МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет
по лабораторной работе №4
на тему «Аналогово-цифровой преобразователь микроконтроллера STM32»

Дисциплина: ПМК
Группа: 21ПИ1
Выполнил: Гусев Д. А.
Количество баллов:
Дата сдачи:
Принял: Хворостухин С. П .

2024

1 Цель работы: Ознакомиться с программными средствами работы с контроллером аналогово-цифрового преобразования в микроконтроллерах STM32.

2 Задание на лабораторную работу.

2.1 Получить вариант задания у преподавателя.

2.2 Создать проект в среде Keil uVision5 для микроконтроллера STM32F103RB.

2.3 Выбрать программные компоненты: CMSIS/Core, Device/Startup, Device/StdPeriph Drivers/ADC, Device/StdPeriph Drivers/Framework, Device/StdPeriph Drivers/GPIO, Device/StdPeriph Drivers/RCC; Device/StdPeriph Drivers/TIM; Device/StdPeriph Drivers/USART.

2.4 Выполнить настройку режима отладки для проекта.

2.5 Разработать файл сценария.

2.6 Разработать программу согласно задания.

2.7 Выполнить симуляцию разработанной программы.

2.8 4Зафиксировать результаты функционирования программы: настройки аппаратных средств; содержимое терминала последовательного интерфейса; параметры сигналов.

2.9 Сделать выводы по проделанной работе и оформить отчет.

3 Выполнение лабораторную работы:

3.1 Был получен вариант задания – 8. Данные для варианта заданя представлены в таблице 1.

Таблица 1 — Вариант задания №8

Номер варианта	Номер канала АЦП	Закон изменения входного сигнала	Начальное значение температуры	Закон изменения температуры
8	8	Пилообразный	+60°C	-0,2°C/c

3.2 Был создан проект в среде Keil uVision5 для микроконтроллера STM32F103RB. Результат представлен на рисунке 1.

