

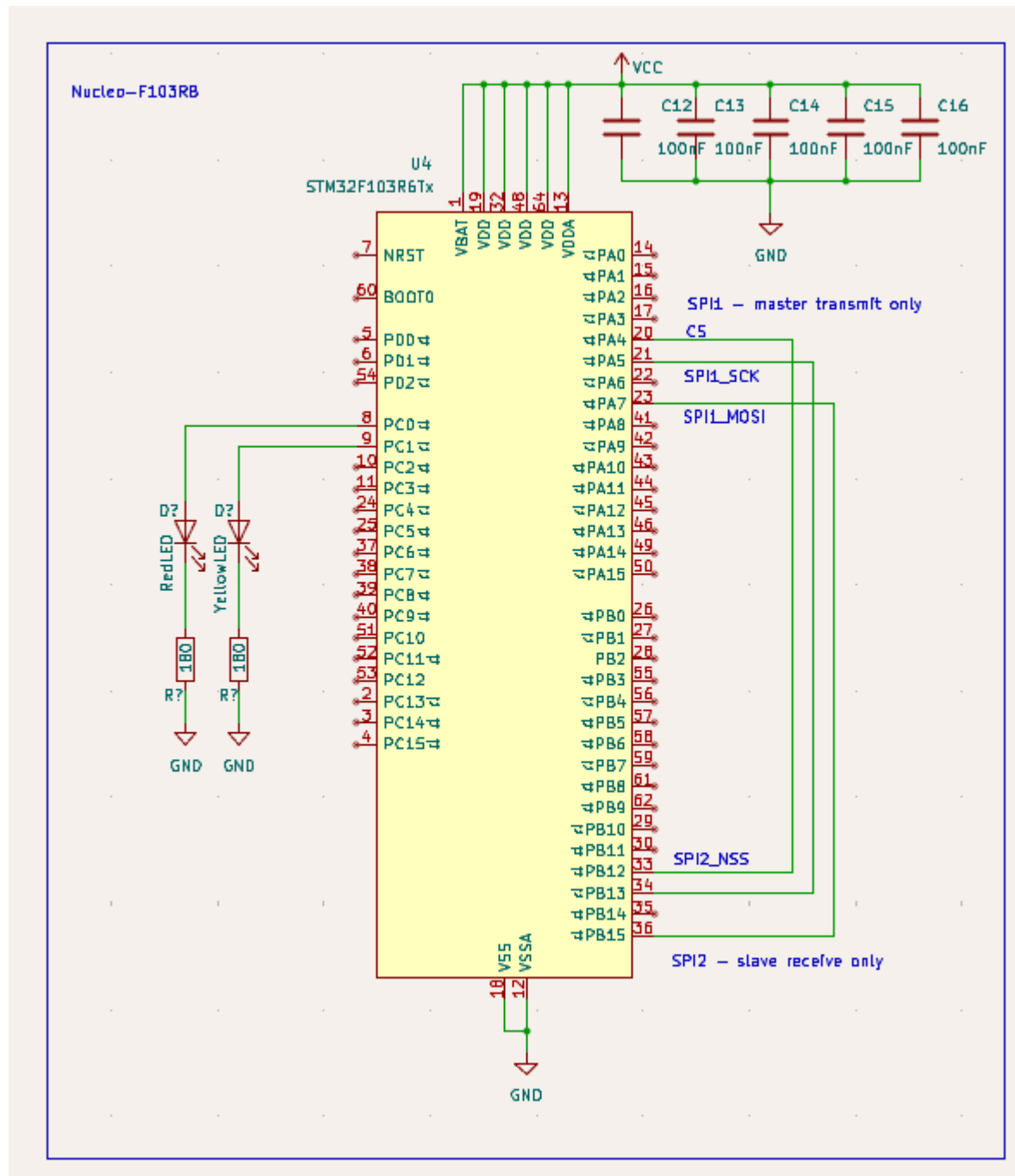
Задание 3

Описание задания

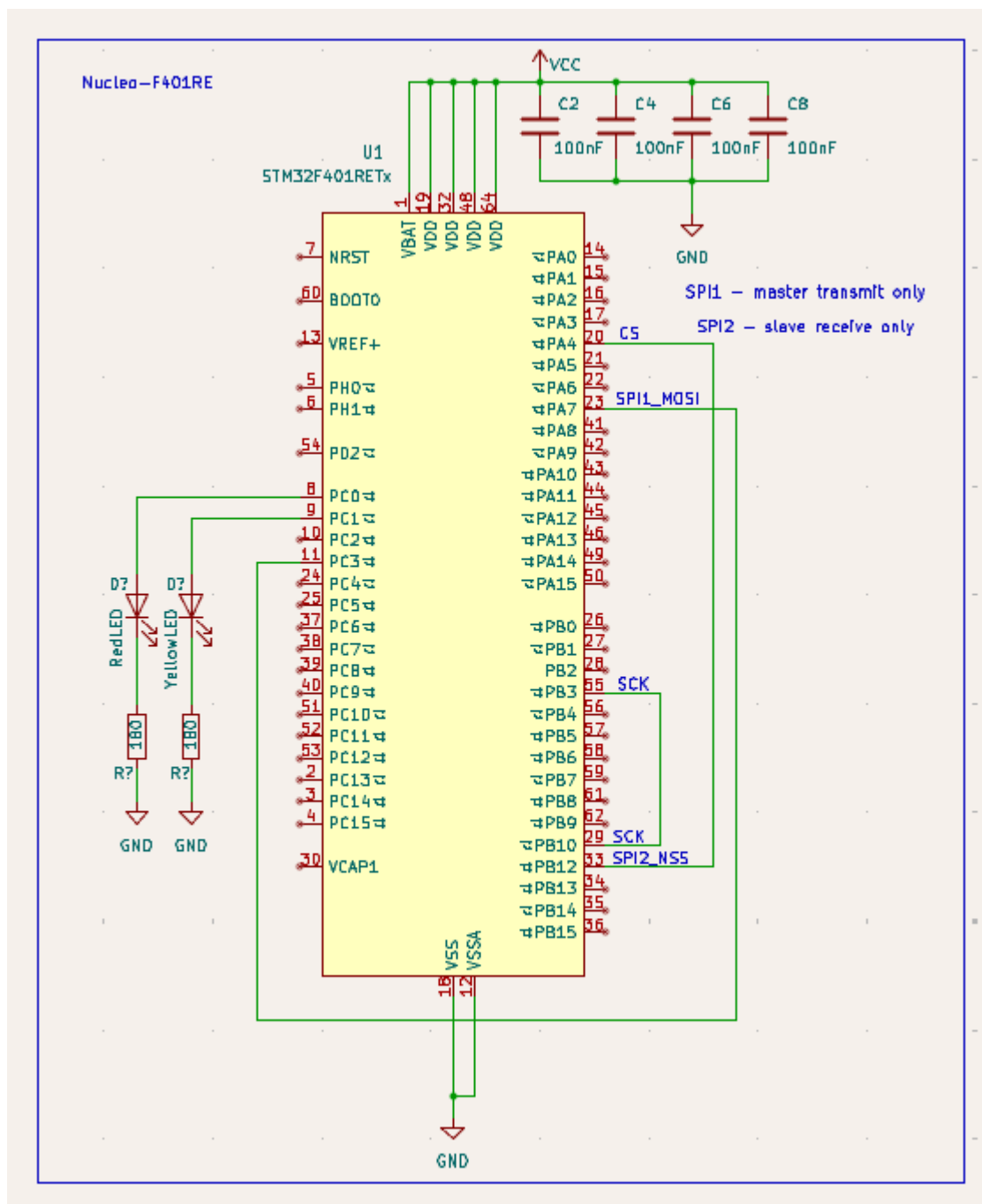
Включите на МК STM32 два интерфейса SPI и передайте данные с одного на другой. Подтверждением может быть скриншот отладчика.

