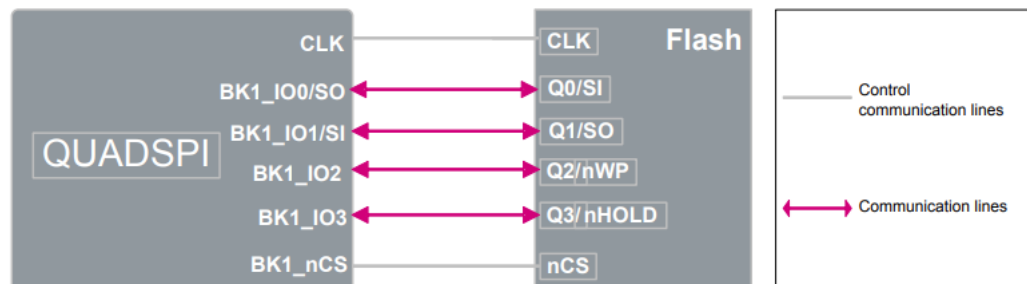
- SPI najpogosteje omogoča delovanje pri **frekvenci 20 MHz**.
- **Dual SPI in Quad SPI omogoča delovanje do 144 MHz**.
 - **Dual SPI** - izmenični prenos (ang. half duplex) dveh bitov hkrati
 - MOSI (SI) definiran kot IO0, in MISO (SO) pa IO1.

Figure 11. Hardware configuration: Dual-SPI mode



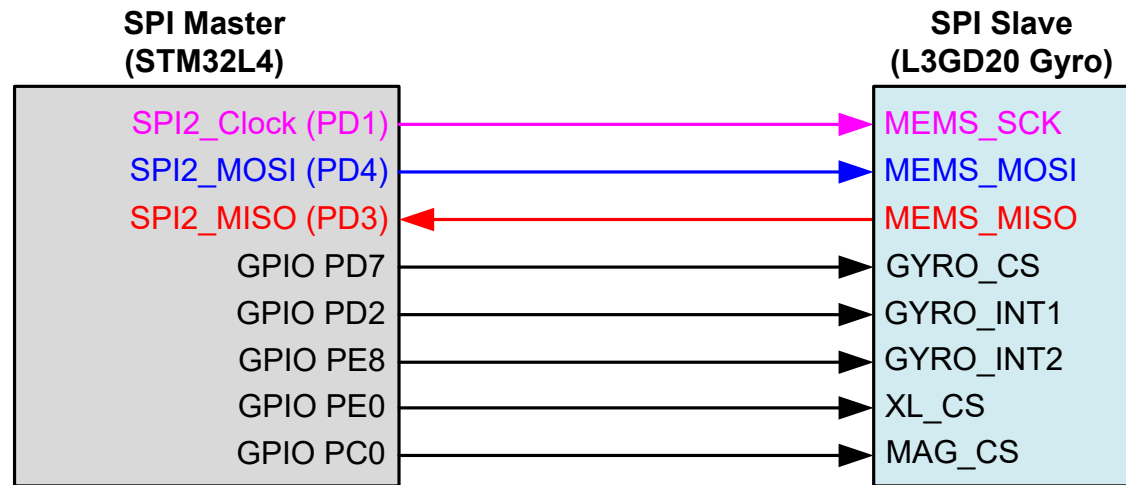
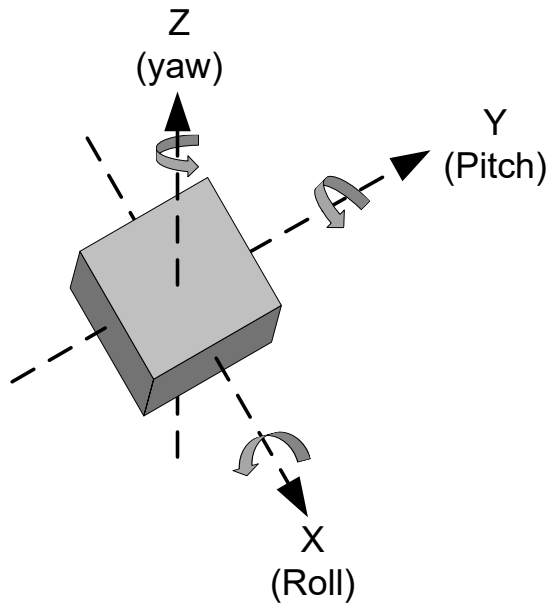
- **QUAD SPI** –izmenični prenos (ang. half duplex) štirih bitov hkrati
 - preko linij IO0, IO1, IO2, IO3.