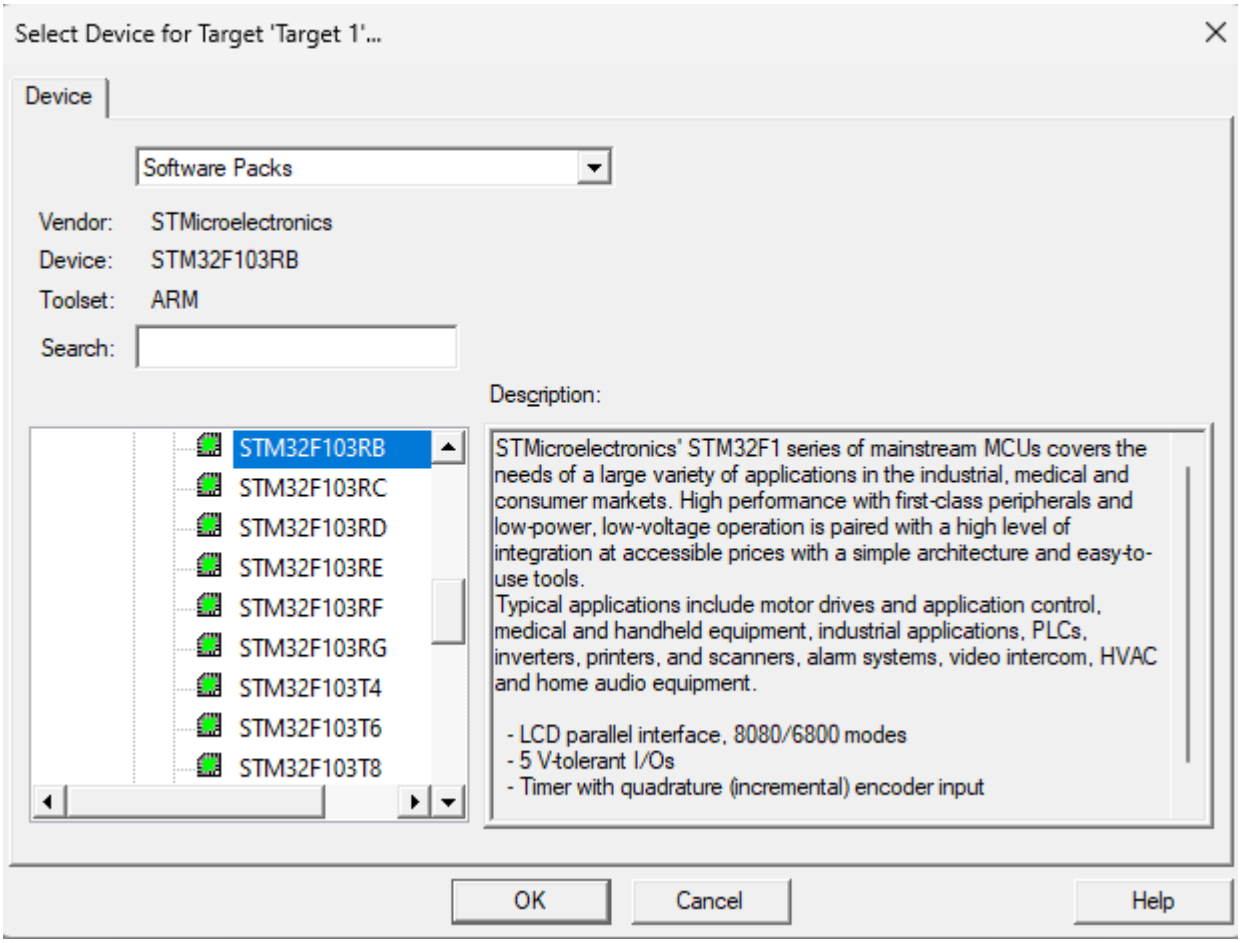

Рисунок 1 — Создание проекта для микроконтроллера STM32F103RB

3.3 Были выбраны программные компоненты: CMSIS/Core, Device/Startup, Device/StdPeriph Drivers/Framework, Device/StdPeriph Drivers/GPIO, Device/StdPeriph Drivers/RCC; Device/StdPeriph Drivers/TIM; Device/StdPeriph Drivers/USART. Результат представлен на рисунке 2.