Схемы включений

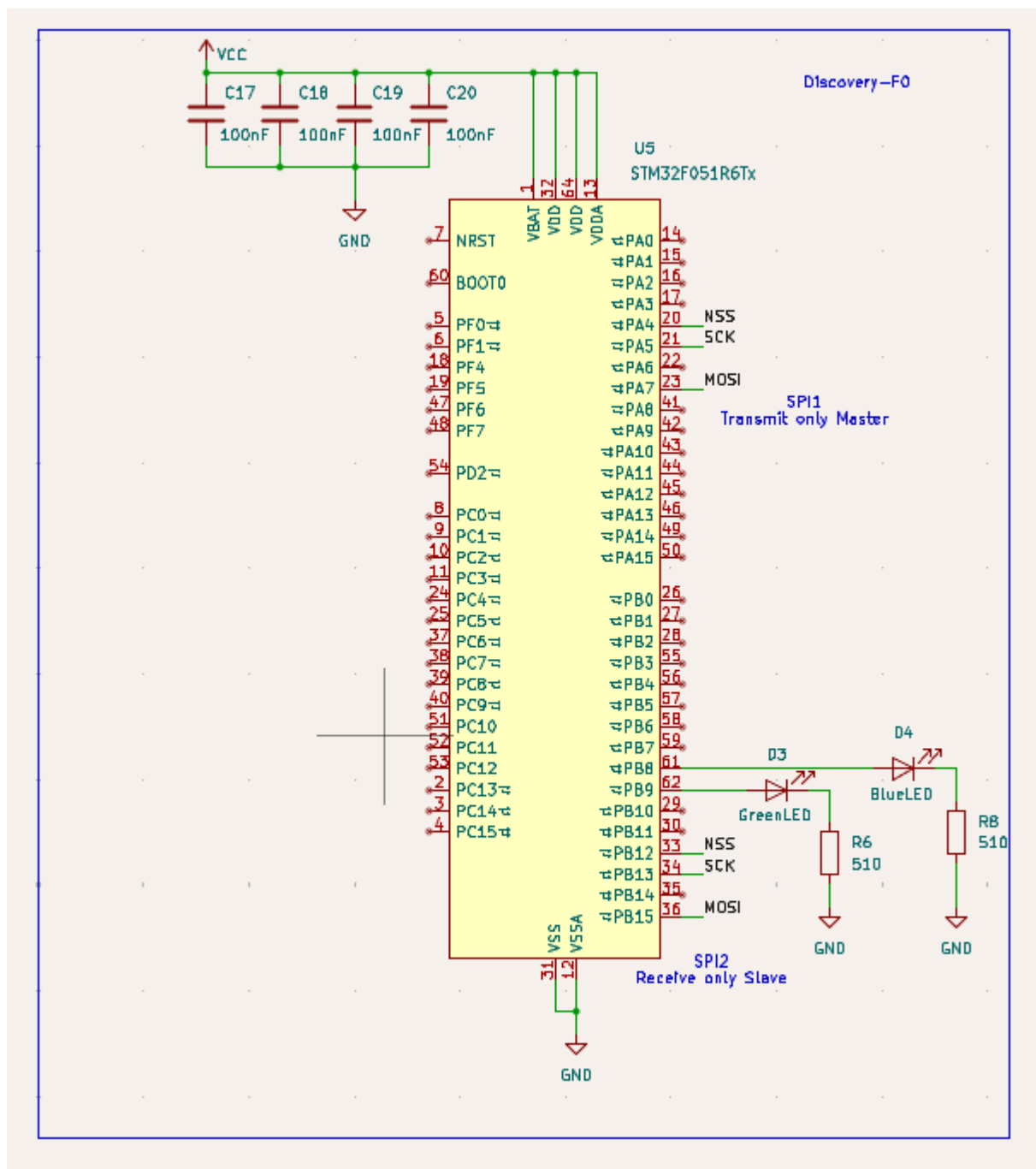
- на основе платы Nucleo-F103RB



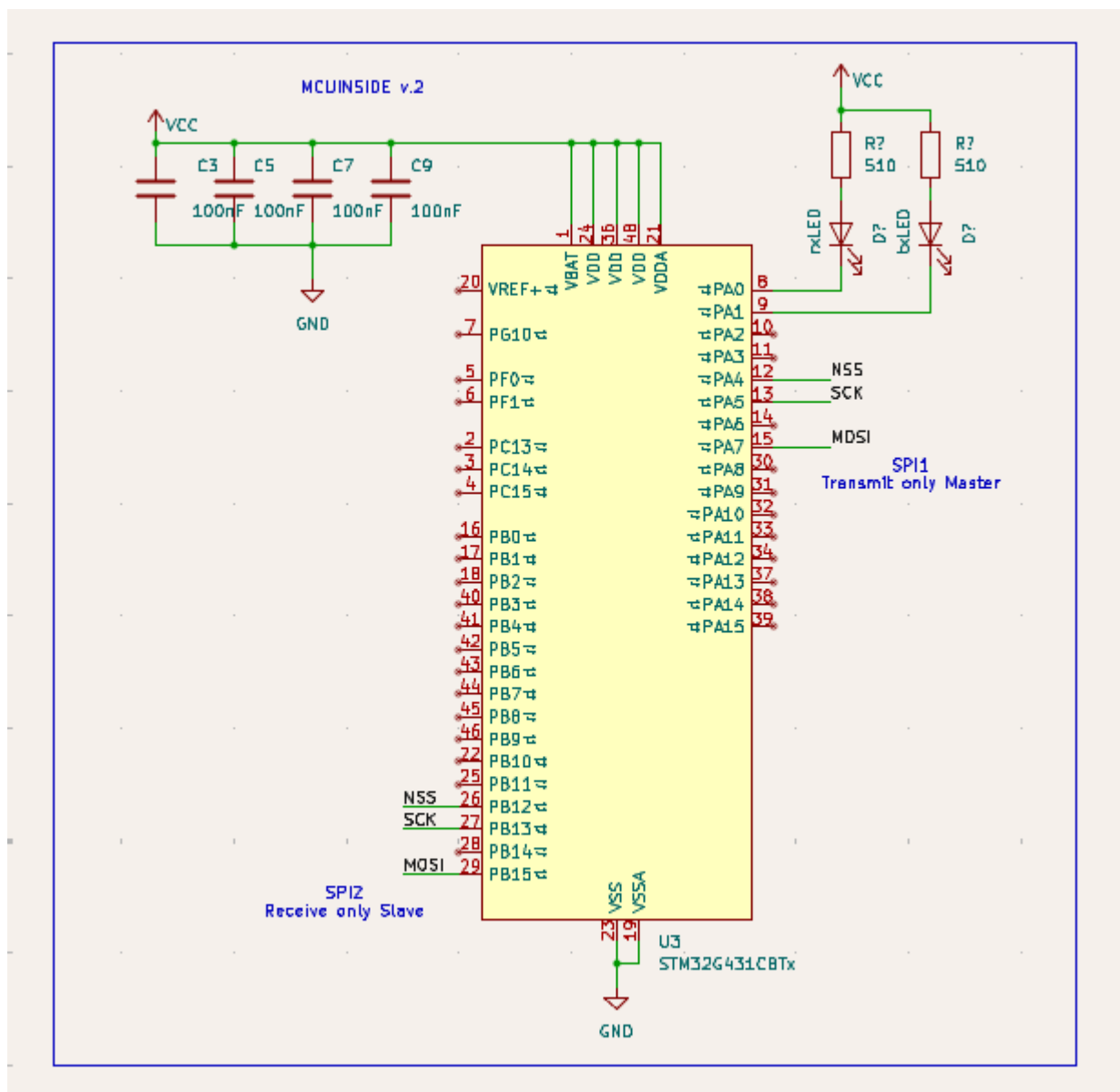
- на основе платы Nucleo-F401RE



- на основе платы Nucleo-F401RE



- на основе платы Nucleo-F401RE



Описание работы программы

1. На каждой плате интерфейс SPI1 сконфигурирован как мастер. Сигнал NSS реализован программно, для него выделен вывод порта GPIO.
2. На каждой плате интерфейс SPI2 сконфигурирован как ведомый. Для обработки приёма включено прерывание в контроллере NVIC.
3. Для приёма и передачи выделены приемные и передающие буферы. В передающий буфер записана строка с названием платы. Принимающий буфер имеет размер несколько больший, чем передающий.
4. Передача данных от мастера происходит периодически в основном цикле (ориентировочно 5 раз в секунду). По окончании передачи переключается один из светодиодов.
5. Приём осуществляется асинхронно, по прерыванию. По окончании приёма буфера в функции HAL_SPI_RxCpltCallback взводится флаг. Если флаг установлен, то в основном цикле программы происходит его сброс и переключение статуса другого светодиода.

Скриншоты отладчика

- Nucleo-F401RB

The screenshot shows the IDE interface for Nucleo-F401RB. The main.c file contains the following code:

```

102 /* Infinite loop */
103 /* USER CODE BEGIN WHILE */
104 while (1)
105 {
106     if (HAL_GetTick() - myTick > 100)
107     {
108         HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
109         myTick = HAL_GetTick();
110         HAL_SPI_Receive_IT(&hspi2, rxBuffer, sizeof(txBuffer));
111         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 0);
112         HAL_SPI_Transmit(&hspi1, txBuffer, sizeof(txBuffer), 1);
113         HAL_GPIO_TogglePin(RedLED_GPIO_Port, RedLED_Pin); // transmit complete
114         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 1);
115         HAL_Delay(200);
116     }
117     if (flag)
118     {
119         HAL_GPIO_TogglePin(YellowLED_GPIO_Port, YellowLED_Pin); //receive complete
120         flag = 0;
121     }
122 }
123 /* USER CODE END WHILE */
124
125 /* USER CODE BEGIN 3 */
126 }
127 /* USER CODE END 3 */
128
129 /**
130  * @brief System Clock Configuration
131  * @retval None
132  */
133 void SystemClock_Config(void)
134 {
135     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
136     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
137
138     /** Configure the main internal regulator output voltage
139     */
140     __HAL_RCC_PWR_CLK_ENABLE();
141     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
142
143     /** Initializes the RCC Oscillators according to the specified parameters
144     * in the RCC_OscInitTypeDef structure.
145     */
146     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
147     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
148 }

```

The Variables window on the right shows the following data:

Expression	Type	Value
txBuffer	uint8_t[12]	0x20017fd8
txBuffer[0]	uint8_t	78 'N'
txBuffer[1]	uint8_t	117 'u'
txBuffer[2]	uint8_t	99 'c'
txBuffer[3]	uint8_t	108 'l'
txBuffer[4]	uint8_t	101 'e'
txBuffer[5]	uint8_t	111 'o'
txBuffer[6]	uint8_t	45 '.'
txBuffer[7]	uint8_t	70 'F'
txBuffer[8]	uint8_t	52 '4'
txBuffer[9]	uint8_t	48 '0'
txBuffer[10]	uint8_t	49 '1'
txBuffer[11]	uint8_t	0 '\0'
rxBuffer	uint8_t[16]	0x20017fe4
rxBuffer[0]	uint8_t	78 'N'
rxBuffer[1]	uint8_t	117 'u'
rxBuffer[2]	uint8_t	99 'c'
rxBuffer[3]	uint8_t	108 'l'
rxBuffer[4]	uint8_t	101 'e'
rxBuffer[5]	uint8_t	111 'o'
rxBuffer[6]	uint8_t	45 '.'
rxBuffer[7]	uint8_t	70 'F'
rxBuffer[8]	uint8_t	52 '4'
rxBuffer[9]	uint8_t	48 '0'
rxBuffer[10]	uint8_t	49 '1'
rxBuffer[11]	uint8_t	0 '\0'
rxBuffer[12]	uint8_t	0 '\0'
rxBuffer[13]	uint8_t	0 '\0'
rxBuffer[14]	uint8_t	0 '\0'
rxBuffer[15]	uint8_t	0 '\0'

