Vhodno izhodne naprave

Laboratorijska vaja 10

STM32H7 – Generator signalov

Laboratorijska vaja 10
STM32H7 – Generator signalov - Izziv

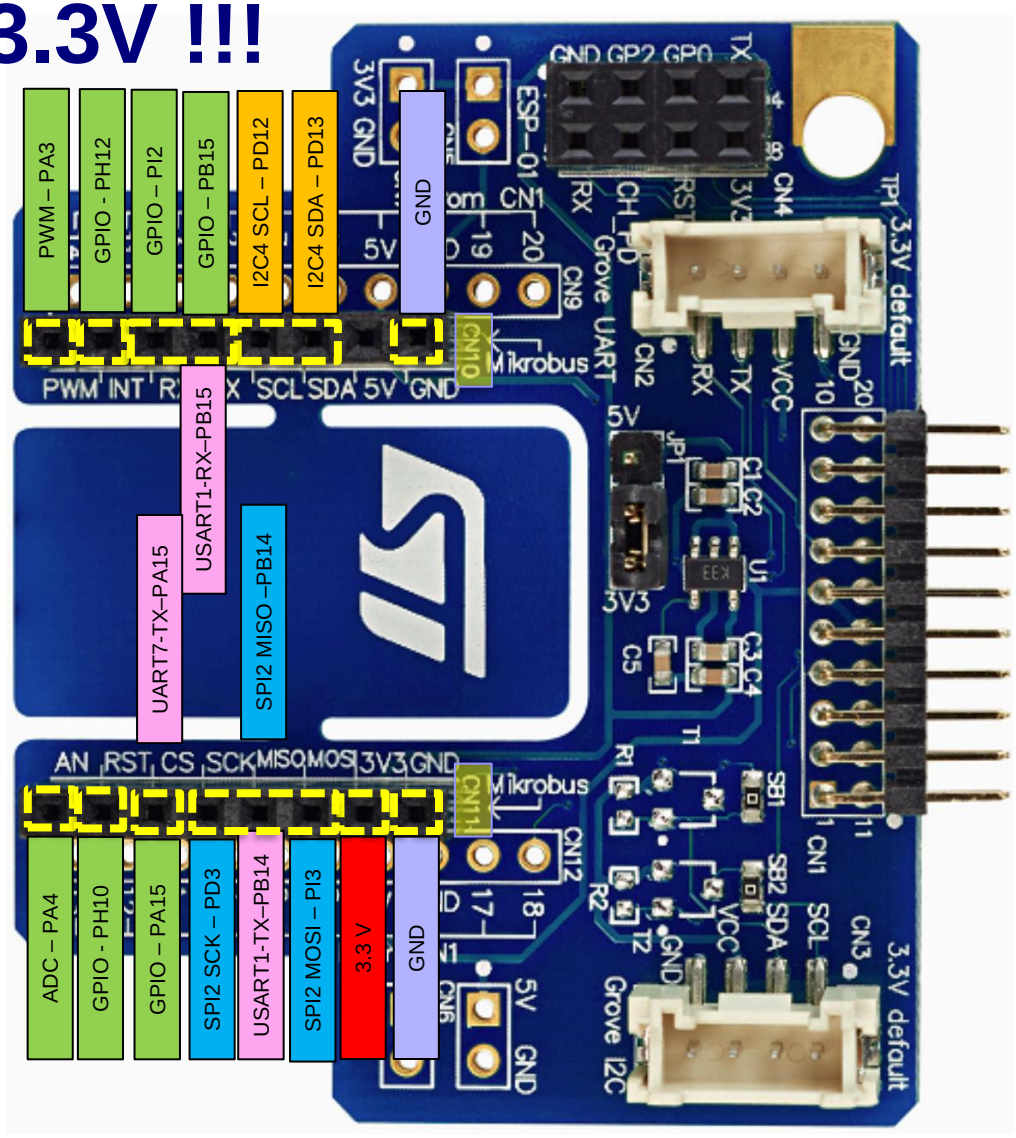
Osvežitev STM32H7

Izzivi:

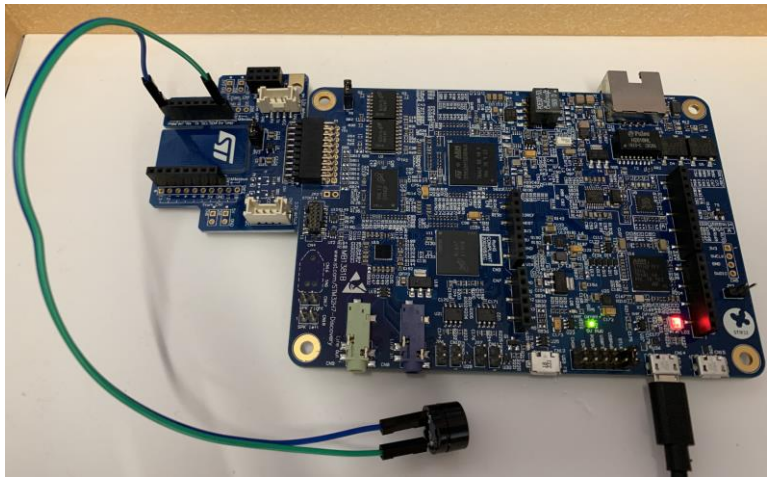
- PWM PA3
- UART PB14
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

STM32H750B – DISCOVERY StMod+ konektor

3.3V !!!



Pravilna priključitev

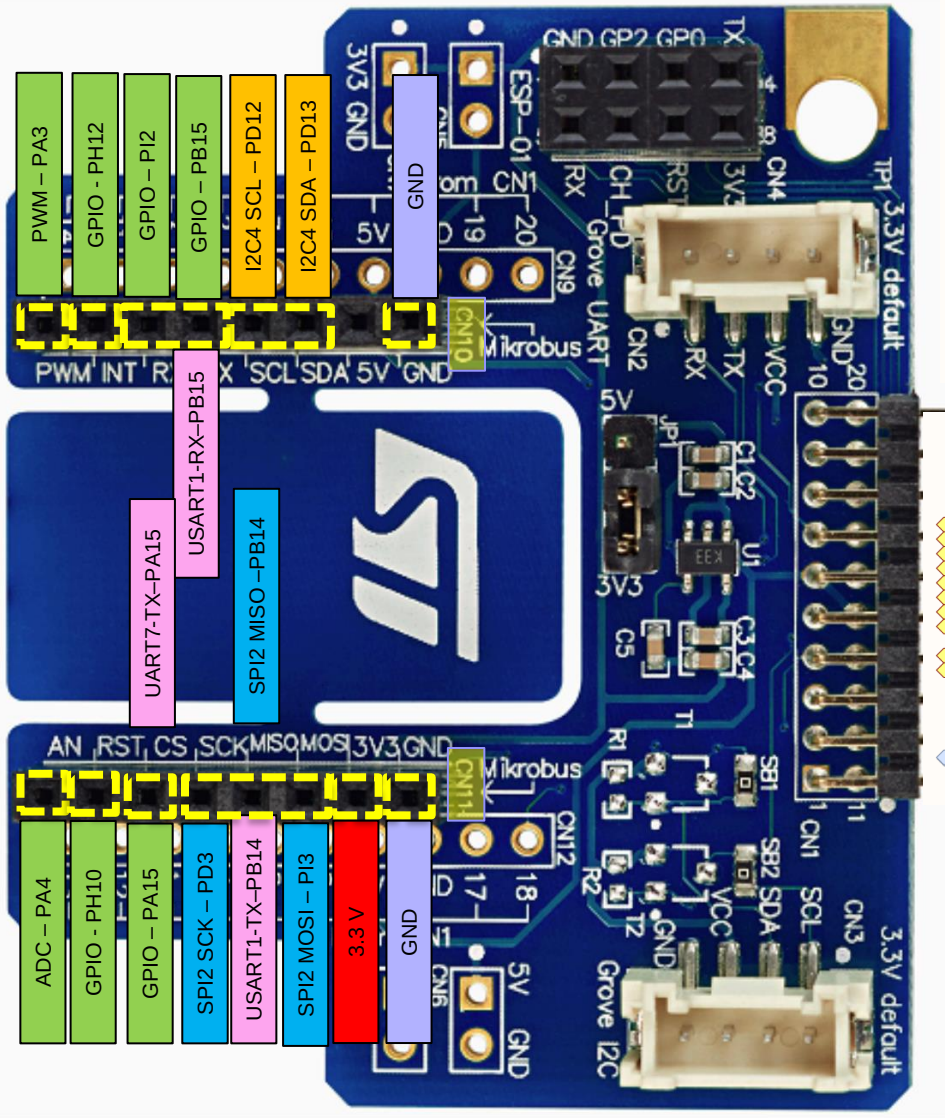


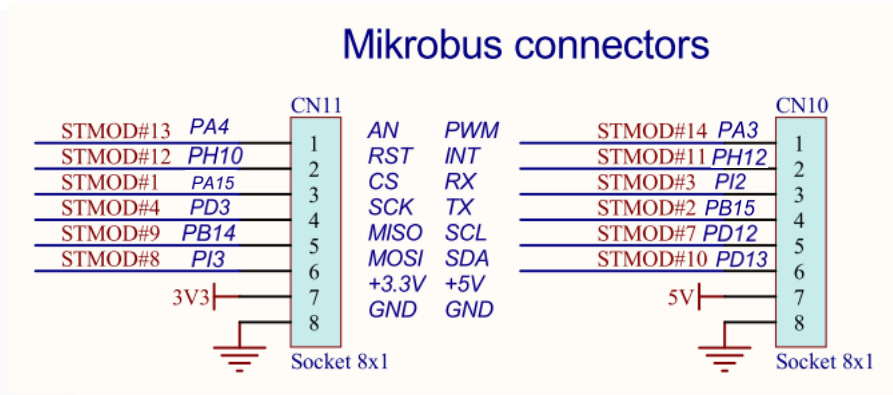
Nepravilna priključitev

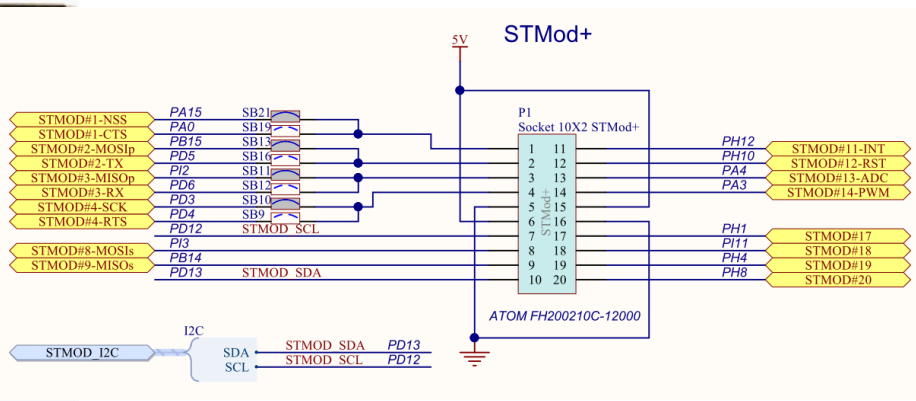
https://www.st.com/en/evaluation-tools/stm32h750b-dk.html

3.3V !!!