Figure 12. Hardware configuration: Quad-SPI mode



Primeri SPI povezav

□ Primer žiroskopa (L3GD20)



Primer SPI: STM32F4 – pospeškomer LIS3DSH

```

/* USER CODE BEGIN PV */
// Global variables
uint8_t indata[2];
uint8_t outdata[2] = {0,0};
uint8_t lis_id;
HAL_StatusTypeDef SPIStatus;
/* USER CODE END PV */

```

```

MX_SPI1_Init();

```

```

/* USER CODE BEGIN 2 */

```

```

// Config accelerometer
// Read WHOAMI register

```

```

HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);

```

```

outdata[0] = 0x0f | 0x80 ; // read whoami (0x0f) + b7=read

```

```

HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);

```

```

lis_id = indata[1];

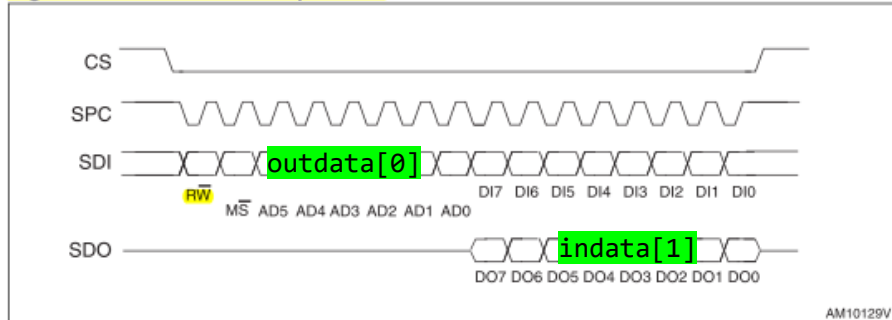
```

```

HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

```

Figure 6. Read and write protocol



LIS3DSH

MEMS digital output motion sensor
 ultra low-power high performance three-axis “nano” accelerometer

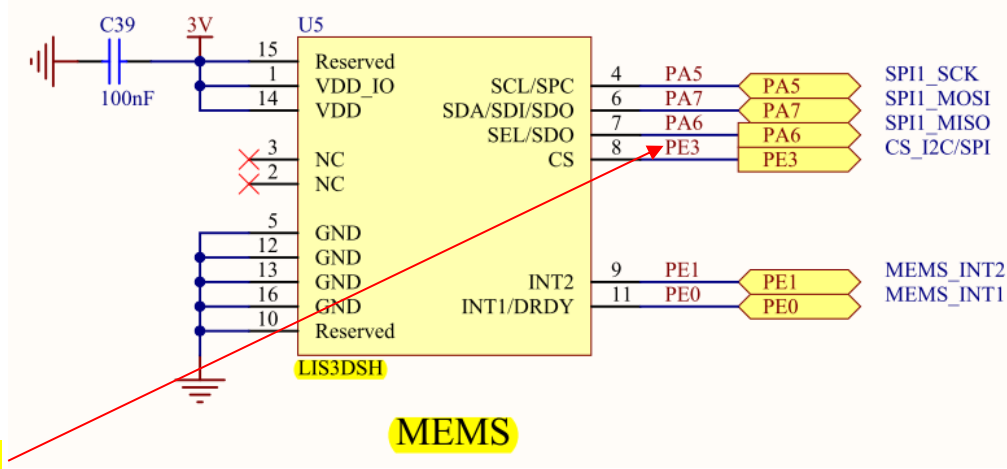


Table 16. Register address map

| Name | Type | Register address | | Default |
|-----------------|------|------------------|-----------------|------------------|
| | | Hex | Binary | |
| INFO1 | r | 0D | 00001101 | 0010 0001 |
| INFO2 | r | 0E | 00001110 | 0000 0000 |
| WHO_AM_I | r | 0F | 00001111 | 0011 1111 |

8.3 WHO_AM_I (0Fh)

Who_AM_I register.

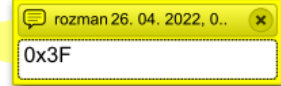


Table 19. WHO_AM_I register default value

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

SPI HAL Read WhoAmI register value

```
outdata[0] = 0x0f | 0x80 ; // read whoami (0x0f) + b7=read  
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
```



UM1725

SPI Firmware driver API description

HAL_SPI_TransmitReceive

Function name

HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)

Function description

Transmit and Receive an amount of data in blocking mode.