- Nucleo-F103RB

The screenshot shows the IDE interface for Nucleo-F103RB. The main.c file contains the following code:

```

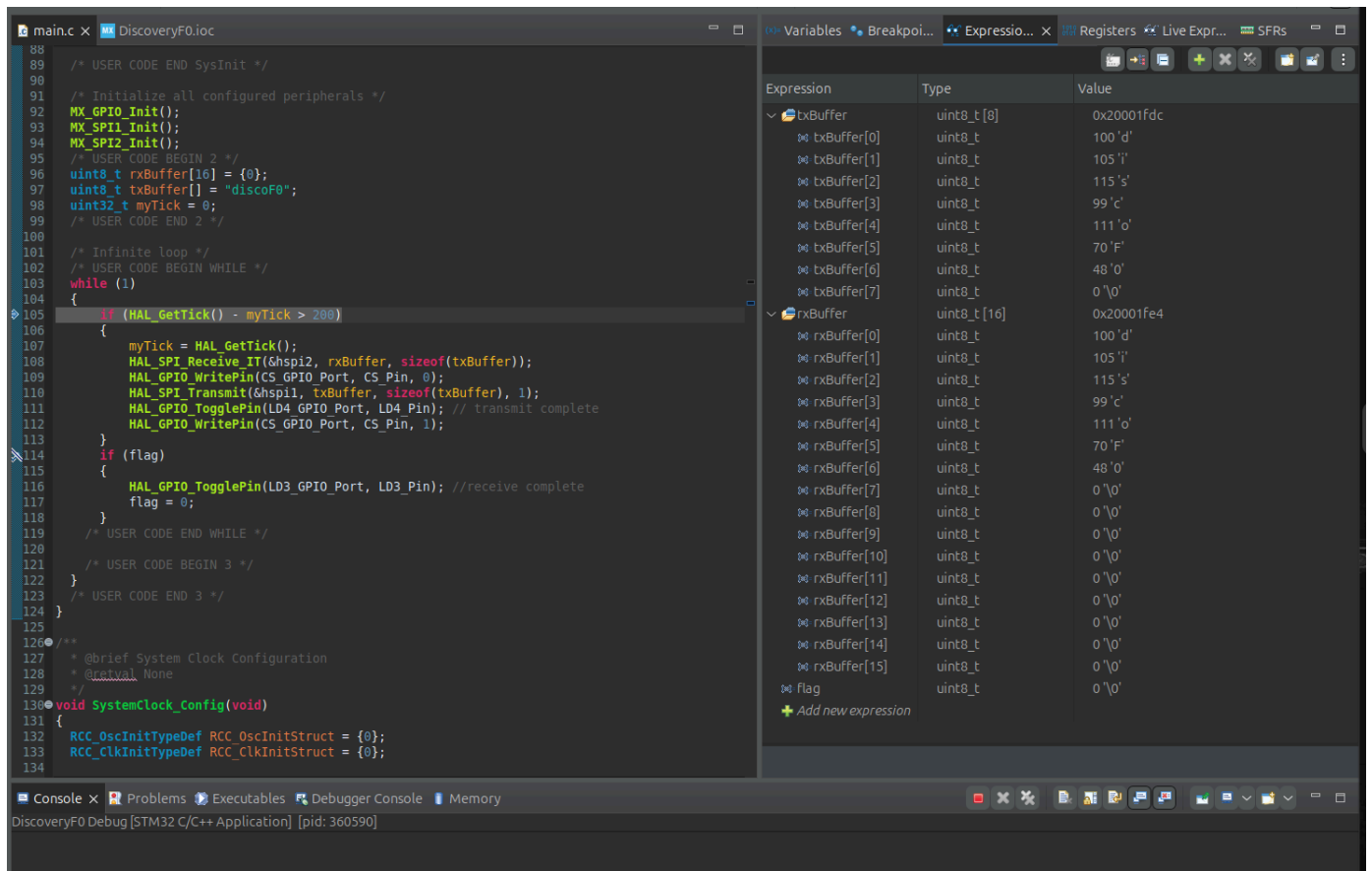
102 /* USER CODE BEGIN WHILE */
103 while (1)
104 {
105     if (HAL_GetTick() - myTick > 200)
106     {
107         myTick = HAL_GetTick();
108         HAL_SPI_Receive_IT(&hspi2, rxBuffer, sizeof(txBuffer));
109         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 0);
110         HAL_SPI_Transmit(&hspi1, txBuffer, sizeof(txBuffer), 1);
111         HAL_GPIO_TogglePin(RedLED_GPIO_Port, RedLED_Pin); // transmit complete
112         HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 1);
113     }
114     if (flag)
115     {
116         HAL_GPIO_TogglePin(YellowLED_GPIO_Port, YellowLED_Pin); //receive complete
117         flag = 0;
118     }
119 }
120 /* USER CODE END WHILE */
121
122 /* USER CODE BEGIN 3 */
123 }
124 /* USER CODE END 3 */
125
126 /**
127  * @brief System Clock Configuration
128  * @retval None
129  */
130 void SystemClock_Config(void)
131 {
132     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
133     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
134
135     /** Initializes the RCC Oscillators according to the specified parameters
136     * in the RCC_OscInitTypeDef structure.
137     */
138     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
139     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
140     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
141     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
142     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
143     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL4;
144     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
145     {
146         Error_Handler();
147     }
148 }