STM32H750B – DISCOVERY StMod+ konektor



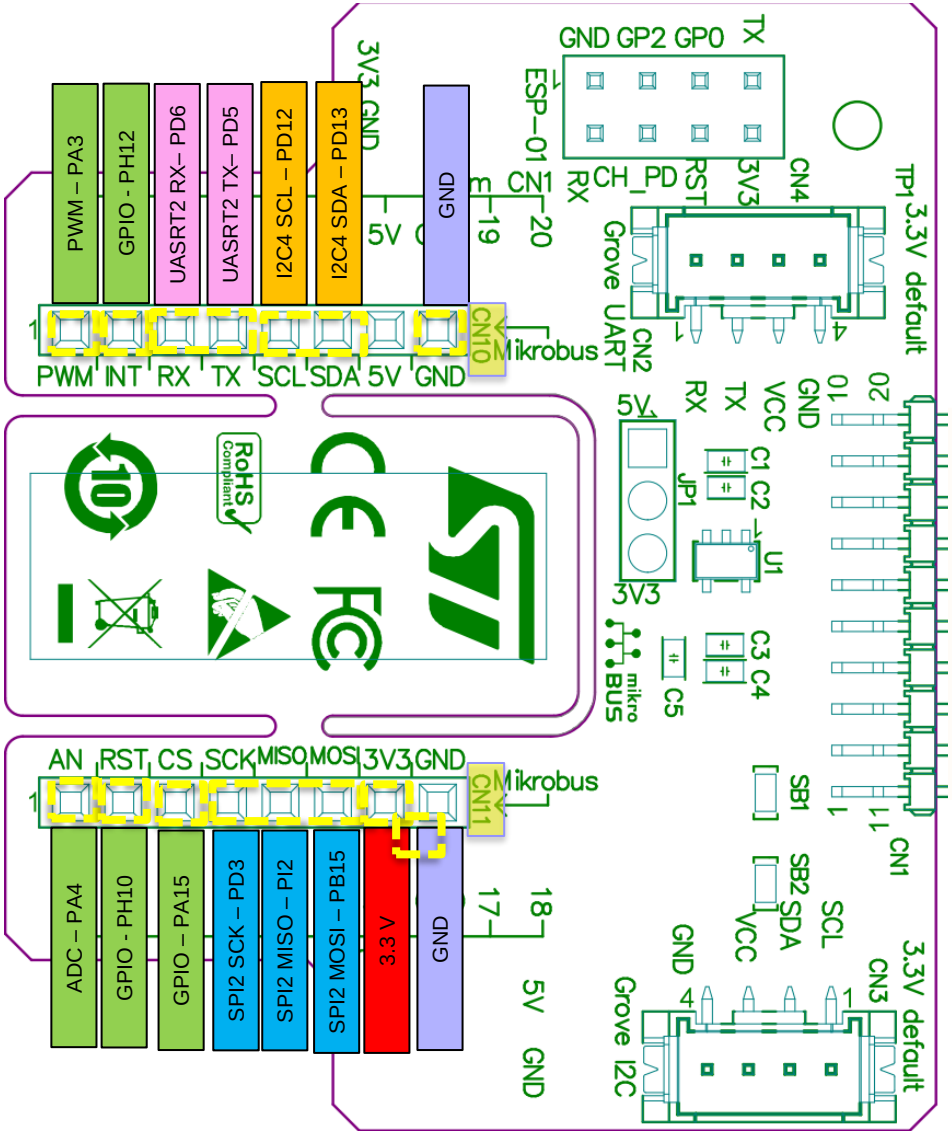


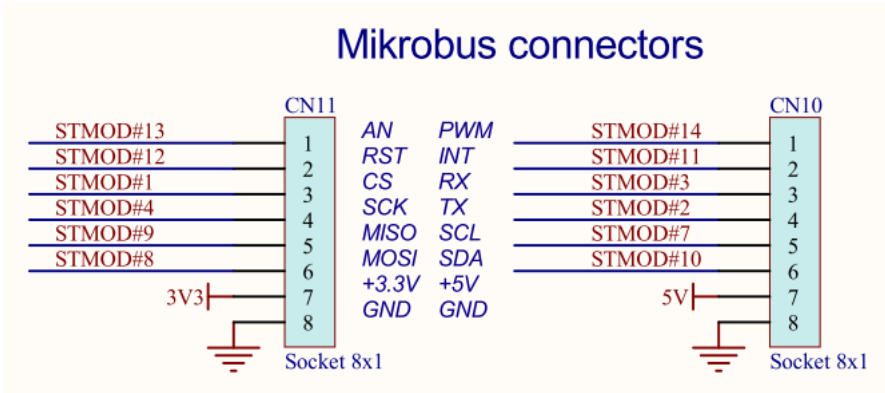


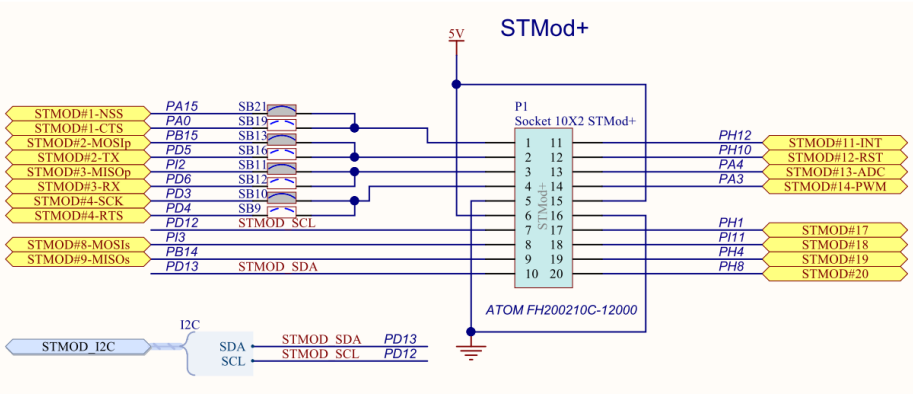
STM32H7

3.3V !!!

STM32H750B – DISCOVERY StMod+ konektor







STM32H7

3.3V !!!

STM32H750B – DISCOVERY CANBUS konektorja

STM32H7

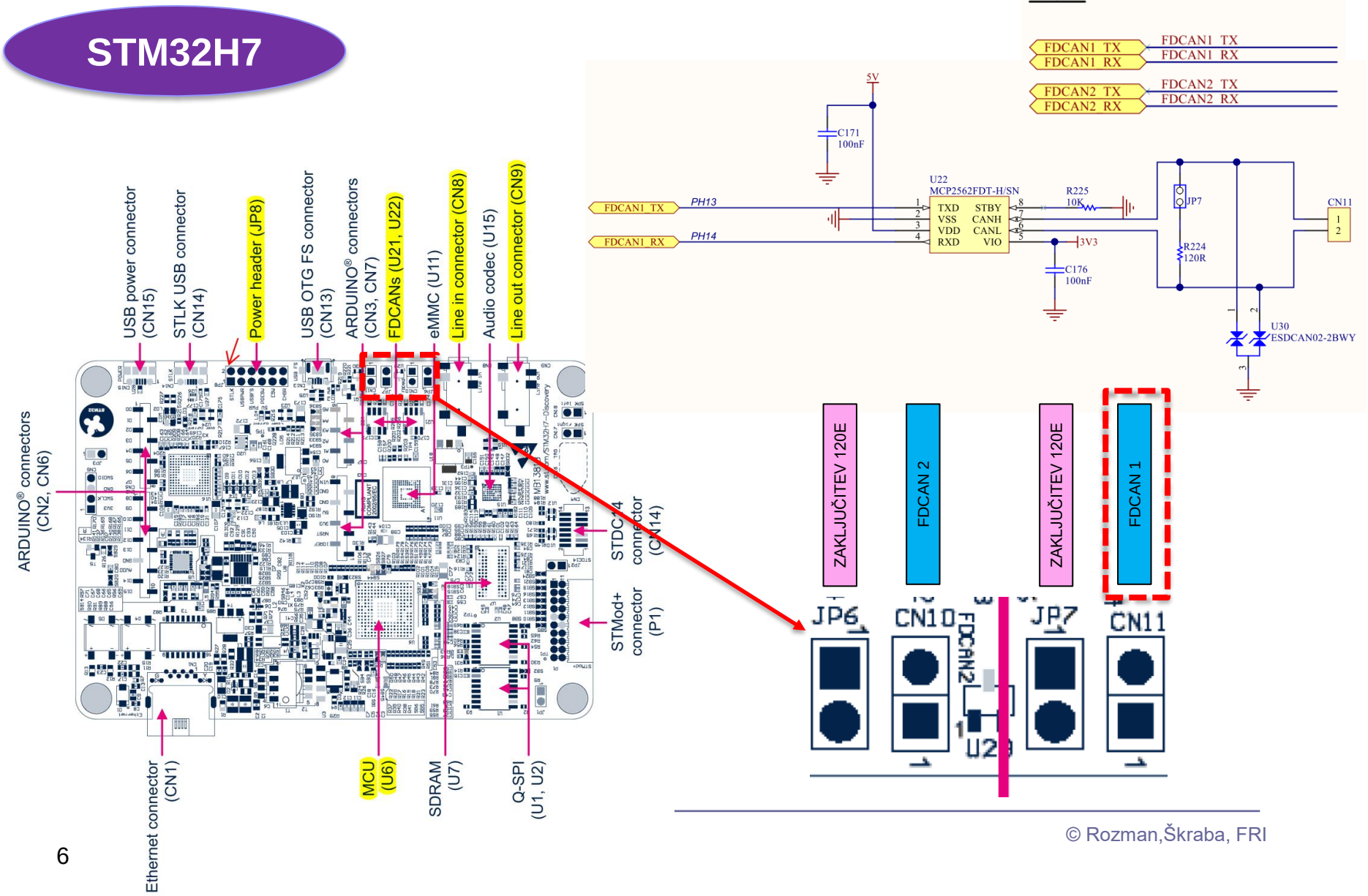


Figure 5. STM32H745I-DISCO and STM32H750B-DK Discovery board bottom layout

Laboratorijska vaja 10
STM32H7 – Generator signalov

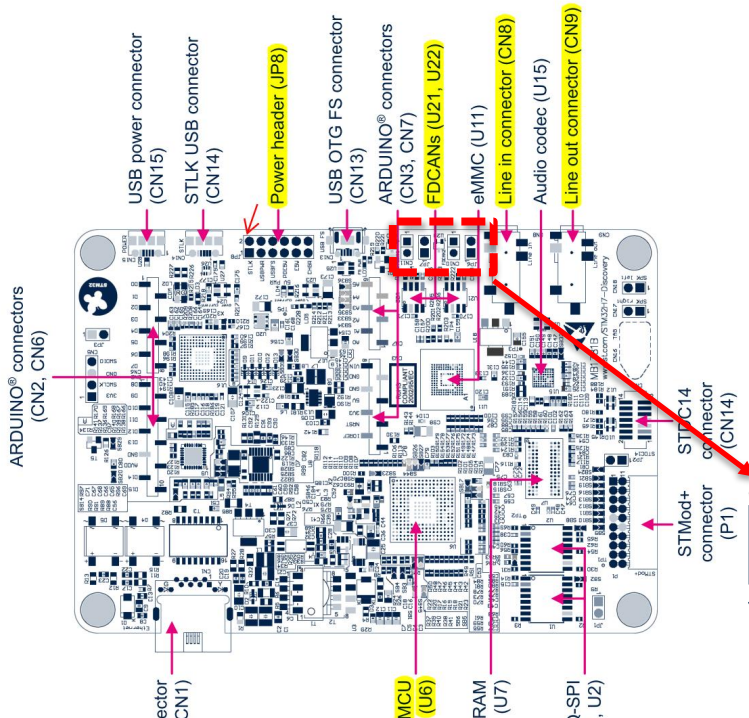
Osvežitev STM32H7

Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

Izzivi - povezave

- PWM PA3
- UART PB14
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1)
  - CAN-L, CAN-H