Parameters

- **hspi**: pointer to a SPI_HandleTypeDef structure that contains the configuration information for SPI module.
- **pTxData**: pointer to transmission data buffer
- **pRxData**: pointer to reception data buffer
- **Size**: amount of data to be sent and received
- **Timeout**: Timeout duration

Return values

- **HAL**: status

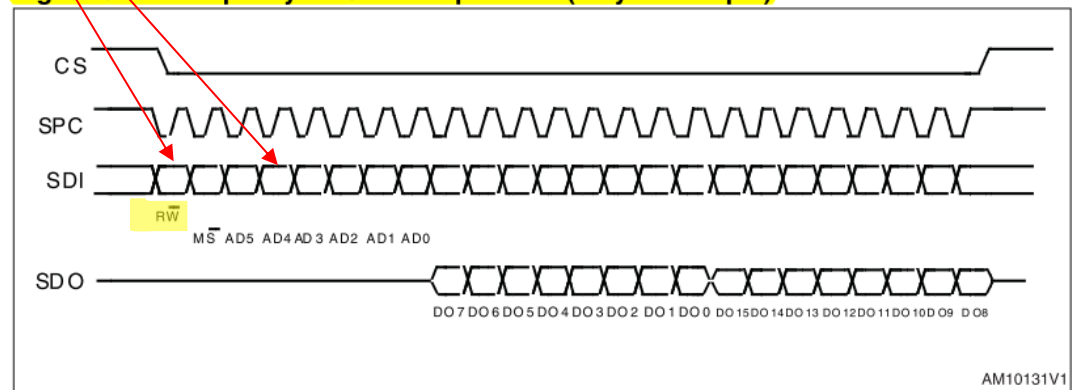
bit 0: READ bit. The value is 1.

bit 1-7: address AD(6:0). This is the address field of the indexed register.

bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

bit 16-...: data DO(...-8). Further data in multiple byte reading.

Figure 8. Multiple bytes SPI read protocol (2-byte example)



7 Register mapping

Table 16 provides a list of the 8/16-bit registers embedded in the device and the related address:

Table 16. Register address map

| Name | Type | Register address | | Default | Comment |
|-----------|------|------------------|----------|-----------|------------------------|
| | | Hex | Binary | | |
| INFO1 | r | 0D | 00001101 | 0010 0001 | Information register 1 |
| INFO2 | r | 0E | 00001110 | 0000 0000 | Information register 2 |
| WHO_AM_I | r | 0F | 00001111 | 0011 1111 | Who I am ID |
| CTRL_REG3 | r/w | 23 | 00100011 | - | Control registers |
| CTRL_REG4 | r/w | 20 | 00100000 | - | |
| CTRL_REG5 | r/w | 24 | 00100100 | - | |
| CTRL_REG6 | r/w | 25 | 00100101 | - | |
| STATUS | r | 27 | 00100111 | - | Status data register |

| | | | | | |
|---------|---|----|----------|-----------|------------------|
| OUT_X_L | r | 28 | 00101000 | 0000 0000 | Output registers |
| OUT_X_H | r | 29 | 00101001 | | |
| OUT_Y_L | r | 2A | 00101010 | | |
| OUT_Y_H | r | 2B | 00101011 | | |
| OUT_Z_L | r | 2C | 00101100 | | |
| OUT_Z_H | r | 2D | 00101101 | | |

```
// read whoami (0x0f) + b7=read
outdata[0] = 0x0f | 0x80 ;
```

3.3 Communication interface characteristics

3.3.1 SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

Table 5. SPI slave timing values

| Symbol | Parameter | Value (1) | | Unit |
|---------|---------------------|-----------|------|------|
| | | Min. | Max. | |
| tc(SPC) | SPI clock cycle | 100 | | ns |
| fc(SPC) | SPI clock frequency | | 10 | MHz |

6 Digital interfaces

The registers embedded inside the LIS3DSH may be accessed through both the I²C and SPI serial interfaces. The latter may be SW configured to operate either in 3-wire or 4-wire interface mode.

The serial interfaces are mapped onto the same pins. To select/exploit the I²C interface, the CS line must be tied high (i.e. connected to Vdd_IO).

