Введение в микроконтроллеры

Домашнее задание №3

Евгений Зеленин

18 января 2025 г.

1. Постановка задачи

Условия задачи. *Включите на МК STM32 два интерфейса SPI и передайте данные с одного на другой. Подтверждением может быть скриншот отладчика.*

2. Схема устройства

На рисунке 1 показана схема назначения пинов и подключений.

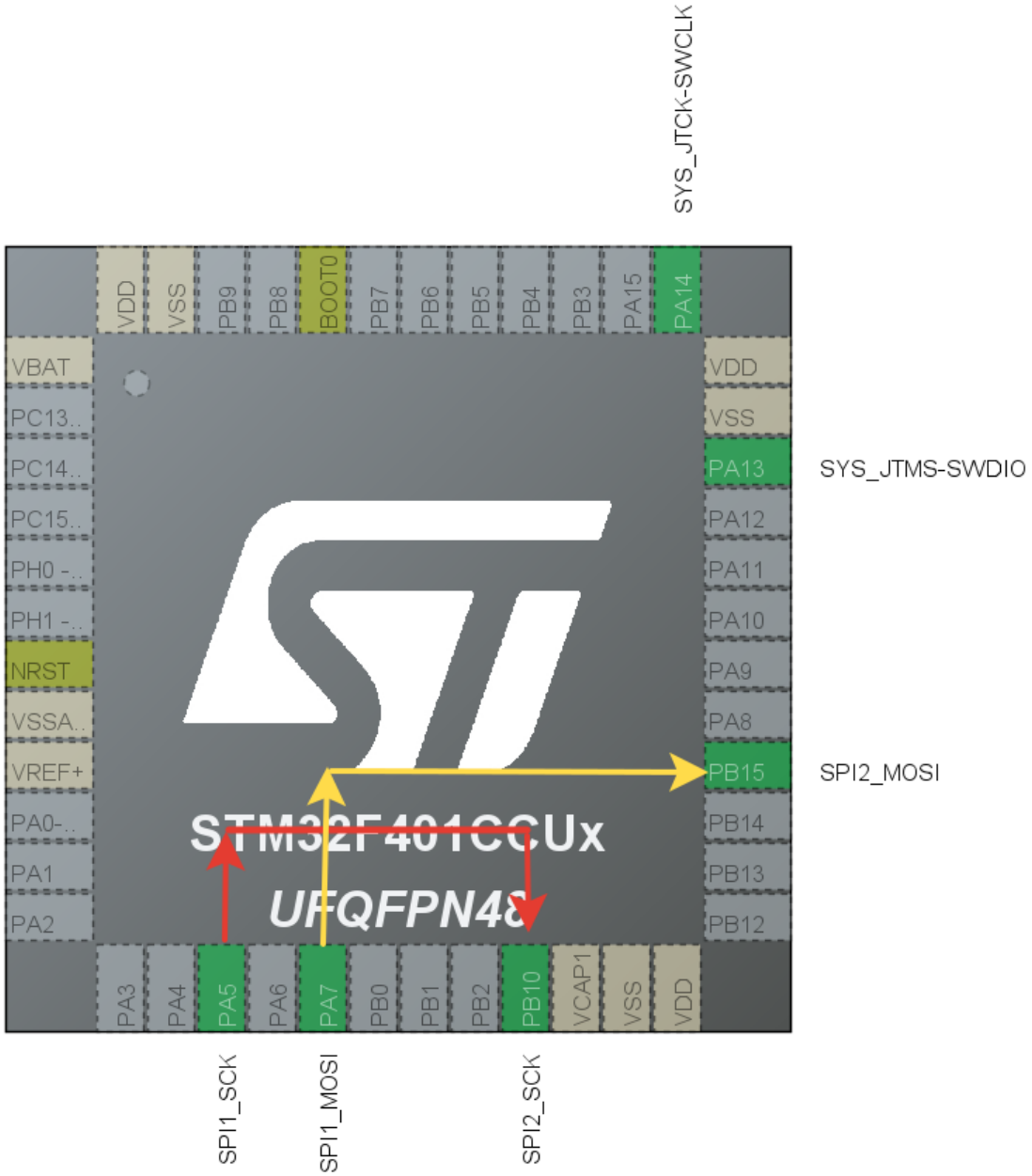

Рис. 1: Назначение портов

На рисунке 2 показано подключение на отладочной плате "в натуре".