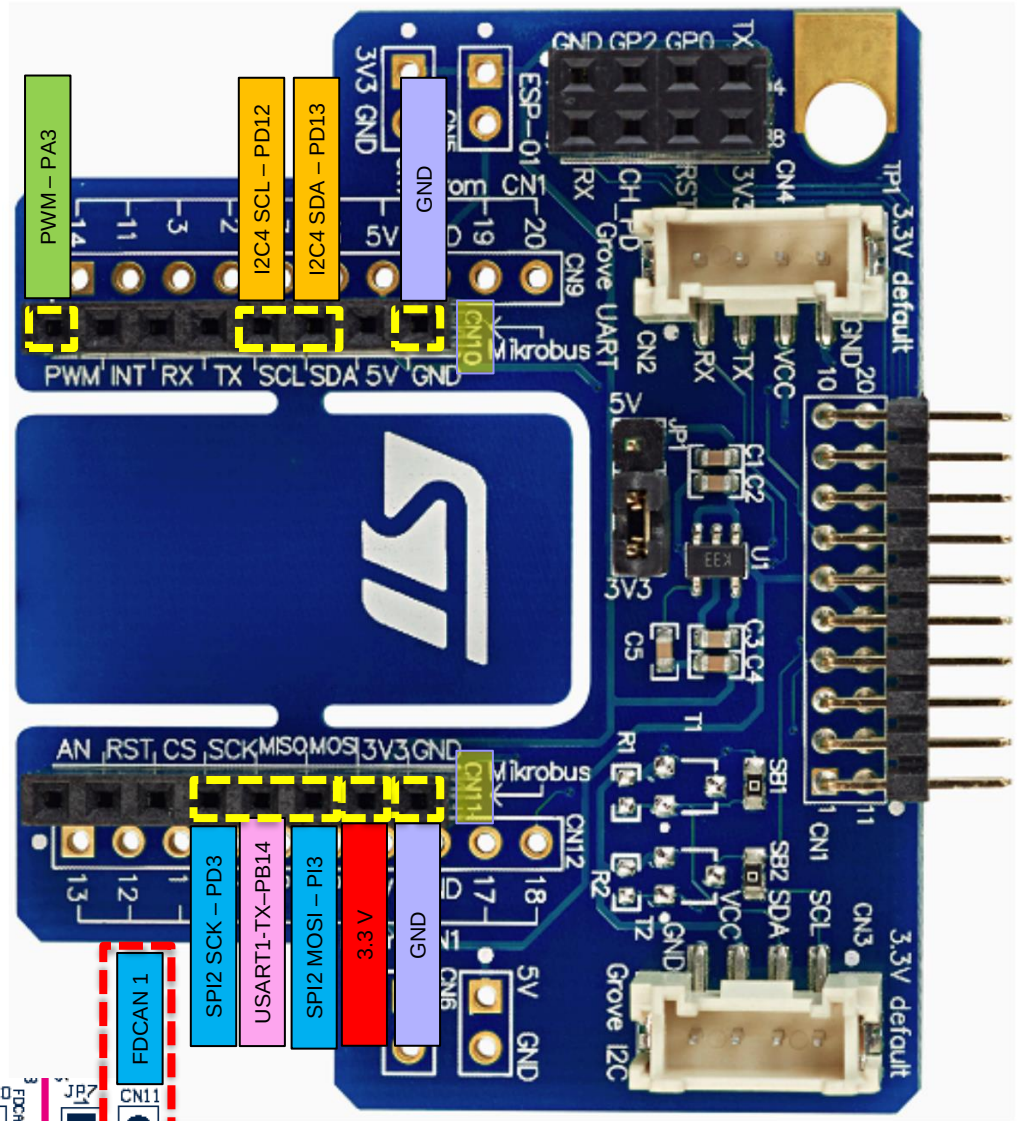


Figure 5. STM32H745I-DISCO and STM32H750B-DK Discovery board bottom layout

Laboratorijska vaja 10
STM32H7 – Generator signalov

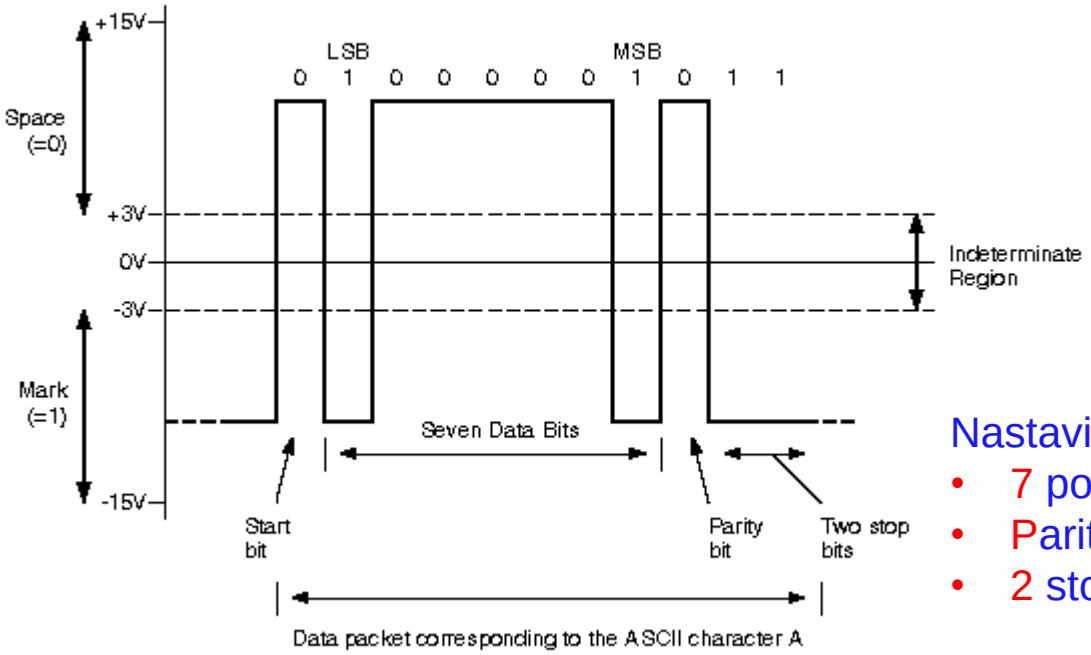
Osvežitev STM32H7

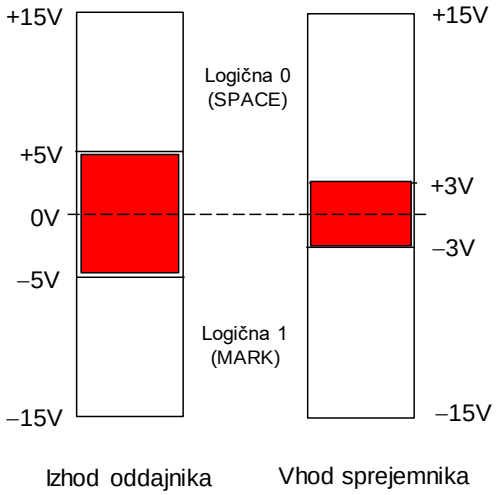
Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

Določite bitno hitrost prenosa in ugotovite ASCII kodo znakov, ki se prenašajo ob nastavitvi 8N1 (8 podatkovnih bitov, brez paritetnega bita, 2 stop bita).

- Primer poteka signala RS232 – nastavitve „7P2“:
  - Napetostna in logična nivoja

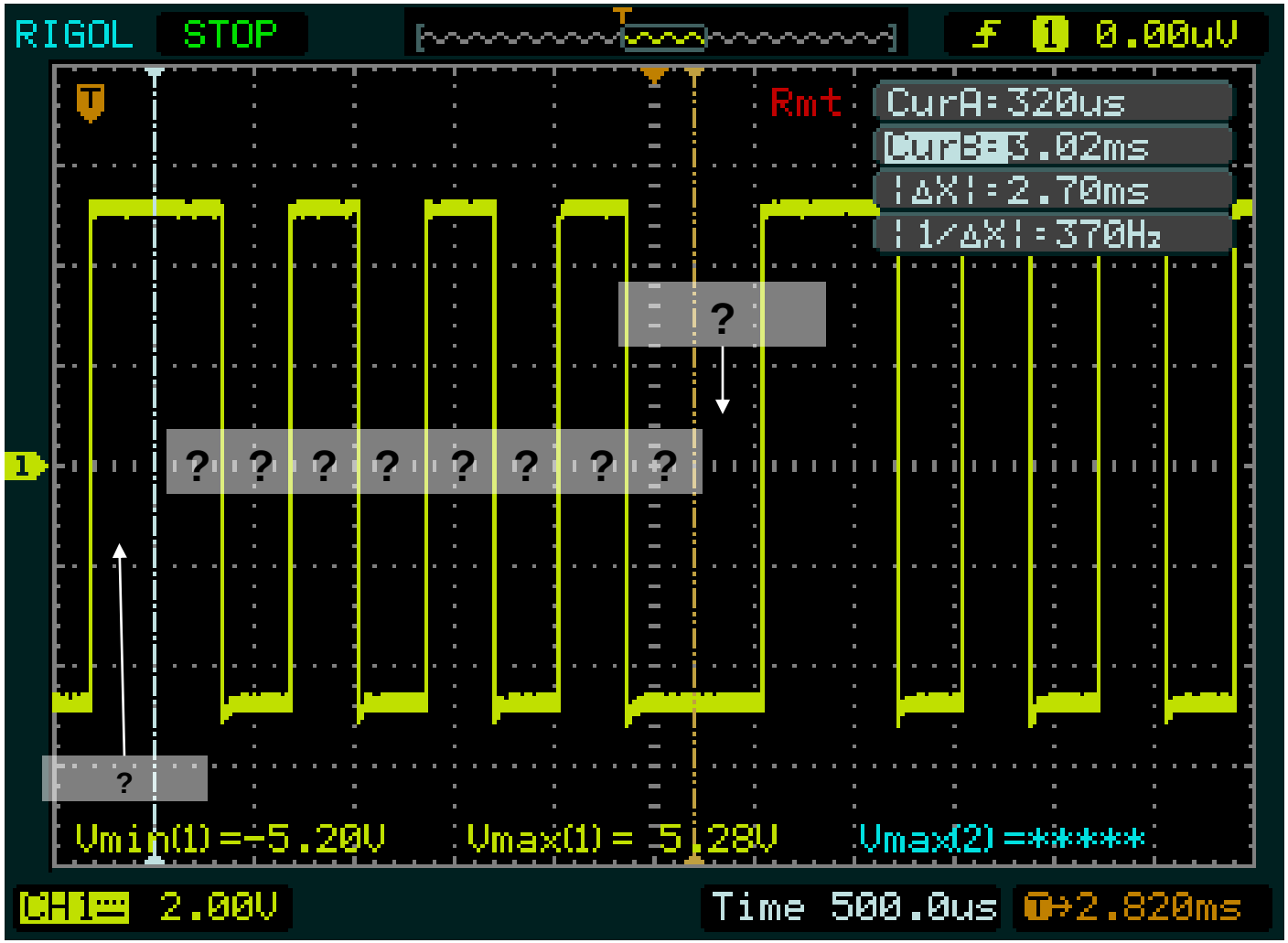




Nastavitve RS232 na prikazani sliki – „7P2“:

- 7 podatkovnih bitov
- Paritetni bit
- 2 stop bita

Primer reševanja izziva:



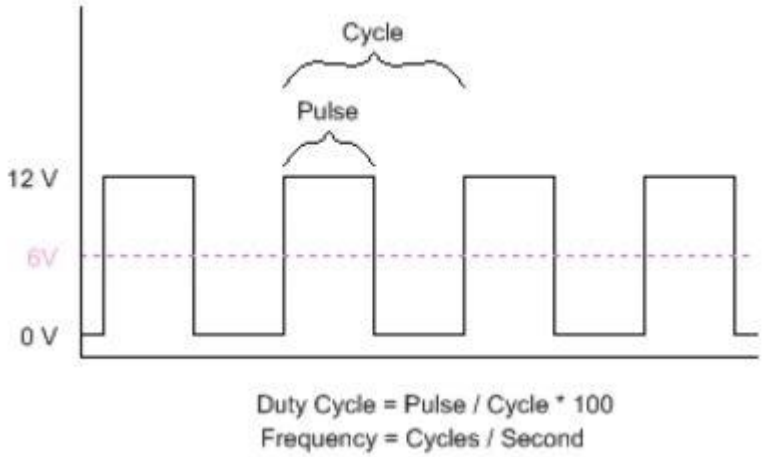
Laboratorijska vaja 10
STM32H7 – Generator signalov

Osvežitev STM32H7

Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

Določite frekvenco PWM signala in ustrezno noto.



Example of a PWM signal at 50% duty cycle

```
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
```

Laboratorijska vaja 10
STM32H7 – Generator signalov

Osvežitev STM32H7

Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

5 Digital main blocks

5.1 State machine

The LIS3DSH embeds two state machines able to run a user defined program.

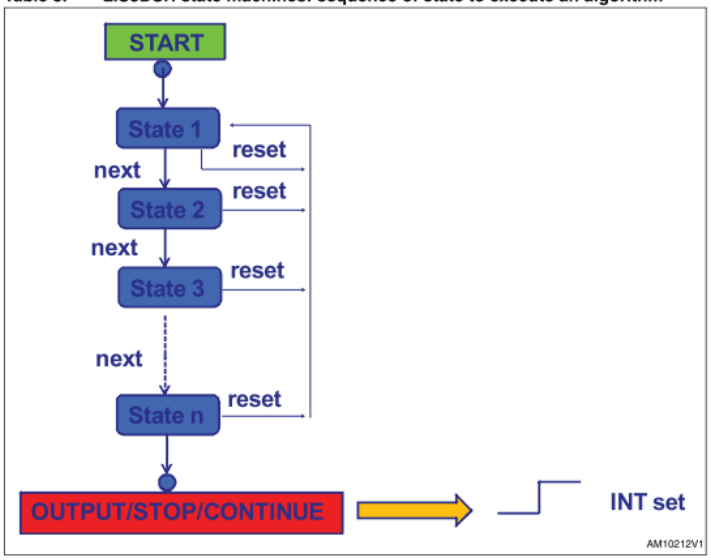
The program is made up of a set of instructions that define the transition to successive states. Conditional branches are possible.