Table 9. Serial interface pin description

| Pin name | Pin description |
|----------|---|
| CS | SPI enable I2C/SPI mode selection (1: SPI idle mode / I2C communication enabled; 0: SPI communication mode / I2C disabled) |

VP 4 - STM32 CubeIDE, SPI in LIS3DSH - Osciloskop

```
// Read x,y,z axes
```

```
outdata[0] = 0x29 | 0x80 ; // read x
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
AccelX = indata[1];
```

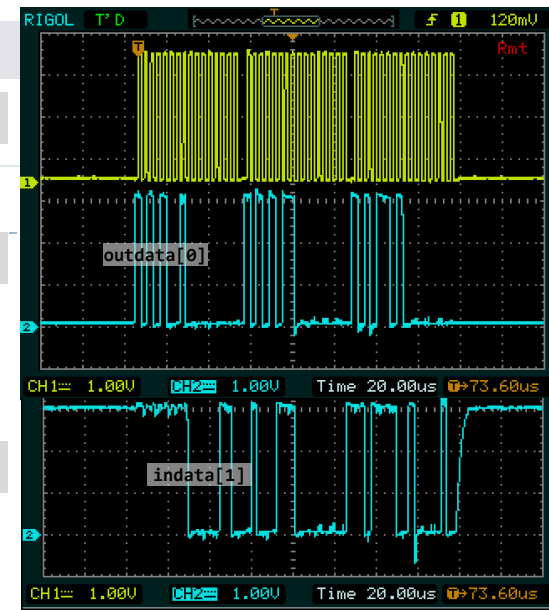
```
outdata[0] = 0x2B | 0x80 ; // read y
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
AccelY = indata[1];
```

```
outdata[0] = 0x2D | 0x80 ; // read z
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
AccelZ = indata[1];
```

SCK

MOSI

MISO

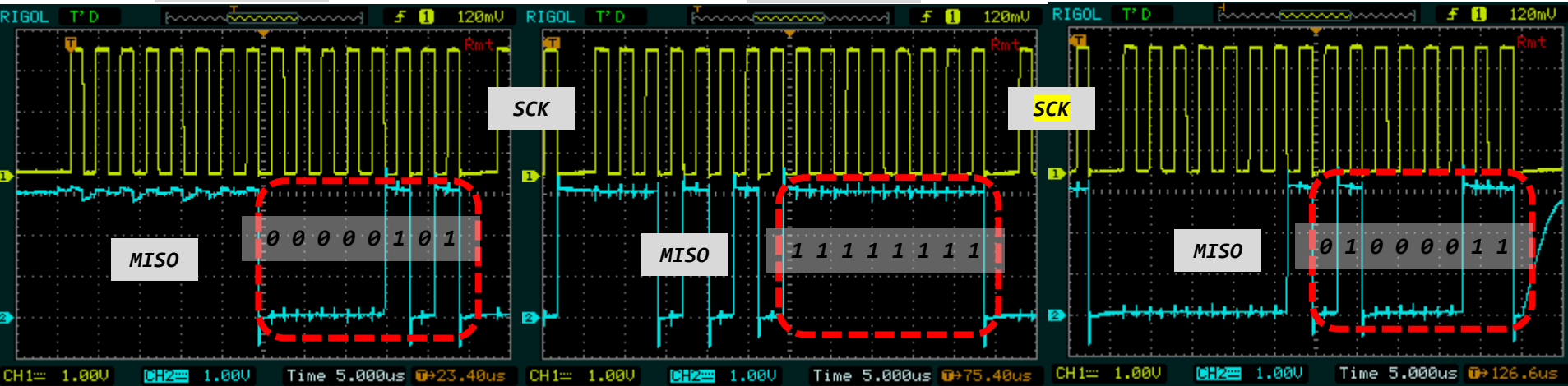


```
Hello World [3530]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0066
Hello World [3531]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0067
```

