```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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"
#include "i2c.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
typedef struct HTS221_RegistersContent{
      uint8_t whoAmI;
      uint8_t avConf;
      uint8_t ctrlReg1;
      uint8_t ctrlReg2;
      uint8_t ctrlReg3;
      uint8_t statusReg;
      uint8_t humidityOut[2];
    uint8_t tempOut[2];
    uint8_t calib[16];
}HTS221_RegistersContentTypeDef;
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define HTS221_ADDRESS 0x05F //101 1111
#define HTS221_WHO_AM_I 0x0F //Default: 0xBC
#define HTS221_AV_CONF 0x10 //Humidity and temperature resolution mode (Default:
?? 011 011)
#define HTS221_CTRL_REG1 0x20  //Default: 0 ???? 0 00
#define HTS221_CTRL_REG2 0x21 //Default: 0 ????? 0 0
#define HTS221_CTRL_REG3 0x22  //Default: 0 0 ??? 0 ??
#define HTS221_STATUS_REG 0x27 //Default: ?????? H_DA T_DA
#define HTS221_HUMIDITY_OUT_L 0x28
#define HTS221_HUMIDITY_OUT_H 0x29
#define HTS221_TEMP_OUT_L 0x2A
#define HTS221_TEMP_OUT_H 0x2B
#define HTS221_CALIB_0 0x30
```

```c
/* USER CODE END PD */

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

/* USER CODE END PM */

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

/* USER CODE BEGIN PV */
uint8_t userButtonFlag = 0;
HTS221_RegistersContentTypeDef humSensor;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
int __io_putchar(int ch){
  HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
  return ch;
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin == USER_BUTTON_Pin) userButtonFlag = 1;
}
/* 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_I2C1_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_WHO_AM_I,
I2C_MEMADD_SIZE_8BIT, &humSensor.whoAmI, 1, HAL_MAX_DELAY);
  printf("WHO AM I: %hx\r\n", humSensor.whoAmI);

  HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_AV_CONF,
I2C_MEMADD_SIZE_8BIT, &humSensor.avConf, 1, HAL_MAX_DELAY);
  printf("AV CONF: %hx\r\n", humSensor.avConf);

  //ODCZYT REJESTROW KONTROLNYCH
  HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_CTRL_REG1 | 0x80,
I2C_MEMADD_SIZE_8BIT, &humSensor.ctrlReg1, 3, HAL_MAX_DELAY);
  printf("CTRL REG1: %hx\r\nCTRL REG2: %hx\r\nCTRL REG1: %hx\r\n",
humSensor.ctrlReg1, humSensor.ctrlReg2, humSensor.ctrlReg3);

  //URUCHOMIENIE CZUJNIKA, URUCHMIENIE KONIECZNOSCI ODCZYTU STARSZEGO I MLODSZEGO
WYNIKU, TRYB ONE-SHOT
  humSensor.ctrlReg1 |= 0x84;
  printf("WRITE TO CTRL REG1: %hx\r\n", humSensor.ctrlReg1);
  HAL_I2C_Mem_Write(&hi2c1, HTS221_ADDRESS<<1, HTS221_CTRL_REG1,
I2C_MEMADD_SIZE_8BIT, &humSensor.ctrlReg1, 1, HAL_MAX_DELAY);

  HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_CTRL_REG1 | 0x80,
I2C_MEMADD_SIZE_8BIT, &humSensor.ctrlReg1, 3, HAL_MAX_DELAY);
  printf("CTRL REG1: %hx\r\nCTRL REG2: %hx\r\nCTRL REG1: %hx\r\n",
humSensor.ctrlReg1, humSensor.ctrlReg2, humSensor.ctrlReg3);

  HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_CALIB_0 | 0x80,
I2C_MEMADD_SIZE_8BIT, humSensor.calib, 16, HAL_MAX_DELAY);
  for(uint8_t i = 0; i < 16; i++){
        printf("CALIB_%hd: %hx\r\n", i, humSensor.calib[i]);
  }
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        if(userButtonFlag == 1){
            userButtonFlag = 0;
            HAL_GPIO_TogglePin(USER_LED_GPIO_Port, USER_LED_Pin);

            //WYKONANIE POMIARU
            humSensor.ctrlReg2 |= 0x01;
            printf("WRITE TO CTRL REG2: %hx\r\n", humSensor.ctrlReg2);
            HAL_I2C_Mem_Write(&hi2c1, HTS221_ADDRESS<<1, HTS221_CTRL_REG2,
I2C_MEMADD_SIZE_8BIT, &humSensor.ctrlReg2, 1, HAL_MAX_DELAY);

            do{
              HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_STATUS_REG,
I2C_MEMADD_SIZE_8BIT, &humSensor.statusReg, 1, HAL_MAX_DELAY);
              printf("STATUS REG: %hx\r\n", humSensor.statusReg);
            }while(humSensor.statusReg ^ 0x03);
```

```
            HAL_I2C_Mem_Read(&hi2c1, HTS221_ADDRESS<<1, HTS221_HUMIDITY_OUT_L |
0x80, I2C_MEMADD_SIZE_8BIT, humSensor.humidityOut, 4, HAL_MAX_DELAY);
            printf("HUMIDITY OUT L: %hx  HUMIDITY OUT H: %hx\r\nTEMP OUT L: %hx
TEMP OUT H: %hx\r\n",\
            humSensor.humidityOut[0], humSensor.humidityOut[1],
humSensor.tempOut[0], humSensor.tempOut[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};

  /** 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.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  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_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* 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 */
  /* 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 */
```