```

The Variables window on the right shows the following data:

Expression	Type	Value
txBuffer	uint8_t[12]	0x20004fd8
txBuffer[0]	uint8_t	78 'N'
txBuffer[1]	uint8_t	117 'u'
txBuffer[2]	uint8_t	99 'c'
txBuffer[3]	uint8_t	108 'l'
txBuffer[4]	uint8_t	101 'e'
txBuffer[5]	uint8_t	111 'o'
txBuffer[6]	uint8_t	45 '.'
txBuffer[7]	uint8_t	70 'F'
txBuffer[8]	uint8_t	49 '1'
txBuffer[9]	uint8_t	48 '0'
txBuffer[10]	uint8_t	51 '3'
txBuffer[11]	uint8_t	0 '\0'
rxBuffer	uint8_t[16]	0x20004fe4
rxBuffer[0]	uint8_t	78 'N'
rxBuffer[1]	uint8_t	117 'u'
rxBuffer[2]	uint8_t	99 'c'
rxBuffer[3]	uint8_t	108 'l'
rxBuffer[4]	uint8_t	101 'e'
rxBuffer[5]	uint8_t	111 'o'
rxBuffer[6]	uint8_t	45 '.'
rxBuffer[7]	uint8_t	70 'F'
rxBuffer[8]	uint8_t	49 '1'
rxBuffer[9]	uint8_t	48 '0'
rxBuffer[10]	uint8_t	51 '3'
rxBuffer[11]	uint8_t	0 '\0'
rxBuffer[12]	uint8_t	0 '\0'
rxBuffer[13]	uint8_t	0 '\0'
rxBuffer[14]	uint8_t	0 '\0'
rxBuffer[15]	uint8_t	0 '\0'

- Discovery F0



• MCUINSIDE v.2