X-Accel: 5

Y-Accel: -1

Y-Accel: 67



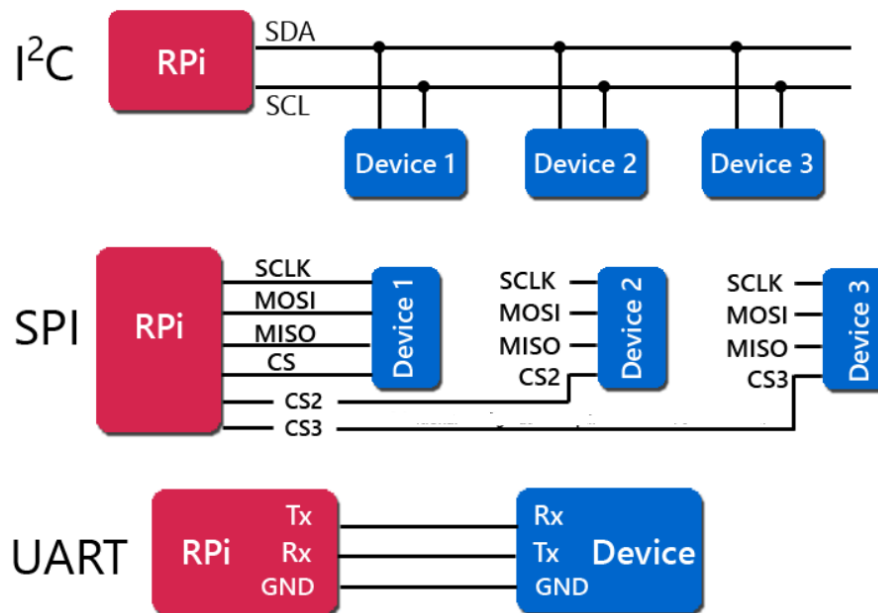
Primerjava SPI vs I²C

- ❑ Oba sinhronska protokola za lokalne komunikacije
- ❑ SPI (Motorola), I2C (Philips – danes NXP)

| | SPI | I2C |
|-----------|--|--|
| Prednosti | <ul style="list-style-type: none">• Hitrejši (20Mbps+)• Full duplex• „Low power“• DualSPI, QuadSPI• Enostavna implementacija• Izbira dolžine enot (npr. 12 bitov) | <ul style="list-style-type: none">• Enostavnost (manj povezav, standard za 2 povezavi)• Dodajanje novih naprav• Multi-Master• Ima mehanizem ACK |

Primer: Raspberry Pi ↔ I²C, SPI, UART

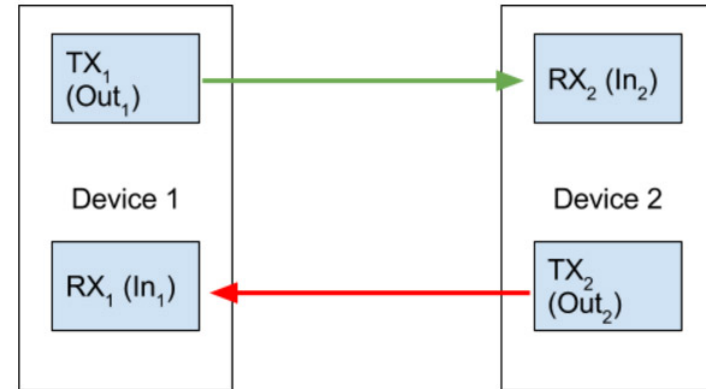
- Diagram povezav
Ri = Raspberry Pi



- **UART** - preprost; majhna hitrost; ura ni potrebna; omejena na eno napravo, priključeno na Ri.
- **I²C** - hitrejši od UART, vendar ne tako hitro kot SPI; lažje povezovanje številnih naprav; Ri poganja uro, tako da ni težav s sinhronizacijo.
- **SPI** - najhitrejši od treh; Ri poganja uro, tako da ni težav s sinhronizacijo; praktična omejitev števila naprav na Ri.
- <https://www.mbtechworks.com/hardware/raspberry-pi-UART-SPI-I2C.html>