From each state (n) it is possible to have transition to the next state (n+1) or to reset state. Transition to reset point happens when "RESET condition" is true; Transition to the next step happens when "NEXT condition" is true.

Interrupt is triggered when output/stop/continue state is reached.

Each state machine allows to implement gesture recognition in a flexible way, free-fall, wake-up, 4D/6D orientation, pulse counter and step recognition, click/double click, shake/double shake, face-up/face-down, turn/double turn:

Table 8. LIS3DSH state machines: sequence of state to execute an algorithm



SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

SPI slave timing values

	Parameter	Value (1) Min.	Value (1) Max.	Unit
	SPI clock cycle	100		ns
	SPI clock frequency		10	MHz
	CS setup time	6		

I²C - inter IC control interface

Subject to general operating conditions for Vdd and Top.

I²C slave timing values

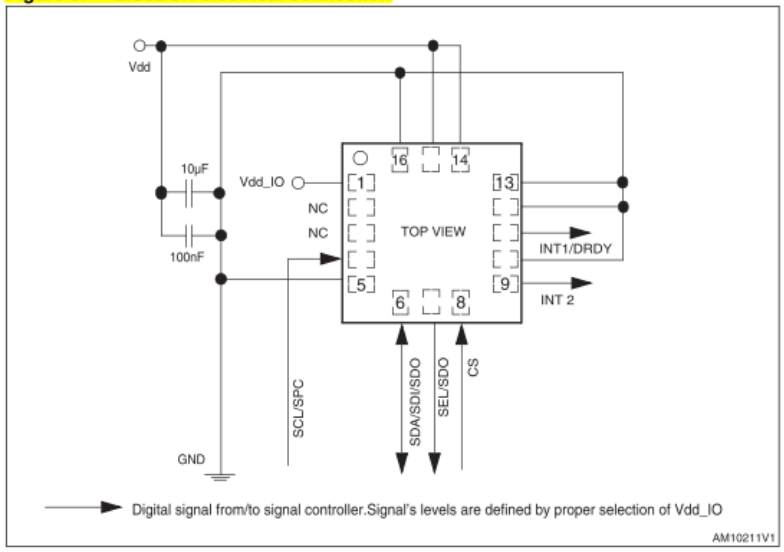
Parameter	I²C standard mode (1) Min.	I²C standard mode (1) Max.	I² fast mode (1) Min	I² fast mode (1) Max.	Unit
SCL clock frequency	0	100	0	400	kHz

Table 7. Absolute maximum ratings

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V

Application hints

Figure 5. LIS3DSH electrical connection

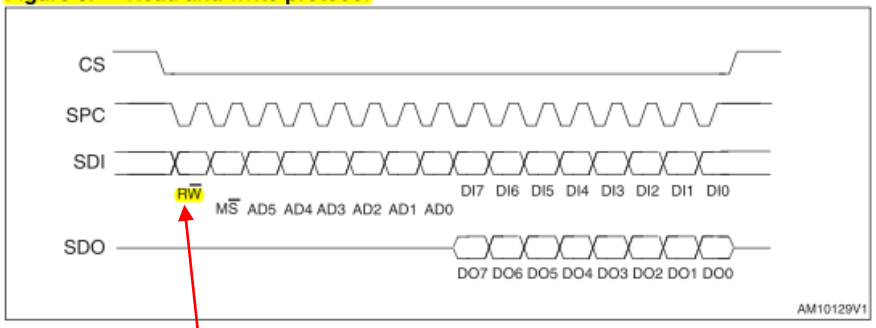


https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/LIS3DSH.pdf

VP 6 - STM32 CubeIDE, SPI in LIS3DSH

Gradiva

Figure 6. Read and write protocol



bit 0: RW bit. When 0, the data DI(7:0) is written into the device. When 1, the data DO(7:0) from the device is read. In the latter case, the chip drives **SDO** at the start of bit 8.

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

bit 8-15: data DI(7:0) (write mode). This is the data that is written into the device (MSb first).

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

8.3 WHO_AM_I (0Fh)

Who_AM_I register.

rozman 26. 04. 2022, 0..
0x3F

Table 19. WHO_AM_I register default value

0	0	1	1	1	1	1	1

```
// Config accelerometer
// Read WHOAMI register
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
outdata[0] = 0x0f | 0x80 ;  // read whoami
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
lis_id = indata[1];
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

// Write to CTRL register (enable 3 axes meaurements on 25Hz)
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
outdata[0] = 0x20 ;  // switch on axes
outdata[1] = 0x47 ;
HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
```

SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

SPI slave timing values

	Parameter	Value (1) Min.	Value (1) Max.	Unit
	SPI clock cycle	100		ns
	SPI clock frequency		10	MHz

Table 7. Absolute maximum ratings

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V

8.5 CTRL_REG4 (20h)

Control register 4.

rozman 26. 04. 2022, 0..
0x47 (25Hz, all axes on)

Table 22. Control register 4

ODR3	ODR2	ODR1	ODR0	BDU	ZEN	YEN	XEN

Table 23. CTRL_REG4 register description

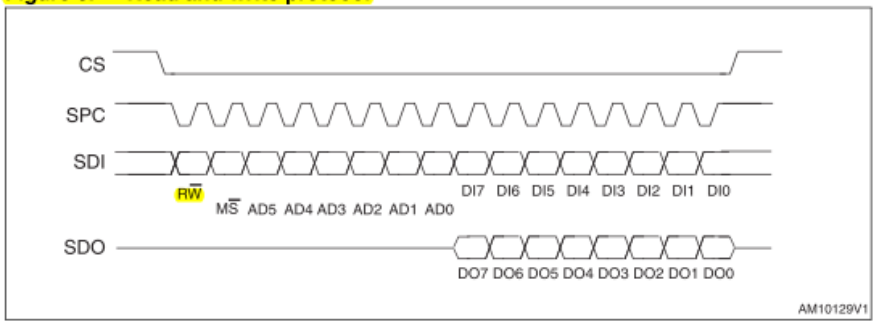
ODR 3:0	Output data rate and power mode selection. Default value:0000 (see *Table 24*)
BDU	Block data update. Default value:0 0:continuos update,1:output registers not updated until MSB and LSB reading)
Zen	Z axis enable. Default value:1 (0:Z axis disabled; 1:Z axis enabled)
Yen	Y axis enable. Default value:1 (0:Y axis disabled; 1:Y axis enabled)
Xen	X axis enable. Default value:1 0=X axis disabled; 1=X axis enabled

Table 24. CTRL4 ODR configuration

ODR3	ODR2	ODR1	ODR0	ODR selection
0	0	0	0	Power down
0	0	0	1	3.125 Hz
0	0	1	0	6.25 Hz
0	0	1	1	12.5 Hz
0	1	0	0	25 Hz

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/Documentation/STM32F407_Discovery_kit/LIS3DSH.pdf

Figure 6. Read and write protocol



bit 0: RW bit. When 0, the data DI(7:0) is written into the device. When 1, the data DO(7:0) from the device is read. In the latter case, the chip drives **SDO** at the start of bit 8.

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

bit 8-15: data DI(7:0) (write mode). This is the data that is written into the device (MSb first).

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

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 Hex	Register address Binary	Default	Comment
INFO1	r	0D	00001101	0010 0001	Information register 1
INFO2	r	0E	00001110	0000 0000	Information register 2
WHO_AM_I	r	OF	00001111	0011 1111	Who I am ID
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		

8.23 OUT_X (28h - 29h)

X-axis output register.

Table 49. OUT_X_L register default value

0	0	0	0	0	0	0	0

Table 50. OUT_X_H register default value

0	0	0	0	0	0	0	0

```
// 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];
```

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/Documentation/STM32F407_Discovery_kit/LIS3DSH.pdf

VP 6 - STM32 CubeIDE, SPI in LIS3DSH

main.c : dodana koda

Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;

// Global variables
uint8_t indata[2];
uint8_t outdata[2] = {0,0};
uint8_t lis_id;
int8_t AccelX;
int8_t AccelY;
int8_t AccelZ;

HAL_StatusTypeDef SPIStatus;

/* USER CODE END PV */
```

Glavna zanka

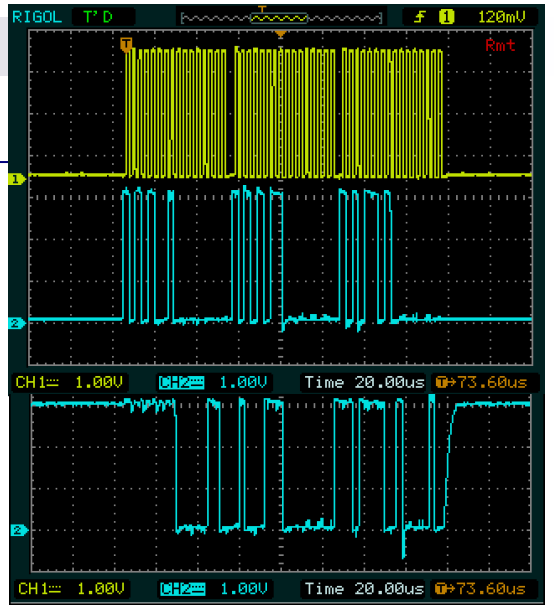
```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {

// 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];

…

snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d PWM-Freq:%d PWM-Period:%d
Accel[ID:%02x] X:%04d Y:%d
Z:%04d\r\n",Counter++,KeyState,Duty,NoteFreq,NotePeriod,lis_id,AccelX,AccelY,AccelZ);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    /* USER CODE END WHILE */
```

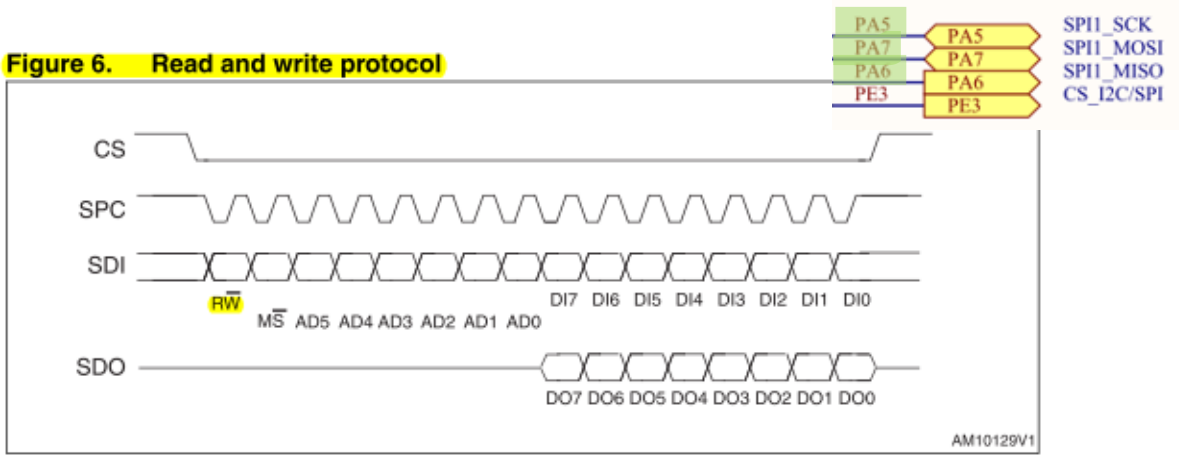
Inicializacija

```
/* 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
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2,
HAL_MAX_DELAY);
  lis_id = indata[1];
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

  HAL_Delay(500);

  // Set CTRL register 0x47 -> [0x20]
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  outdata[0] = 0x20 ;  // switch on axes
  outdata[1] = 0x47 ;
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2,
HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

  HAL_Delay(500);
  outdata[1] = 0x00 ;