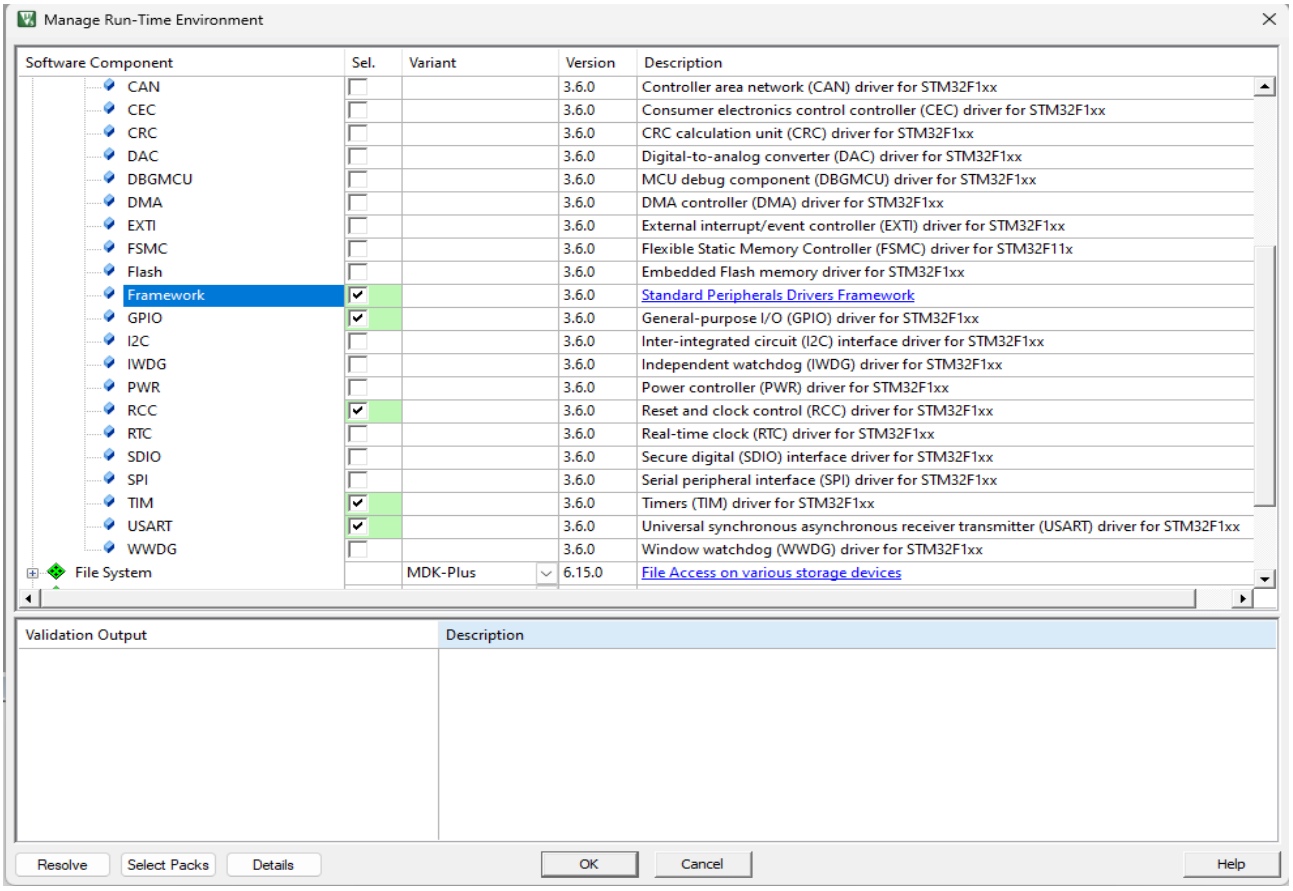

Рисунок 2 — Выбор компонентов

3.4 Была выполнена настройка режима отладки для проекта. Для этого в папке проекта был создан файл MAP.ini со следующим содержанием:

```
MAP 0x40000000, 0x47FFFFFF READ WRITE;
```

Далее была открыта вкладка «Option for Target...» затем была открыта вкладка «Debug», был включен переключатель «UseSimulator». В поле «DialogDLL» было записано DARMSTM.DLL, в поле «Parameter» было записано: - pSTM32F103RB. Был установлен путь к файлу MAP.ini в поле «InitializationFile». Результат представлен на рисунке 3.