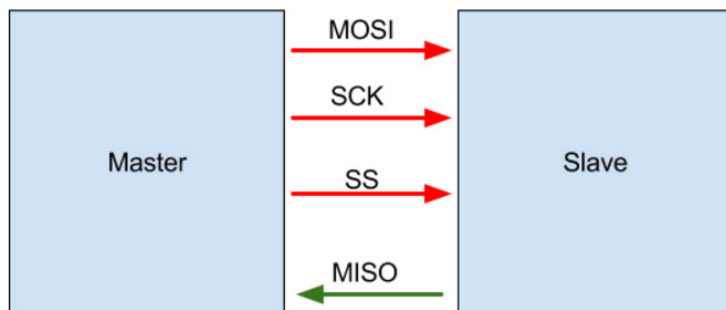
Primer: Arduino ↔ UART, SPI, I2C

- Asinhronski serijski prenos: **UART**

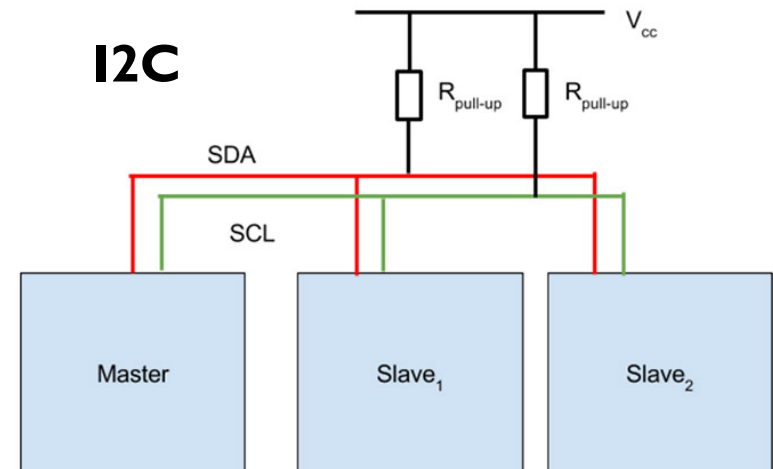


- Sinhronski serijski prenos:

SPI



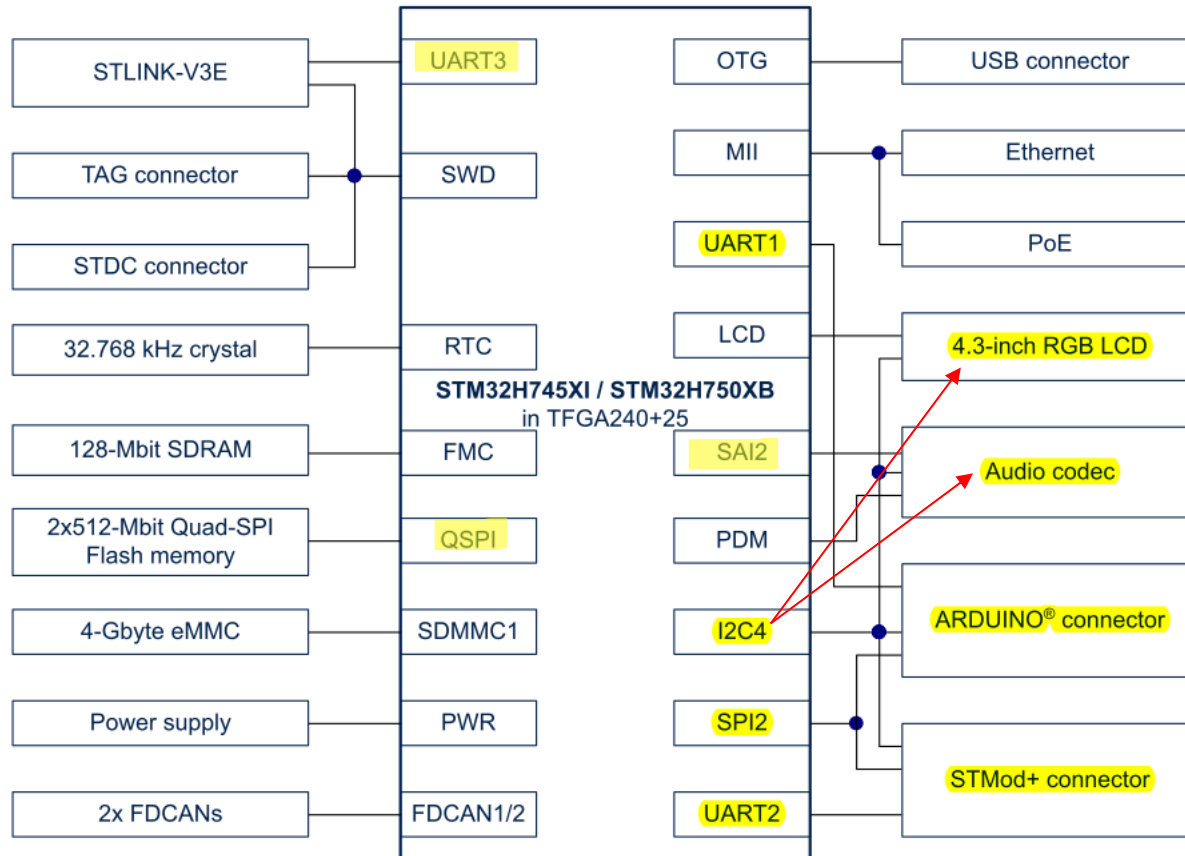
I2C



- <https://www.deviceplus.com/arduino/arduino-communication-protocols-tutorial/>

Primeri naprav: STM32H750B ↔ UART, SPI, I2C

Figure 3. Hardware block diagram



Primer LAB vaje