  /* USER CODE END 2 */
```

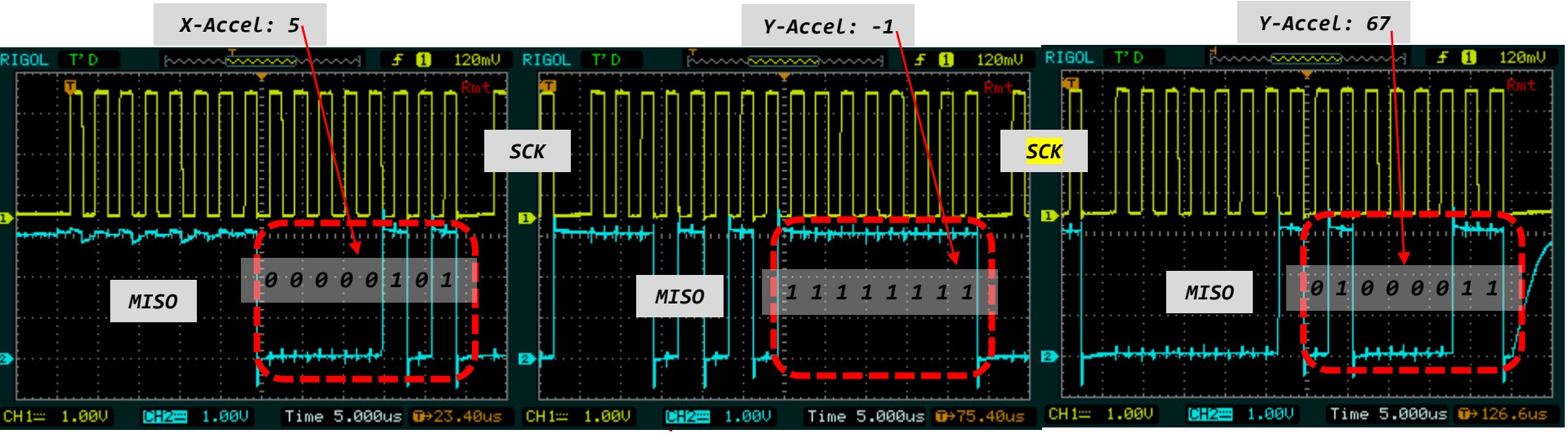
https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32_SPI_LIS302DL_Basic

VP 6 - STM32 CubeIDE, SPI in LIS3DSH - Osciloskop

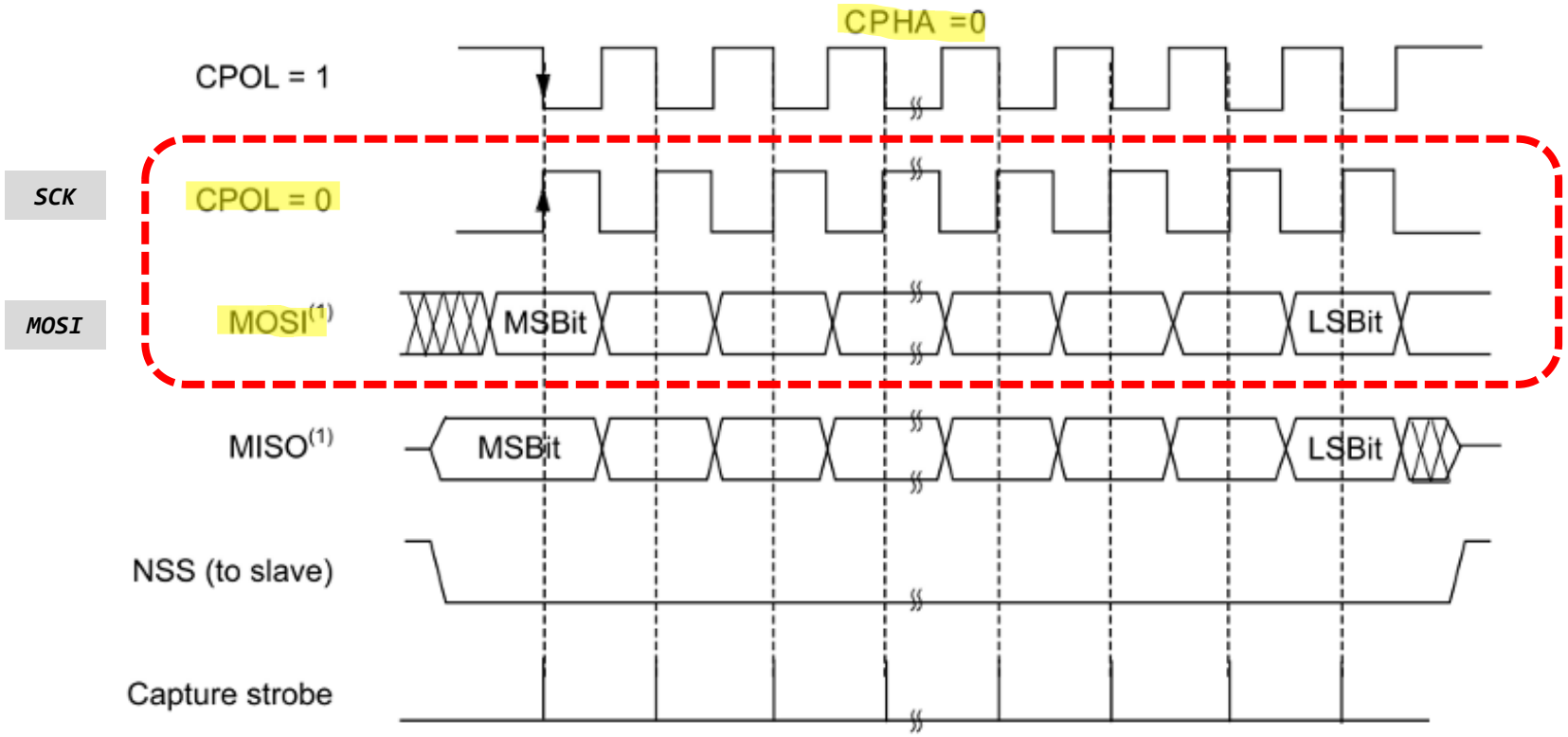





```
Hello World [3530]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0066
Hello World [3531]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0067
```



Določite bitno hitrost prenosa in ugotovite vsebino signala SPI2 z nastavitvami: CPOL=0, CPHA=0, naprava LIS3DSH, …



Laboratorijska vaja 10
STM32H7 – Generator signalov

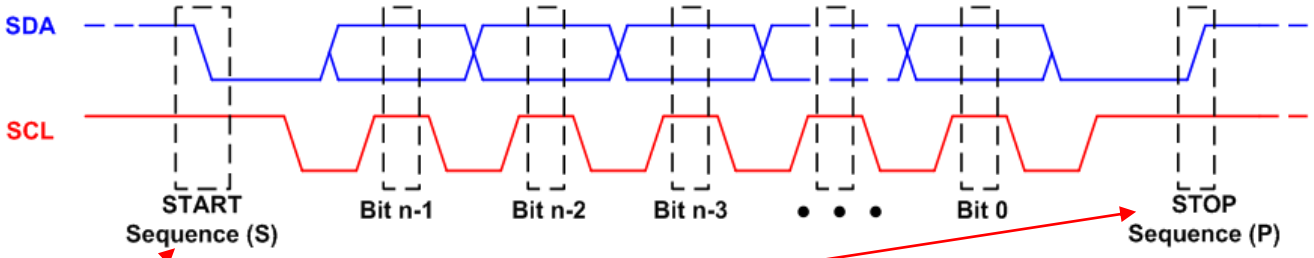
Osvežitev STM32H7

Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

Izhodišča

- Signali na linijah



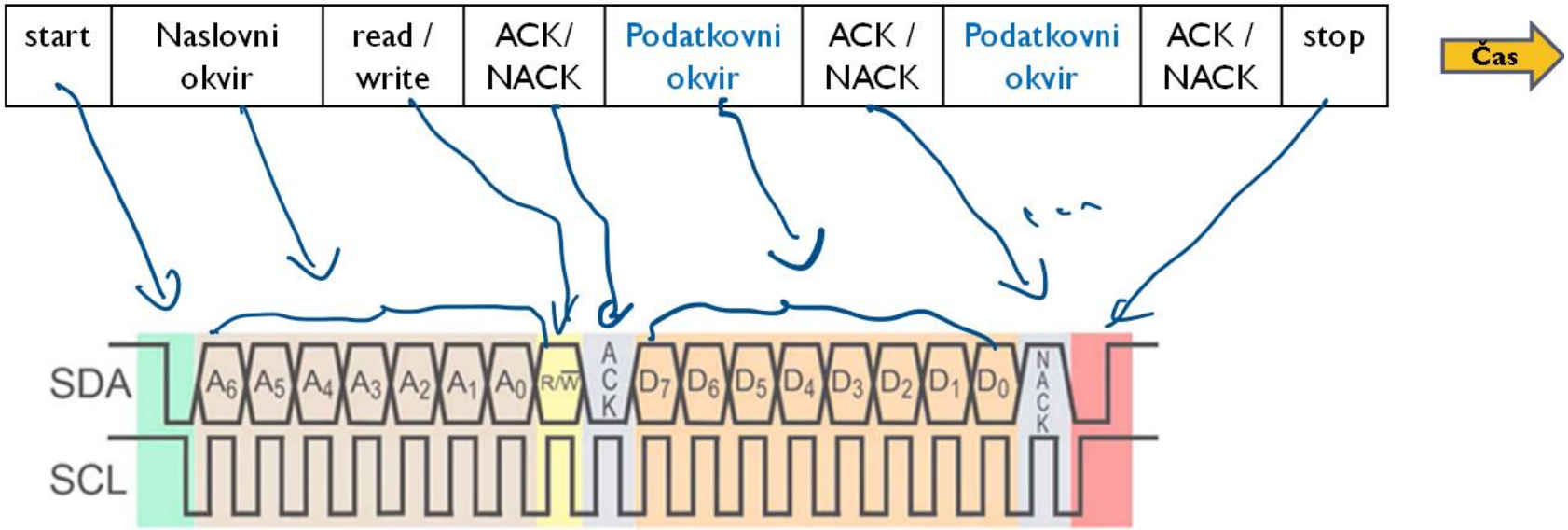
- Dodatna bita (vedno generira master) :
  - pogoj **start** – linija SDA preklopi iz 1 v 0 preden linija SCL preklopi iz 1 v 0
  - pogoj **stop** - linija SDA preklopi iz 0 v 1 potem ko linija SCL preklopi iz 0 v 1

- Primer komunikacije

start	Naslovni okvir	read / write	ACK/ NACK	Podatkovni okvir	ACK / NACK	Podatkovni okvir	ACK / NACK	stop

- 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/noacknovledge‘. Če je bil naslovni ali podatkovni okvir uspešno prejet je pošiljatelju vrnjen bit ACK, sicer NACK.

Izhodišča



Dodatni biti med okvirji:

- **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/noacknovledge‘. Če je bil naslovni ali podatkovni okvir uspešno prejet je pošiljatelju vrnjen bit ACK, sicer NACK.
- **start** – linija SDA preklopi iz 1 v 0 preden linija SCL preklopi iz 1 v 0
- **stop** - linija SDA preklopi iz 0 v 1 potem ko linija SCL preklopi iz 0 v 1

Primer I2C komunikacije STM32H7 - Touch

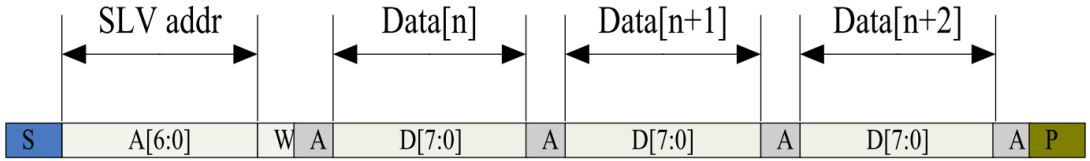


Figure 2-5 I2C master write, slave read

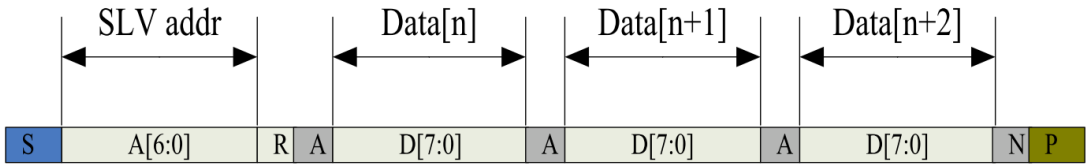
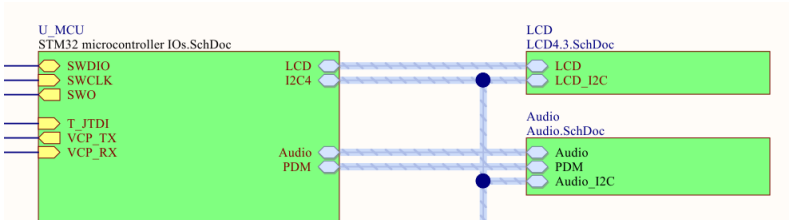


Figure 2-6 I2C master read, slave write



FocalTech

FT5336GQQ

True Multi-Touch Capacitive Touch Panel Controller

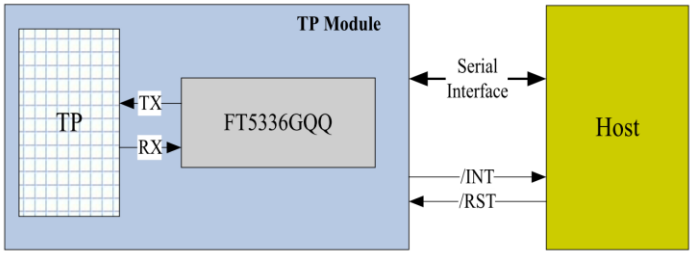


Figure 2-3 Host Interface Diagram

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	1st Event Flag				1st Touch X Position[11:8]				R
04h	TOUCH1_XL	1st Touch X Position[7:0]								R
05h	TOUCH1_YH	1st Touch ID[3:0]				1st Touch Y Position[11:8]				R
06h	TOUCH1_YL	1st Touch Y Position[7:0]								R
A8h	ID_G_FT5201ID	CTPM Vendor ID								R

Primer I2C komunikacije
STM32H7 - Touch