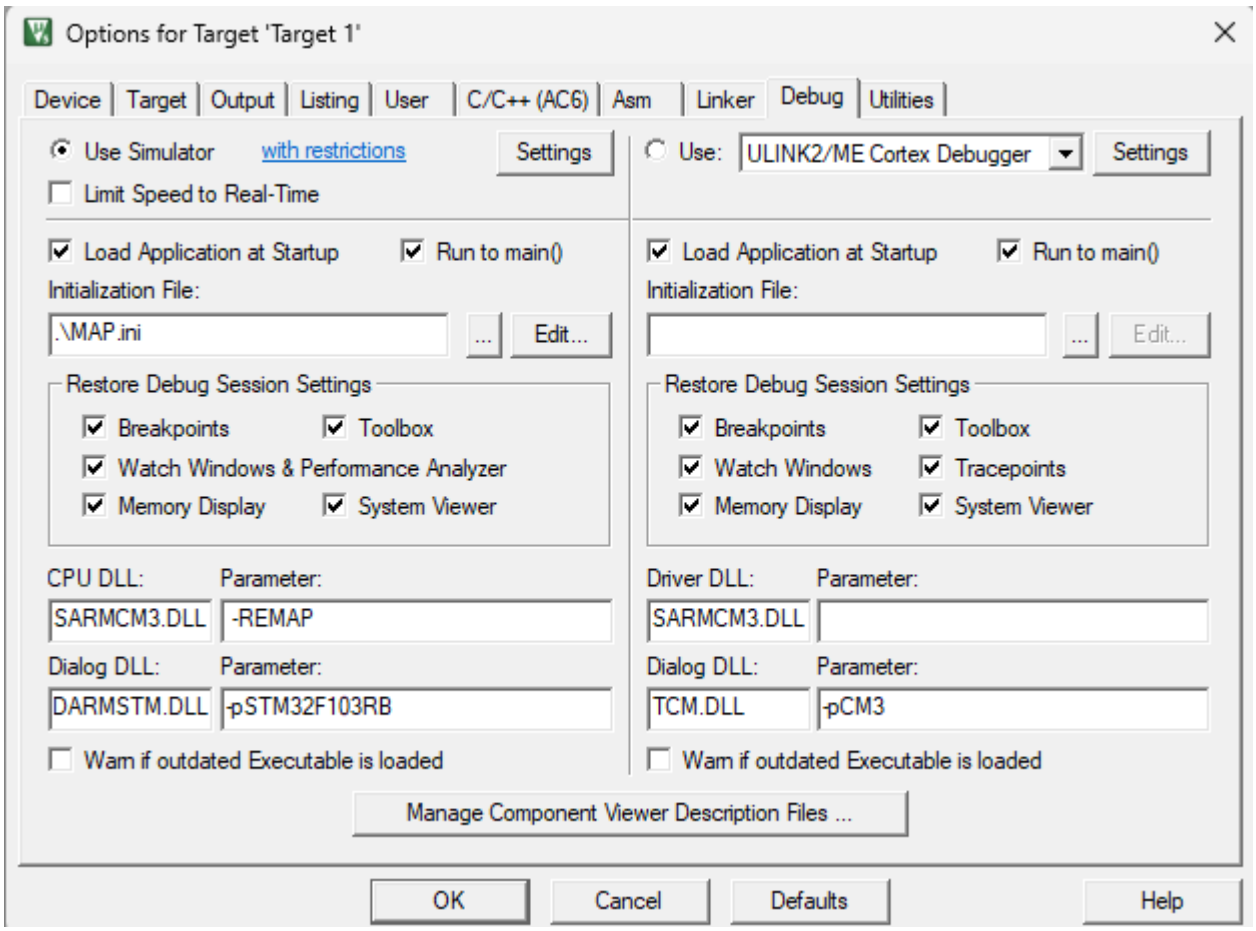

Рисунок 3 — Настройка проекта

3.5 Был разработан файл сценария. Код файла сценария представлен ниже:

```
signal void input(void)
{
    float volts, frequency, offset, val, steps;
    long i;
    volts = 2.0; // амплитуда изменения сигнала
    offset = 0.2; // минимальное значение напряжения
    frequency = 1400; // частота
    i = 0;
    steps = (100000 * (1 / frequency));
    while (1) {
        val = (i % steps) / ((float)steps);
        ADC1_IN8 = (val * volts) + offset;
        i++;
        swatch(0.00001); // изменение сигнала раз в 10 мкс
    }
}
```

3.6 Была разработана программа согласно задания. Код программы представлен в Приложении А.

3.7 Была выполнена симуляция разработанной программы с использованием функций отладки, а также зафиксированы результаты функционирования программы. Результаты представлены на рисунке 4.

