```
/* USER CODE BEGIN Header */
/**
SIGMA DELTA ADC project
by Szymon Filipkowski

sigma delta adc code main file

KOD JEST W TRAKCIE PRACY //TODO //TODO //TODO
**/
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define true 1
#define false 0
#define HIGH 1
#define LOW 0

//ADC settings
#define MAX_TICKS 255
#define MAGIC_VOLTAGE_MULTIPLIER 1 //high state / low state *
MAGIC_VOLTAGE_MULTIPLIER = voltage //TODO maybe proporcja NA PEWNO TO ADJUST /
cos innego wymyslic
#define STATIC_VOLTAGE_MULTIPLIER 1 //static, every time used multiplier to
multiply voltage by, beacuse of hardware issues
#define VOLTAGE_OFFSET 1  //static, every time used voltage offset to add,
beacuse of hardware issues
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

TIM_HandleTypeDef htim17;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
uint8_t VOLTAGE = 255; //real voltage = VOLTAGE / 100
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM17_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
uint8_t ANALOG_TO_DIGITAL(uint8_t is_it_first);
void SEND_VIA_UART(uint8_t toSend);
void MANUAL_MODE(void);
void EXIT_DEEP_SLEEP_MODE(void);
void SEND_VIA_UART(void);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM17_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim17);
  __HAL_RCC_PWR_CLK_ENABLE(); //TODO TO CHECK IF WORKS STOP MODE //
https://www.youtube.com/watch?v=td_CbkFBCfE
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  __HAL_FLASH_SET_LATENCY(FLASH_LATENCY_1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSIDiv = RCC_HSI_DIV1;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM17 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM17_Init(void)
{

  /* USER CODE BEGIN TIM17_Init 0 */

  /* USER CODE END TIM17_Init 0 */

  /* USER CODE BEGIN TIM17_Init 1 */

  /* USER CODE END TIM17_Init 1 */
  htim17.Instance = TIM17;
  htim17.Init.Prescaler = 46875-1;
  htim17.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim17.Init.Period = 2-1;
  htim17.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim17.Init.RepetitionCounter = 0;
  htim17.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim17) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM17_Init 2 */

  /* USER CODE END TIM17_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.Init.ClockPrescaler = UART_PRESCALER_DIV1;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin : MODE_SELECT_Pin */
  GPIO_InitStruct.Pin = MODE_SELECT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(MODE_SELECT_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : DIGITAL_INPUT_Pin */
  GPIO_InitStruct.Pin = DIGITAL_INPUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DIGITAL_INPUT_GPIO_Port, &GPIO_InitStruct);
  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI0_1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI0_1_IRQn);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
//==============================================================================
================================================================================
======
//==============================================================================
================================================================================
======
//==============================================================================
================================================================================
======
/*      //TODO things
 * try to do digital read on EXTI???
deep sleep/light sleep cpu?
err show in console via uart?

//OTHER INFO

*/
uint8_t ANALOG_TO_DIGITAL(uint8_t is_it_first) //conversion from sigma delta
hardware output to digital data
{
        static uint8_t was_high = false; //have i already been at the top of the
func?
        static uint8_t data = 0; //current state of function
        static uint16_t ticks_high=0; //how many tick was i on high state
        static uint16_t ticks_low=0; //how many tick was i on low state
        static uint16_t ticks=0;

        if(is_it_first == true)
        {
                was_high = false; //have i already been at the top of the func?
                data = 0; //current state of function
                ticks_high=0; //how many tick was i on high state
                ticks_low=0; //how many tick was i on low state
                ticks=0;
        }

        data = HAL_GPIO_ReadPin(DIGITAL_INPUT_GPIO_Port, DIGITAL_INPUT_Pin);

        ticks = ticks + 1;

        if((was_high == true && data == HIGH) || ticks >= MAX_TICKS) //back on
high
        {
                if(ticks_high >= MAX_TICKS) VOLTAGE = 255;
                else if(ticks_low >= MAX_TICKS) VOLTAGE = 0;
                else VOLTAGE = ticks_high / ticks_low *
MAGIC_VOLTAGE_MULTIPLIER;

                VOLTAGE = VOLTAGE * STATIC_VOLTAGE_MULTIPLIER + VOLTAGE_OFFSET;
//final voltage calculation

                //for new run
                ticks = 1;
                ticks_high = 1;
                ticks_low = 0;
                ticks = 1;
                was_high=false;
                return 1; //voltage analysis done
        }
        else if(data == HIGH) //first high
        {
                ticks_high = ticks_high + 1;
        }
        else //func went down
        {
                ticks_low = ticks_low + 1;
                was_high = true;
        }

        return 0; //nothing
}

void SEND_VIA_UART() //TODO //TODO //TODO
{

}


void MANUAL_MODE()
{
        HAL_TIM_Base_Stop_IT(&htim17); //stops auto mode

        ANALOG_TO_DIGITAL(true); //resets local vars from, auto mode
        while(ANALOG_TO_DIGITAL(false) == 0); //do until done, one full check

        //STOP MODE of mcu
        HAL_SuspendTick();
        HAL_PWR_EnterSTOPMode(PWR_MAINREGULATOR_ON, PWR_STOPENTRY_WFI); //TODO
low power regulator?
}

void EXIT_DEEP_SLEEP_MODE()
{
        HAL_ResumeTick();
        SystemClock_Config();
        HAL_TIM_Base_Start_IT(&htim17); //resums auto mode
}
```

```
//========================================================INTERRUPTS=============
=======================================================



void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin) //MODE pin was shorted
with 3.3 => we are into manual mode
{
    if(GPIO_Pin == MODE_SELECT_Pin)
    {
        MANUAL_MODE();
    }
}

void HAL_GPIO_EXTI_Falling_Callback(uint16_t GPIO_Pin) //MODE pin got release =>
we are going into auto mode
{
    if(GPIO_Pin == MODE_SELECT_Pin)
    {
        EXIT_DEEP_SLEEP_MODE();
    }
}


void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) //time based
interrupts
{
          if(htim->Instance == TIM17) //execute every something of time
          {
                  ANALOG_TO_DIGITAL(false);
          }
}

//===============================================================================
=================================================================================
======
//===============================================================================
=================================================================================
======
//===============================================================================
=================================================================================
======
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
//TODO user output of error in uart?

  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line 
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```