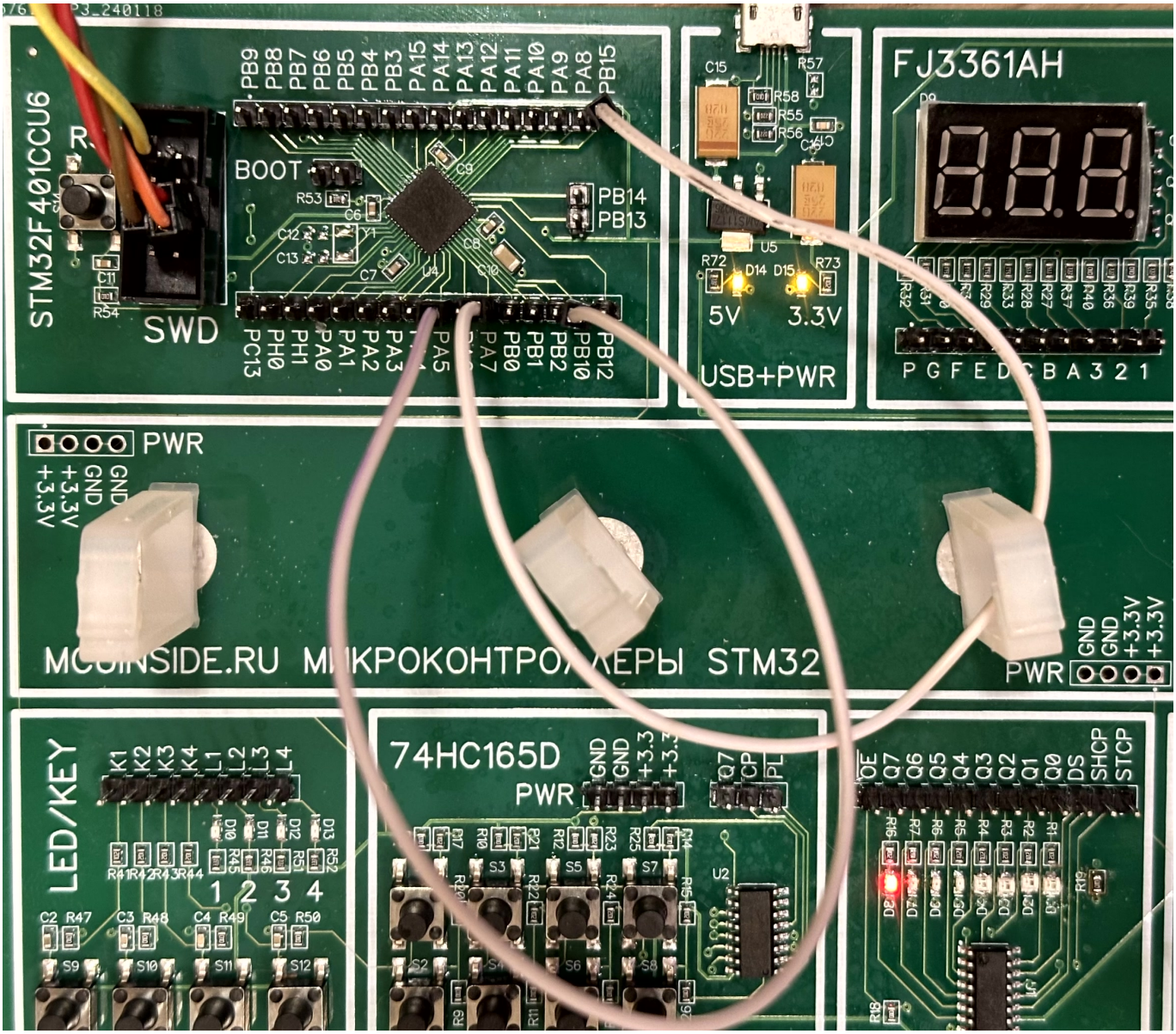

Рис. 2: Назначение портов

3. Описание работы устройства

Данные передаются с порта SPI_1 на порт SPI_2. Передача осуществляется в одну сторону, поэтому используется только порты MOSI и SCK. SPI_1 сконфигурирован как Transmit Only Master, а SPI_2 как Receive Only Slave. В начале, запустим порт SPI_2 на прослушивание неблокирующей функцией HAL_SPI_Receive_IT, после чего, отправим в него данные с помощью HAL_SPI_Transmit_IT. Для обработки состояний отправки и приема используются обработчики прерываний HAL_SPI_TxCpltCallback и HAL_SPI_ErrorCallback. По факту приема и передачи поднимаются флаги "sent"и "received".

4. Исходный код проекта (main.c) Содержимое файла main.c:

```
/* USER CODE BEGIN Header */
/**
```

```
 ******************************************************************************
 * @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 */

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi1;
SPI_HandleTypeDef hspi2;

/* USER CODE BEGIN PV */
uint8_t mreceive;
uint8_t msend;
uint8_t received = 1;
```

```
uint8_t sent = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_SPI2_Init(void);
/* USER CODE BEGIN PFP */

/* 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_SPI1_Init();
 MX_SPI2_Init();
 /* USER CODE BEGIN 2 */
```

```
 msend = 1;
 /* USER CODE END 2 */

 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1) {
  if(received){
  mreceive = 0;
  received = 0;
  sent = 0;
  HAL_SPI_Receive_IT(&hspi2, &mreceive, 1);
  HAL_SPI_Transmit_IT(&hspi1, &msend, 1);
  ++msend;
  }
  HAL_Delay(100);

  ///HAL_SPI_Abort_IT(&hspi1);
  /* 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 };

 /** Configure the main internal regulator output voltage
  */
 __HAL_RCC_PWR_CLK_ENABLE();
 __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);

 /** 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();
 }
}

/**
 * @brief SPI1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI1_Init(void) {

 /* USER CODE BEGIN SPI1_Init 0 */

 /* USER CODE END SPI1_Init 0 */

 /* USER CODE BEGIN SPI1_Init 1 */

 /* USER CODE END SPI1_Init 1 */