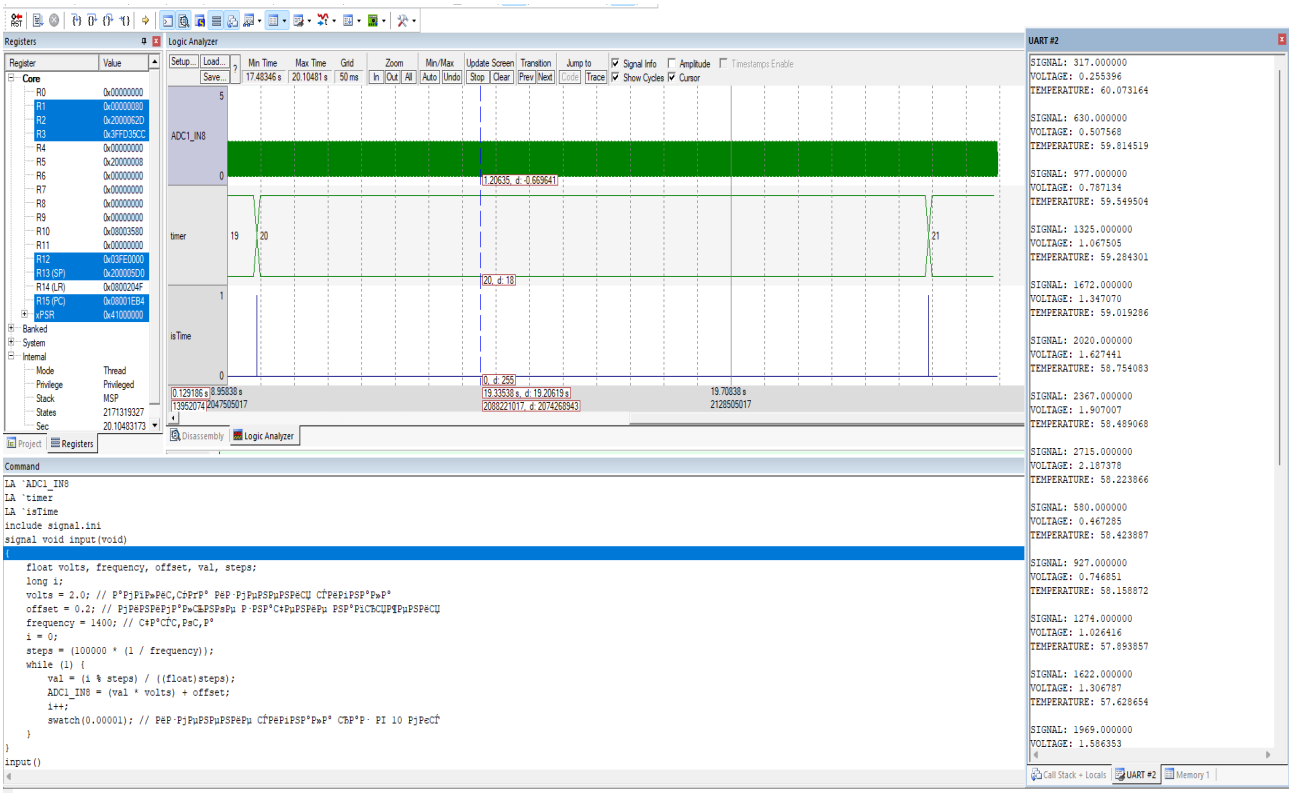

Рисунок 4 - Симуляция

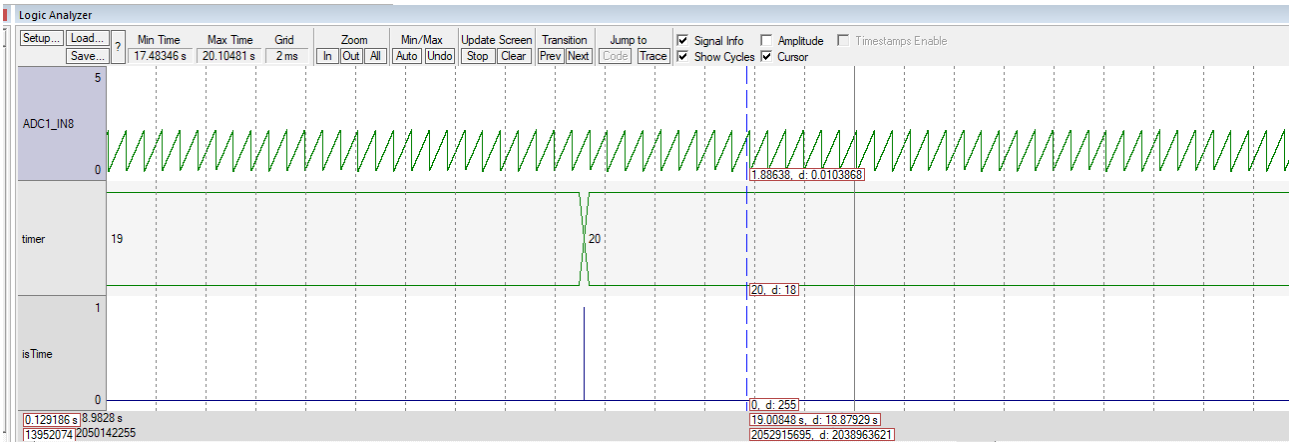

Рисунок 5 - Симуляция

4 Вывод: было выполнено ознакомление с программными средствами работы с контроллером аналогово-цифрового преобразования в микроконтроллерах STM32.

