



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

ELEKTRONINIŲ SISTEMŲ KATEDRA

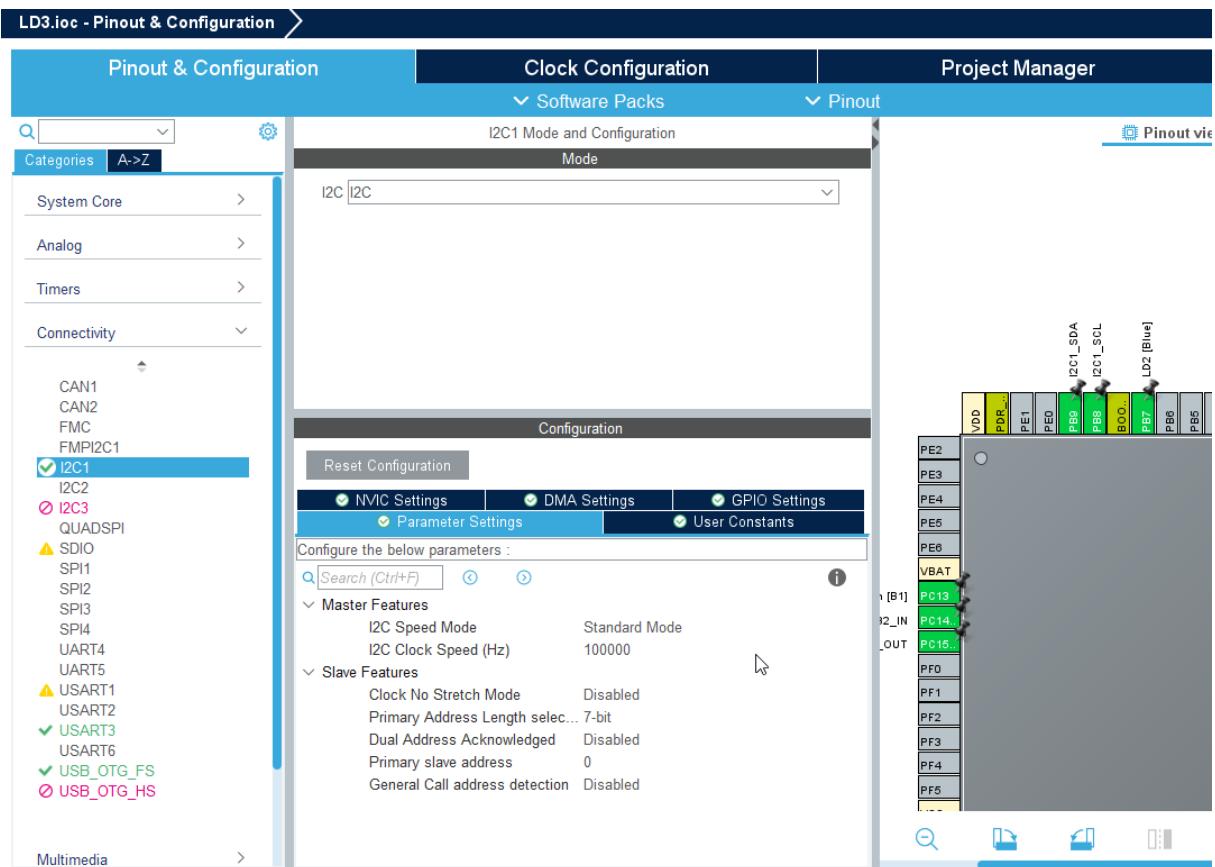
LABORATORINIS DARBAS 3

Įterptinių sistemų inžinerija

Atliko: EKSfm-24 gr. Ignas Malinauskas

Tikrino: dr. Eldar Šabanovič

1. Projekto konfigūracija (be pull-up)



2. I²C užduotis su MCP4725

Prie STM32F446ZE I²C kaiščių D15(PB8) SCL ir D14(PB9) SDA prijungta 12-bit DAC MCP4725. Programoje paduota 2048 reikšmę (~1.6V) Fast-Mode formatu, kas yra 2 baitų komanda. Duomenys perduodami per tx_buffer masyvą, kur pirmam elemente iš 12 duomenų baitų paliekami 4 ir nustumiami į dešinę, paliekant viršu (MSB) komandų bitams. Likę 8 duomenų bitai išrašomi į 2 tx_buffer elementą.