```
// 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;
}
```

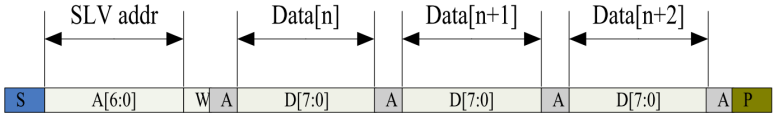


Figure 2-5 I2C master write, slave read

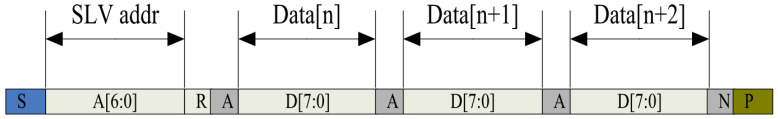


Figure 2-6 I2C master read, slave write

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	1st Event Flag				1st Touch X Position[11:8]				R
04h	TOUCH1_XL	1st Touch X Position[7:0]								R
05h	TOUCH1_YH	1st Touch ID[3:0]				1st Touch Y Position[11:8]				R
06h	TOUCH1_YL	1st Touch Y Position[7:0]								R
A8h	ID_G_FT5201ID	CTPM Vendor ID								R

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

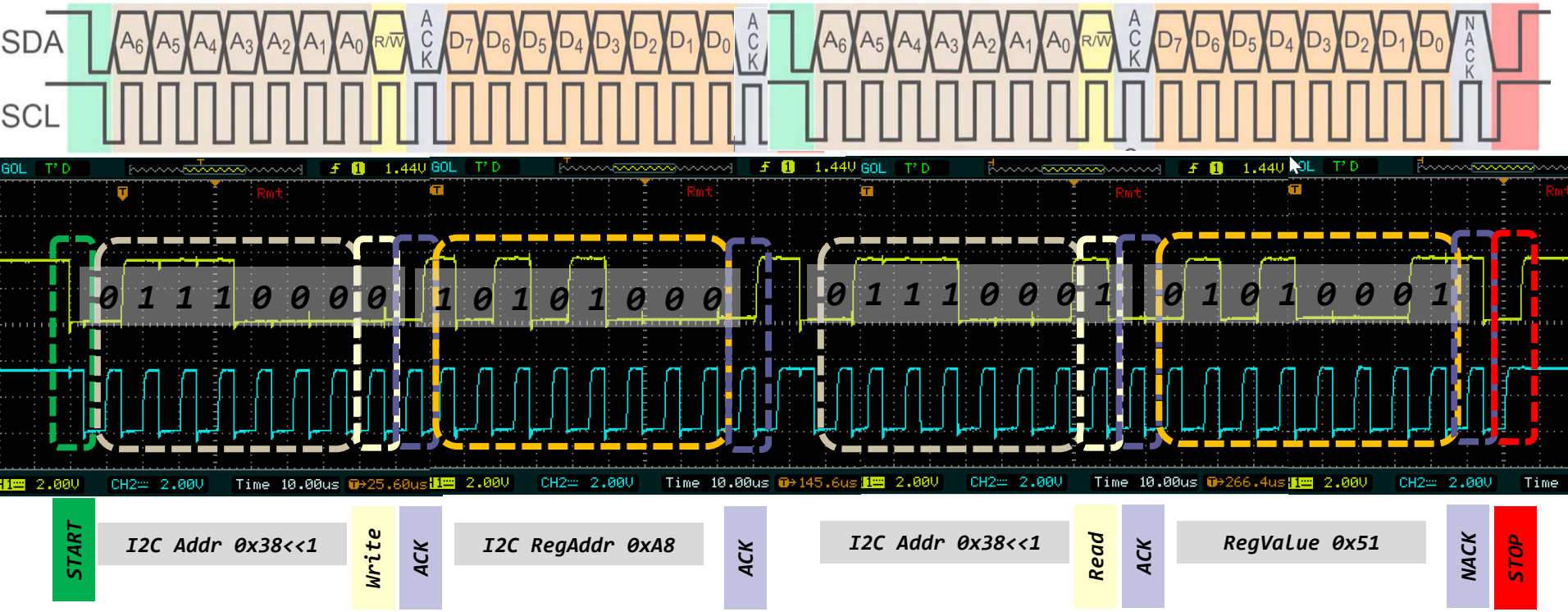
STM32H7

Primer I2C komunikacije

STM32H7 - Touch

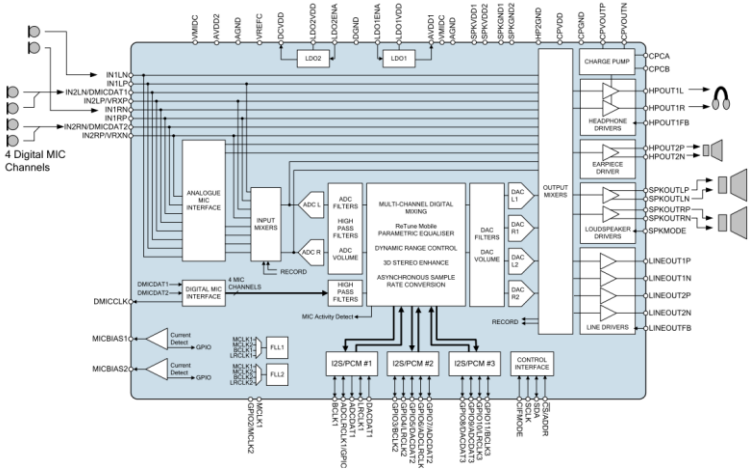
I2C branje

```
// 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);
```



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

Primer I2C komunikacije STM32H7 - Audio



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

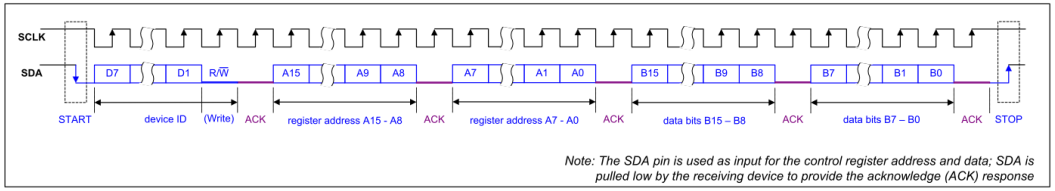


Figure 72 Control Interface 2-wire (I2C) Register Write

[Brez naslova]

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

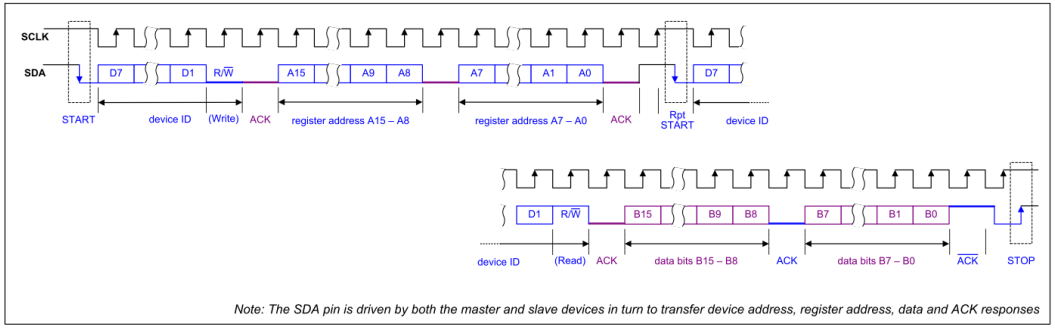


Figure 73 Control Interface 2-wire (I2C) Register Read

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/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	SPKOUTR_ENA	SPKOUTL_ENA	HPOUT2_ENA	0	HPOUT1L_ENA	HPOUT1R_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

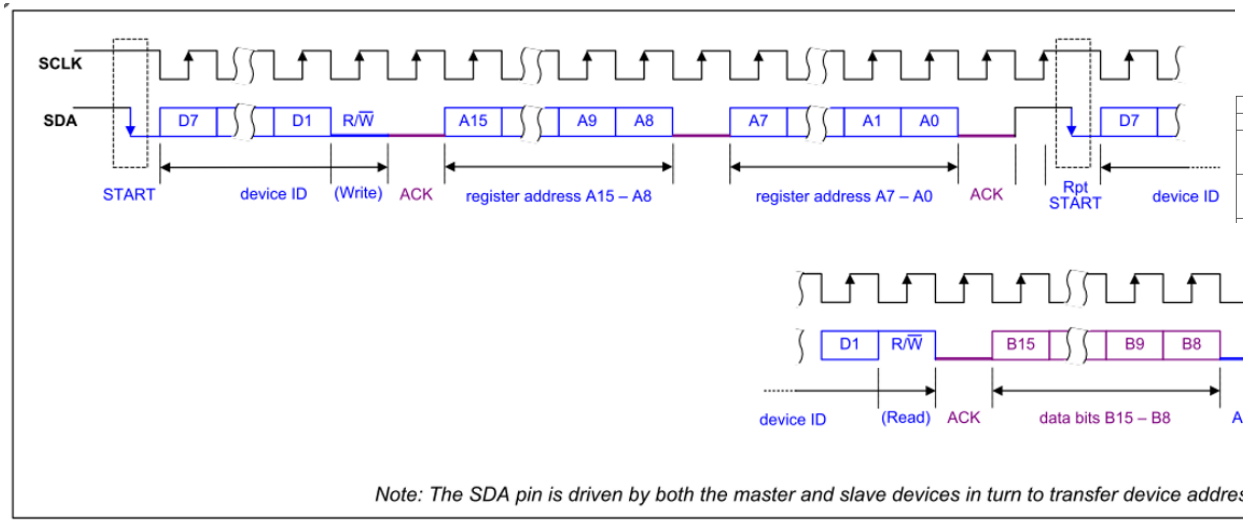
CIRRUS LOGIC

WM8994

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);
```