Приложение А

Код программы

```
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"

uint16_t timer = 0;
uint8_t isTime = 1;

static void USART_SendArray(const char *data)
{
    uint64_t i = 0;
    while (data[i])
    {
        USART_SendData(USART2, data[i]);
        while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
            ; // Ожидание завершения передачи
        i++;
    }
}

static uint32_t readADC()
{
    ADC_SoftwareStartConvCmd(ADC1, ENABLE); /* Запуск преобразования */
    while (ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET)
        ;                                   /* Ожидание завершения преобразования */
    return ADC_GetConversionValue(ADC1); /* Чтение и возврат результата преобразования
*/
}
static void initGPIO(void) // Функция инициализации GPIO
{
    GPIO_InitTypeDef PortObj;                              // Объявление структуры для
инициализации GPIO
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); // Включение тактирования
порта A


    /* Настройка порта PA2 */
    PortObj.GPIO_Pin = GPIO_Pin_2;         // Настройка пина 2
    PortObj.GPIO_Speed = GPIO_Speed_50MHz; // Установка скорости GPIO
    PortObj.GPIO_Mode = GPIO_Mode_AF_PP;   // Установка режима альтернативной функции
push-pull
    GPIO_Init(GPIOA, &PortObj);            // Инициализация GPIOA с использованием
структуры Obj


    /* Настройка порта PA3 */
    PortObj.GPIO_Pin = GPIO_Pin_3;             // Настройка пина 3
    PortObj.GPIO_Mode = GPIO_Mode_IN_FLOATING; // Установка режима входа с подтяжкой к
"плавающему" уровню
    GPIO_Init(GPIOA, &PortObj);                // Инициализация GPIOA с использованием
структуры Obj
}
static void initTIM3(void)
{
    TIM_TimeBaseInitTypeDef TIMER;


    /* Включение тактирования таймера TIM3 */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);


    /* Настройка периода и прескеллера TIM3 */
    TIMER.TIM_Period = 10000;
    TIMER.TIM_Prescaler = 10799;


    /* Настройка делителя частоты, режим счета таймера TIM3 */
    TIMER.TIM_ClockDivision = 0;                /* Делитель частоты таймера */
    TIMER.TIM_CounterMode = TIM_CounterMode_Up; /* Режим счета вверх */
    TIM_TimeBaseInit(TIM3, &TIMER);             /* Применение настроек */
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);  /* Включение прерывания по событию
захвата на канале 1 */


    TIM_Cmd(TIM3, ENABLE);     /* Включение таймера TIM3 */
    NVIC_EnableIRQ(TIM3_IRQn); /* Разрешить прерывания от таймера 3 */
}
```

```
void TIM3_IRQHandler(void)
{
    if (!TIM_GetITStatus(TIM3, TIM_IT_Update))
        return;

    TIM_ClearITPendingBit(TIM3, TIM_IT_Update); /* Сброс флага прерывания */
    timer += 1;
    isTime = 1; /* Запись значения готовности времени */
}
static void initUSART(void) // Функция инициализации USART
{
    USART_InitTypeDef UsartObj;                              // Объявление структуры для
инициализации USART
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE); // Включение тактирования