1 baitas

- bit7,6 = 0, 0; reiškia greito režimo formatą
- bit5,4 - PD1, PD0 = 00; nurodo išvesties režimą į normalų
- bit3-0 – D11-D8 = duomenų bitai

2 baitas

- bit7-0 – D7-D0 = duomenų bitai

Eilutėje HAL_I2C_Master_Transmit(&hi2c1, 0x60<<1, tx_buffer, 2, 1000); 0x60 bitai nurodo DAC adresą, kur nurodo kuris įrenginys dabar turės klausyti.

3. Programos kodas it Tera Term rezultatai

DAC išvedamą reikšmę nuskaito ADC ir rezultatas išvedamas per UART.

The screenshot shows a development environment for STM32 C/C++ applications. The left side displays the source code for `main.c`, which includes I2C and ADC initialization, a main loop with DAC value transmission, and ADC reading/printing. The right side shows a terminal window titled "COM4 - Tera Term VT" displaying a series of I2C transactions where the ADC value is read and printed as 2045.

```
118     MX_I2C1_Init();
119     MX_ADC1_Init();
120
121     /* USER CODE BEGIN 2 */
122     uint16_t dac_value;
123     uint16_t value;
124
125     /* USER CODE END 2 */
126
127     while (1)
128    {
129        //half reference voltage
130        dac_value = 2048;
131        //shift right, get upper portion, extract 4 bits
132        tx_buffer[0] = ((dac_value >> 8) & 0x0F);
133        tx_buffer[1] = (uint8_t)(dac_value & 0x00FF);
134        HAL_StatusTypeDef ret = HAL_I2C_Master_Transmit(&i2c1, 0x60<<1, tx_buffer, 2, 1000);
135
136        if(ret == HAL_OK){
137            HAL_ADC_Start(&hadc1);
138
139            value = (uint16_t)HAL_ADC_GetValue(&hadc1) ;
140            printf("I2C OK, ADC value: %d\r\n",value);
141            HAL_Delay(100);
142        }
143
144     /* USER CODE END WHILE */
145
146     /* USER CODE BEGIN 3 */
147 }
```

	File	Edit	Setup	Control	Window	Help
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2048				
12C OK,	ADC value:	2046				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2047				
12C OK,	ADC value:	2048				
12C OK,	ADC value:	2046				
12C OK,	ADC value:	2048				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2043				
12C OK,	ADC value:	2048				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2045				
12C OK,	ADC value:	2047				
12C OK,	ADC value:	2048				
12C OK,	ADC value:	2044				
12C OK,	ADC value:	2043				
12C OK,	ADC value:	2045				

The screenshot shows a development environment for a STM32 microcontroller. The main window displays the source code for `main.c`, which includes initialization for I2C and ADC, a loop that transmits data over I2C, starts an ADC conversion, and prints the result. The code uses HAL libraries for the peripherals.

The right side of the interface features a terminal window titled "COM4 - Tera Term VT" showing the output of the program. The output consists of a series of I2C transactions where the device sends an 8-bit value (0x00FF) and receives an ADC value (1647 mV) back. This pattern repeats 20 times.

At the bottom, there are tabs for "Console", "Problems", "Tasks", and "Properties". A status bar at the bottom indicates the session was terminated by the user.

```
MX_I2C1_Init();
MX_ADC1_Init();
/* USER CODE BEGIN 2 */
uint16_t dac_value;
uint16_t value;
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    //half reference voltage
    dac_value = 2048;
    //shift right, get upper portion, extract 4 bits
    tx_buffer[0] = ((dac_value >> 8) & 0x0F);
    tx_buffer[1] = (uint8_t)(dac_value & 0x00FF);
    HAL_StatusTypeDef ret = HAL_I2C_Master_Transmit(&i2c1, 0x60<<1, tx_buffer, 2, 1000);

    if(ret == HAL_OK){
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 10);
        value = (uint16_t)HAL_ADC_GetValue(&hadc1) * 3300 / 4095;
        printf("I2C OK, ADC value: %d mV\r\n",value);
        HAL_Delay(100);
    }
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
}

File Edit Setup Control Window Help
```

I2C OK, ADC value:	1647 mV
I2C OK, ADC value:	1648 mV
I2C OK, ADC value:	1647 mV
I2C OK, ADC value:	1645 mV
I2C OK, ADC value:	1647 mV
I2C OK, ADC value:	1648 mV
I2C OK, ADC value:	1651 mV
I2C OK, ADC value:	1648 mV
I2C OK, ADC value:	1647 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1648 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1648 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1647 mV
I2C OK, ADC value:	1650 mV
I2C OK, ADC value:	1654 mV
I2C OK, ADC value:	1650 mV

4. Sujungimas