 /* SPI1 parameter configuration*/
 hspi1.Instance = SPI1;
 hspi1.Init.Mode = SPI_MODE_MASTER;
 hspi1.Init.Direction = SPI_DIRECTION_2LINES;
 hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
 hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
 hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
 hspi1.Init.NSS = SPI_NSS_SOFT;
 hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
 hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
 hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
 hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
 hspi1.Init.CRCPolynomial = 10;
 if (HAL_SPI_Init(&hspi1) != HAL_OK) {
  Error_Handler();
 }
 /* USER CODE BEGIN SPI1_Init 2 */
```

```
 /* USER CODE END SPI1_Init 2 */

}

/**
 * @brief SPI2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI2_Init(void) {

 /* USER CODE BEGIN SPI2_Init 0 */

 /* USER CODE END SPI2_Init 0 */

 /* USER CODE BEGIN SPI2_Init 1 */

 /* USER CODE END SPI2_Init 1 */
 /* SPI2 parameter configuration*/
 hspi2.Instance = SPI2;
 hspi2.Init.Mode = SPI_MODE_SLAVE;
 hspi2.Init.Direction = SPI_DIRECTION_2LINES_RXONLY;
 hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
 hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
 hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
 hspi2.Init.NSS = SPI_NSS_SOFT;
 hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
 hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
 hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
 hspi2.Init.CRCPolynomial = 10;
 if (HAL_SPI_Init(&hspi2) != HAL_OK) {
  Error_Handler();
 }
 /* USER CODE BEGIN SPI2_Init 2 */

 /* USER CODE END SPI2_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void) {
 /* USER CODE BEGIN MX_GPIO_Init_1 */
 /* USER CODE END MX_GPIO_Init_1 */
```

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

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

/* USER CODE BEGIN 4 */
void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi) {
 sent = 1;
}
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi) {
 received = 1;
}
void HAL_SPI_ErrorCallback(SPI_HandleTypeDef *hspi) {
 return;
}