USART2

    UsartObj.USART_BaudRate = 2400;                                          // Установка
скорости передачи данных
    UsartObj.USART_WordLength = USART_WordLength_8b;                         // Установка
длины слова
    UsartObj.USART_StopBits = USART_StopBits_1;                              // Установка
количества стоп-битов
    UsartObj.USART_Parity = USART_Parity_No;                                 // Установка
контроля четности
    UsartObj.USART_HardwareFlowControl = USART_HardwareFlowControl_None; // Установка
аппаратного управления потоком
    UsartObj.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;                     // Установка
режима передачи и приема
    USART_Init(USART2, &UsartObj);                                           //
Инициализация USART2 с использованием структуры Obj

    USART_Cmd(USART2, ENABLE); // Включение USART2
}

static void initADC(void)
{
    ADC_InitTypeDef ADCObj;

    /* Включение тактирования ADC1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
```

```
    /* Сброс калибровки ADC1 */
    ADC_ResetCalibration(ADC1);
    while (ADC_GetResetCalibrationStatus(ADC1))
        ;

    /* Запуск калибровки ADC1 */
    ADC_StartCalibration(ADC1);
    while (ADC_GetCalibrationStatus(ADC1))
        ;

    /* Настройка ADC1 */
    ADCObj.ADC_Mode = ADC_Mode_Independent;
    ADCObj.ADC_ScanConvMode = DISABLE;
    ADCObj.ADC_ContinuousConvMode = ENABLE;
    ADCObj.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADCObj.ADC_DataAlign = ADC_DataAlign_Right;
    ADCObj.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADCObj);

    /* Включение внутреннего источника опорного напряжения */
    ADC_TempSensorVrefintCmd(ENABLE);

    /* Настройка ADC1 для канала 8 с временем выборки 55.5 циклов */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 1, ADC_SampleTime_55Cycles5);
    ADC_Cmd(ADC1, ENABLE);
}

int main(void)
{
    char xbuffer[64];
    double S, V, T;
    __enable_irq();
    initGPIO();
    initTIM3();
    initUSART();
    initADC();
```

```
    while (1)
    {
        if (!isTime)
            continue;
        isTime = 0;

        S = readADC();
        sprintf(xbuffer, "%f", S);
        USART_SendArray("SIGNAL: ");
        USART_SendArray(xbuffer);
        USART_SendArray("\n");

        /*
            2^12 = 4096 из формулы
                                        3.29999995 - Опорное напряжение (debug - console
- DIR VTREG)
        */

        V = (S * 3.29999995) / 4096;
        sprintf(xbuffer, "%f", V);
        USART_SendArray("VOLTAGE: ");
        USART_SendArray(xbuffer);
        USART_SendArray("\n");

        /*
                4.3 - среднее приращение температуры;
                1.43 - напряжение, соответствующее температуре +25°C
        */

        T = (1.43 - V) / 4.3 + 60 - (timer * 0.2); //
        sprintf(xbuffer, "%f", T);
        USART_SendArray("TEMPERATURE: ");
        USART_SendArray(xbuffer);
        USART_SendArray("\n\n");
    }
}
```