REGISTER MAP

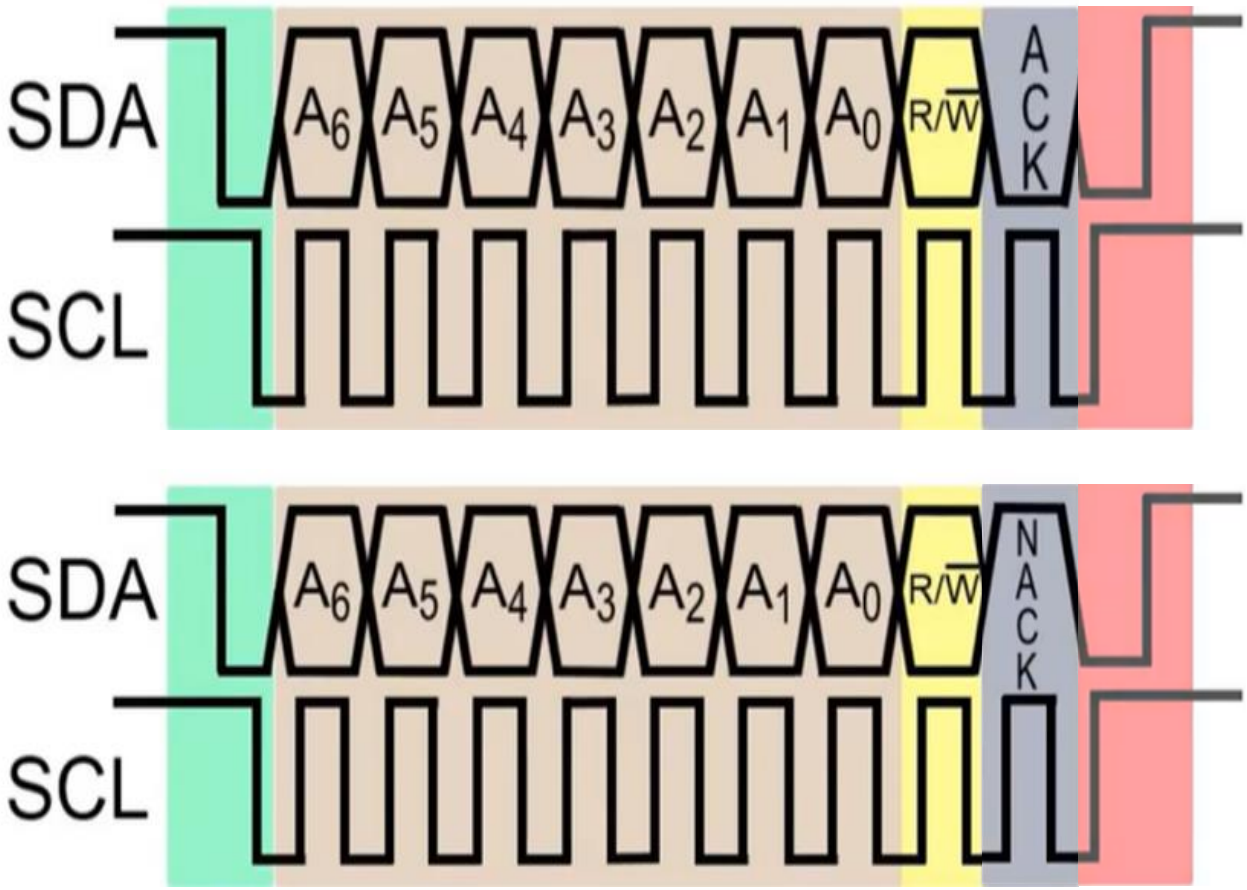
The WM8994 control registers are listed below. Note that only the register addresses described here should be accessed; writing to other addresses may result in undefined behaviour. Register bits that are not documented should not be changed from the default values.

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	SPKOUTR_ENA	SPKOUTL_ENA	HPOUT2_ENA	0	HPOUT1L_ENA	HPOUT1R_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

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

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

Določite bitno hitrost prenosa in ugotovite vsebino signala I2C4:



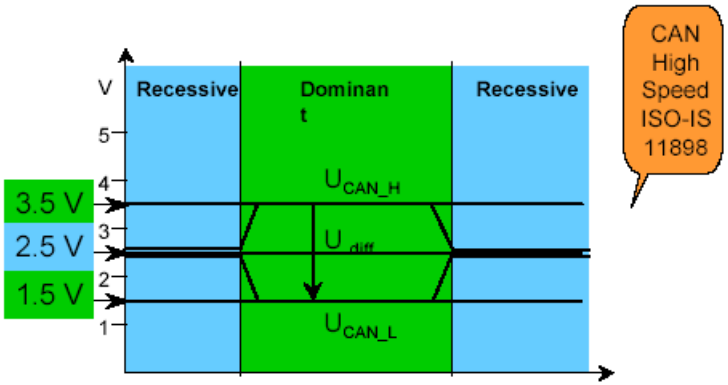
Laboratorijska vaja 10
STM32H7 – Generator signalov

Osvežitev STM32H7

Izzivi:

- UART PB14
- PWM PA3
- SPI PD3(SCK), PI3 (MOSI)
- I2C PD12(SCL), PD13(SDA)
- CANBUS CN1 (FDCAN1) CAN-L, CAN-H

CANbus napetostni nivoji ISO-IS 11898

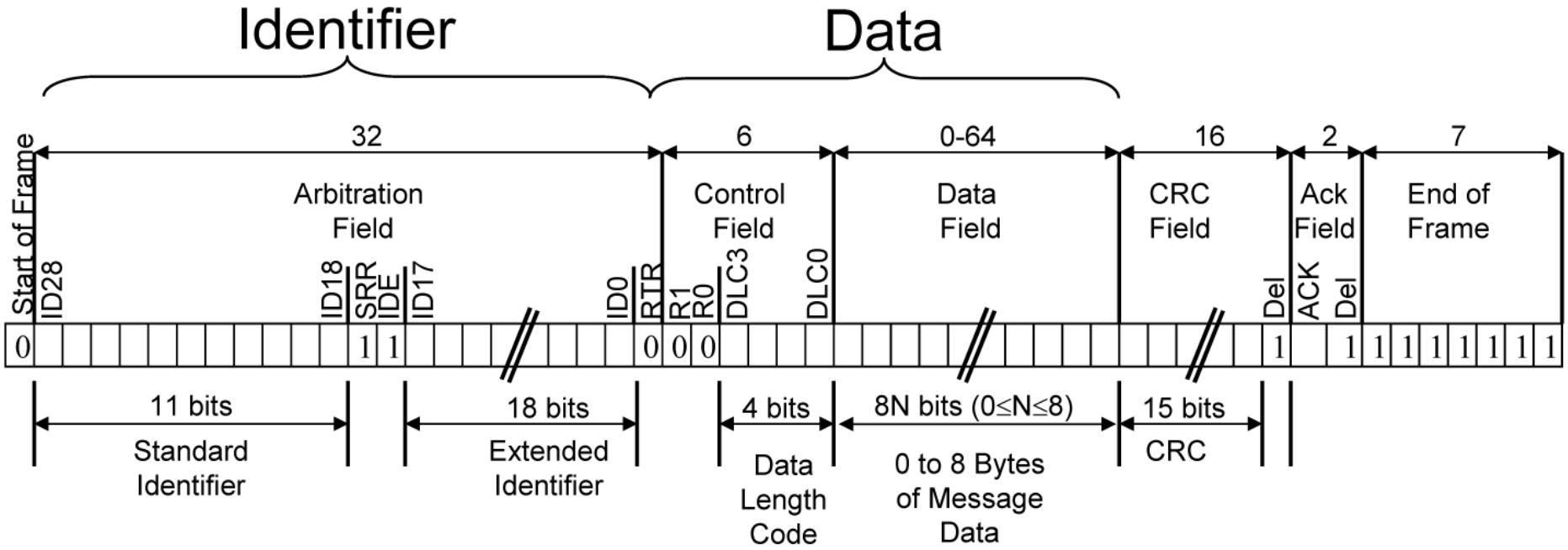


•Recesivni bit „1“:
- obe liniji na približno 2.5V
- diferencialna napetost CAN_H in CAN_L ≈ 0 V

•Dominantni bit „0“:
- CAN_H na pribl. 3.5 V in CAN_L pribl. 1.5 V
- diferencialna napetost CAN_H in CAN_L ≈ 2 V

Format sporočila

- Vsako sporočilo ima ID, podatke in dodatke
- ID - 11 ali 29 bitov
- Data - do 8 bajtov
- Dodatki - start (SOF), CRC, ACK, end (EOF)



Format sporočila

CAN vs. RS-485: Why CAN Is on the Move

By Robert Gee, Executive Business Manager, Core Products Group, Maxim Integrated

- Recesivni bit „1“:
  - obe liniji na približno 2.5V
- Dominantni bit „0“:
  - CAN_H na pribl. 3.5 V in CAN_L pribl. 1.5 V

Field Name	Bit Length	Description
SOF	1	Start of frame
Identifier (green)	11/29; 12/32	Represents the message priority (11 or 29 bits for standard CAN and extended CAN; 12 or 32 bits for CAN-FD)
RTR (blue)	1	Remote transmission request
IDE	1	Identifier extension bit
r0	1	Reserved bit for future protocol expansion
DLC (yellow)	4/8/9	Code for number of data bytes (4-bit for standard CAN; 8 or 9 bits for CAN-FD)
Data Field (red)	0–64 (0–8 bytes); 0–512 (0–64 bytes)	Data to be transmitted (0–8 bytes for standard CAN; 0–64 bytes for CAN-FD)
CRC	15	Cyclic redundancy check
CRC Delimiter	1	Assigned recessive (1)
ACK slot	1	Dominant bit if error-free message; recessive to discard errant message
ACK Delimiter	1	Acknowledgement delimiter
EOF	7	End of frame

Table 1. CAN Message Data-Frame Format

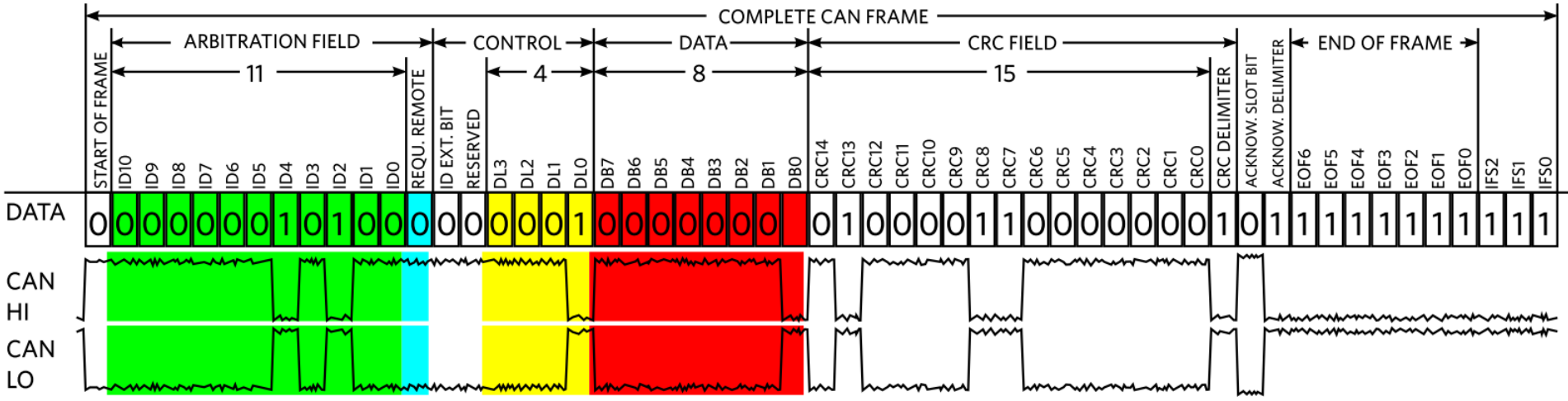
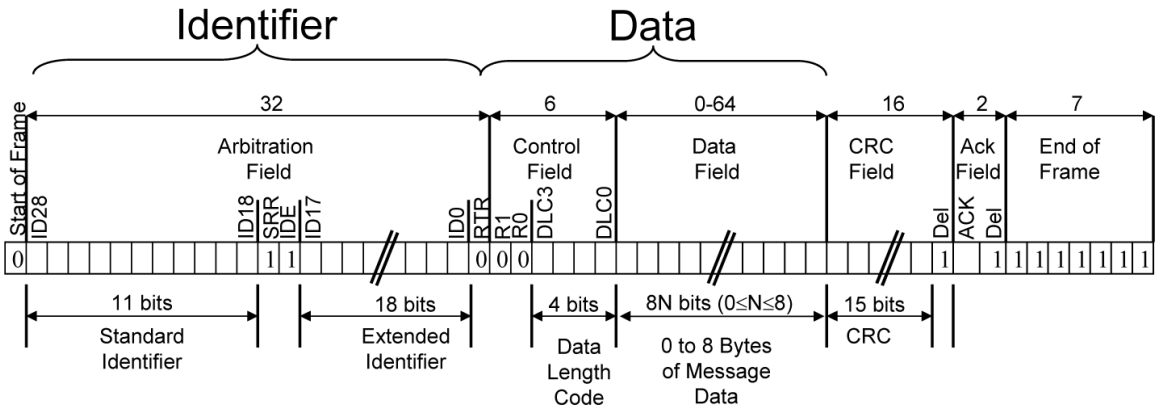