/* 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 */
```

5. Демонстрация работы

На рисунке 3 показаны сообщения отладчика.

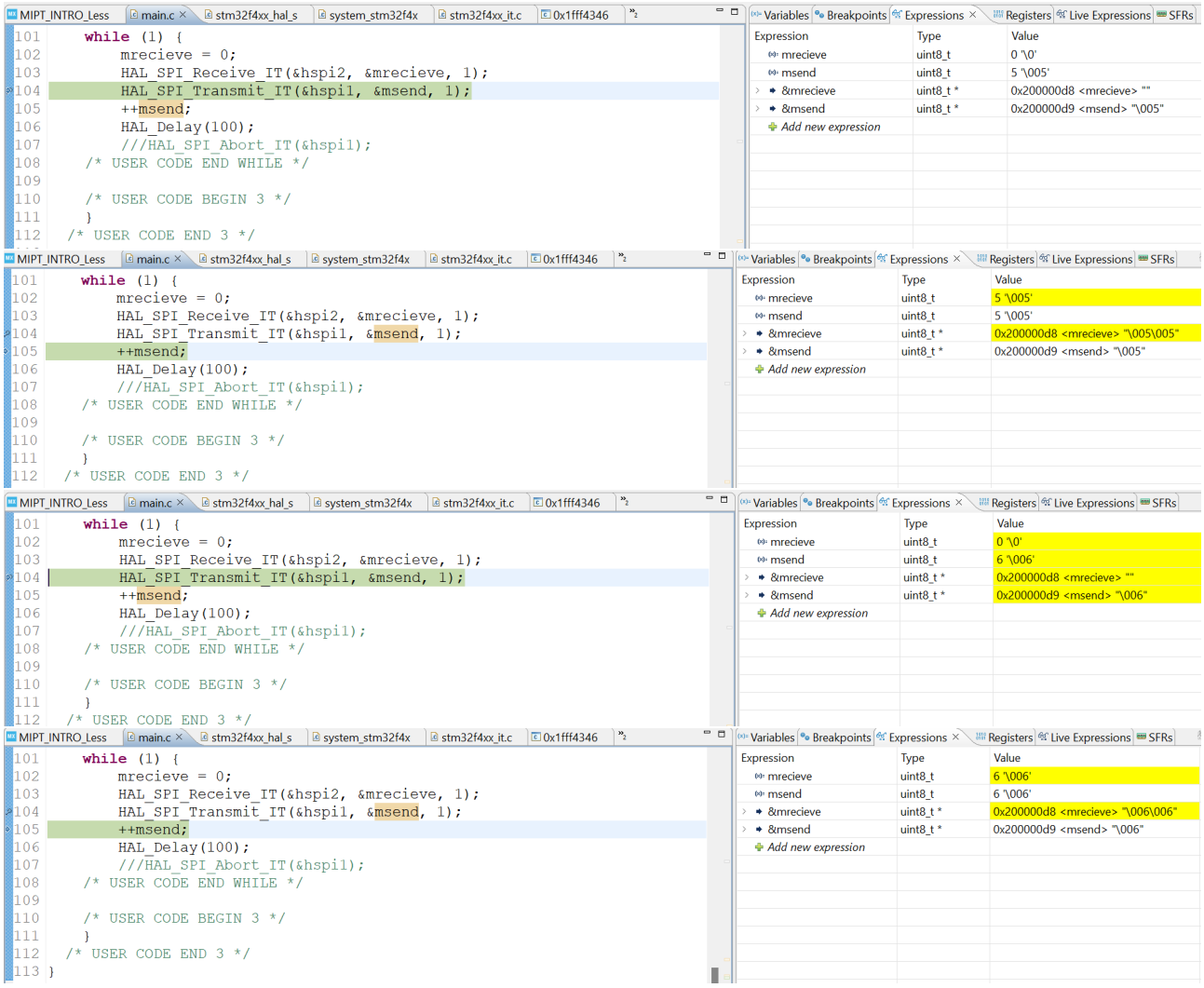

Рис. 3: Демонстрация работы

6. Дополнительные материалы

Видеозапись работы отладчика расположена в папке на google диск по следующей ссылке: Материалы к ДЗ №03