Figure 4. CAN Message Data-Frame Format

Določite bitno hitrost prenosa in določite vsebino signala CANBUS, ki se prenaša ob nastavitvah: 11b ID = 0x555, 2 bajta 0xCC, bit-stuff (po 5 enakih bitih), …

Recesivni bit „1“:
- obe liniji na približno 2.5V

•Dominantni bit „0“:
- CAN_H na pribl. 3.5 V in CAN_L pribl. 1.5 V



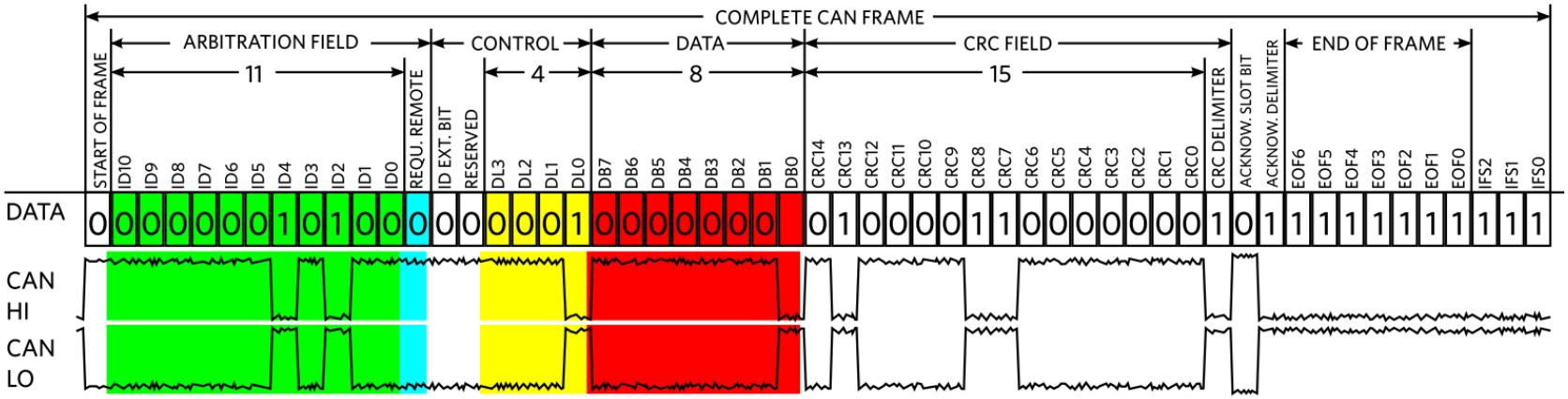
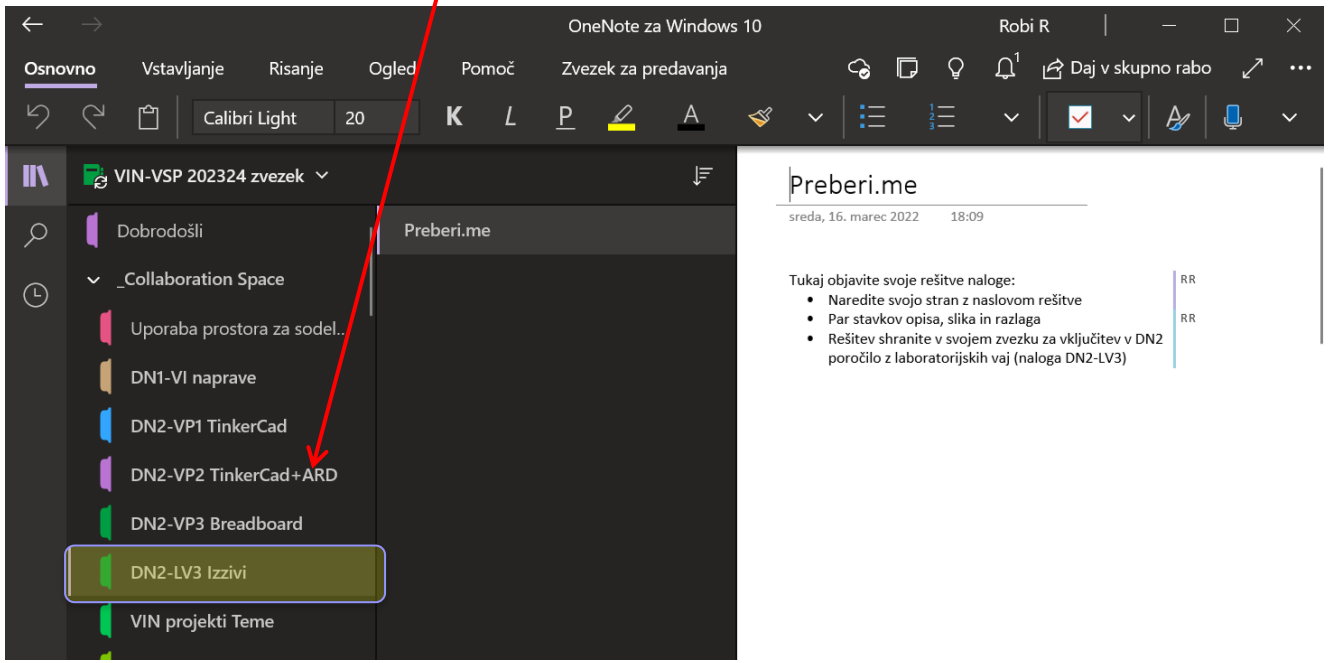


Figure 4. CAN Message Data-Frame Format

Očesni vzorec, STM32H7 izziv – DN2-LV3:

- Spada v sklop poročila z LAB vaj
- Opišite reševanje STM32H7 izzivov
- Objavite v OneNote delovnem zvezku
  - _Prostor za sodelovanje, razdelek DN2-LV3 Izzivi



Očesni vzorec, STM32H7 izziv – DN2-LV3:

LV3-Meritve-očesnega-vzorca-in-ostalih-signalov

LV3-1,2,3,4-Meritve-očesnega-vzorca

LV3-1-Meritev-časa-potovanja
Izmerite čas potovanja signala po kablu (opišite postopek in rezultat). Meritev izvedite ob zaključenih linijah. Določite tudi dolžino linije kasseda natančno.

LV3-2-Meritev-strmine-vzpona/padca-signala
Izmerite in izračunajte strmino prehoda signala (»slew rate«) iz RS232 in CMOS oddajnika. Preverite, ali je vrednost pri RS232 znotraj omejitve maksimalne hitrosti spremembe po RS232 standardu (30V/µs).

LV3-3-Meritev-očesnega-vzorca-ob-zaključeni-liniji
Pri pravilno zaključeni liniji s FRI-SMS (izhod SPI) generirajte psevdonaključni pravokotni signal in opazujte sliko očesnega vzorca na izhodu linije pri različnih frekvencah signala (glejte tabelo v spodaj), ekranske slike podajte za „prelomno" frekvenco):

a) Iz oddajnika RS-232

b) Iz oddajnika CMOS

LV3-4-Meritev-očesnega-vzorca-ob-nezaključeni-liniji
Pri nezaključeni liniji (na vhodu manjša upornost in izhodu odprte sponke) s FRI-SMS (izhod SPI) generirajte psevdonaključni pravokotni signal in opazujte sliko očesnega vzorca na izhodu linije pri različnih frekvencah signala (izpolnite tabelo v spodaj), ekranske slike podajte za „prelomno" frekvenco):

a) Iz oddajnika RS-232

b) Iz oddajnika CMOS

Nasveta za podnaloge (a) in (b):
- → *meritve izvedite z naračanjem frekvence signala in predvsem določite mejo, kjer očesni vzorec postane nezprejemljiv (ekranske slike podajte za najvišjo frekvenco, kjer je očesni vzorec še sprejemljiv). Nadaljnje višanje frekvence ni več potrebno.*
- → *skoliker napetostni nivo ne ustreza že pri prvi meritvi (f=187.5kHz), potem s potenciometroma subjektivno popravite napetostne nivoje, da bodo bolj vidni (ustrezni), določite napetostne meje spodnjega in zgornjega nivoja (»Cursor«) in nadaljujete meritev s povečevanjem frekvence (seveda vse nastavitve in to posebnost omenite tudi v poročilu)*

Prelom strani

Zaključna tabela izvedenih meritev – STM32H7:

Meritev očesnega vzorca ····· Merilna linija št. [1–4] ··· vrsta kabla									
Linija	Zaključena linija				Nezaključena linija				
Oddajnik	CMOS		RS232		CMOS		RS232		
Frekvenca [kHz]	VHOD+	IZHOD-	VHOD+	IZHOD-	VHOD+	IZHOD-	VHOD+	IZHOD-	Komentar
187.5									
375									
750									
1500									
3000			[10]						
Fmax									
Komentar									

Zaključna tabela izvedenih meritev – FRI SMS:

Meritev očesnega vzorca ····· Merilna linija št. [1–4] ··· vrsta kabla									
Linija	Zaključena linija				Nezaključena linija				
Oddajnik	CMOS		RS232		CMOS		RS232		
Frekvenca [kHz]	VHOD+	IZHOD-	VHOD+	IZHOD-	VHOD+	IZHOD-	VHOD+	IZHOD-	Komentar
200									
480									
600									
800									
1200			[10]						
1600									
2400									
4800									
Fmax									
Komentar									

[Brez naslova]

Pomen morebitnih oznak:
+ ··· očesni vzorec zadošča obema kriterijema
+ ··· očesni vzorec zadošča vizualnemu kriteriju (npr. odprtost očesa in ne formalnim napetostnim nivojem)
- → očesni vzorec ne ustreza

*···· Meritve na vhodu niso obvezne (lahko kot dodatna naloga)

***Neobvezno/dodatno**: lahko izvedete meritve še na drug(em'ih) kablih. Za te dodatne meritve lahko dodate manj (le nekaj izbranih) ekranskih slik in nekaj razlage dobljenih rezultatov.*

Prelom strani

LV3-5,6,7,8,9,10-Meritve-ostalih-signalov·–·STM32H7·Signal·generator·–·izziv

Izberite in rešite vsaj 3 izzive spodaj.

LV3-5-Meritve-RS232-napetostnih-nivojev
Pri pravilno zaključeni liniji s programom UART (izhod UART) generirajte asinhronski signal in izmerite ter preverite veljavnost napetostnih nivojev iz oddajnika RS-232.

LV3-6-Meritve-za-ugotovitev-bitne-hitrosti-in-poslanih-znakov-z-UART-enoto-in-RS232
Določite bitno hitrost prenosa in ugotovite, kateri znaki se prenašajo ob predpostavki nastavitve 8N1 (8 podatkovnih bitov, brez paritetnega bita, 1 stop bit). Določite ASCII kode znakov, ki se prenašajo. Prav tako ocenite maksimalno možno število znakov, ki bi lahko bili poslani po RS232 liniji in potem še ocenite dejansko število prenesenih znakov.

Odgovore utemeljite z ekranskimi slikami in razlago poti do rezultatov.

Namig za meritve bitne hitrosti: priporočljivo je, da najprej ugotovite vsebino poslanih znakov in potem lažje določite bitne intervale. Zaradi tresenja (»jitter«) je bolj priporočljivo izvesti meritev na daljšem zaporedju bitov za določitev bitne hitrosti.

LV3-7-Meritve-PWM-signala
Določite periodo, frekvenco PWM signala in ustrezno glasbeno noto.

LV3-8-Meritve-SPI-signala
Določite bitno hitrost prenosa in ugotovite vsebino signala SPI2 z nastavitvami: CPOL=0, CPHA=0, naprava LIS3DSH, …

LV3-9-Meritve-I2C-signala
Določite bitno hitrost prenosa in ugotovite vsebino signala I2C4.

LV3-9-Meritve-CANBUS-signala
Določite bitno hitrost prenosa in določite vsebino signala CANBUS, ki se prenaša ob nastavitvah: 11b ID = 0x555, 2 bajta 0xCC, bit-stuff (po 5 enakih bitih), …

Prelom strani