Electrical and Computer Engineering Department
ECE 4510 Microcontroller Applications

Lab 10

Interfacing a SRAM Chip to the Flexible Memory Controller (FMC)

Isaac Bagley and Rohullah Sah

March 23, 2023

Introduction

The goal of this lab was to learn the basics on how to use a Flexible Memory Controller(FMC) to interface with an SRAM chip and send data to it, as well as read the contents of the memory at specific locations. The lab will help us to learn how to interface with external memories through a microcontroller, an extremely useful skill in system design.

Procedure

Task 1

In this task, we were to design a hardware interface to an SRAM chip using a customized version of the FMC using the specifications given. We were to develop a C program which would write a single byte to a specific address location and then read it back ten times. Each time the data was read, it was compared with the data written and either a green LED light would turn on if the read data matched the write data, or a red LED light would turn on if the data did not match.

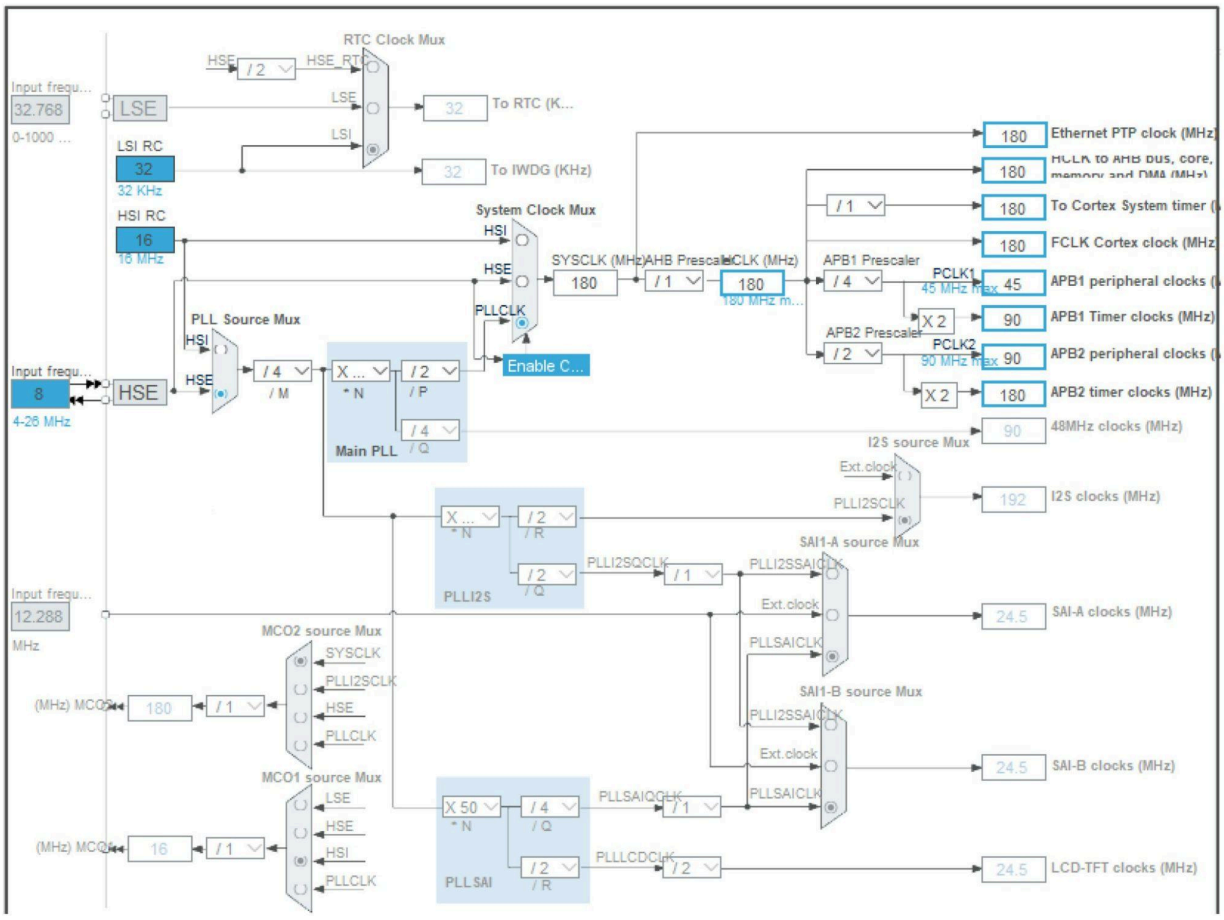

Figure1: Clock Configuration for FMC

The HCLK and AHB clock rate were set up to 180MHz using the HSE clock at 8MHz just like mentioned in the prelab.

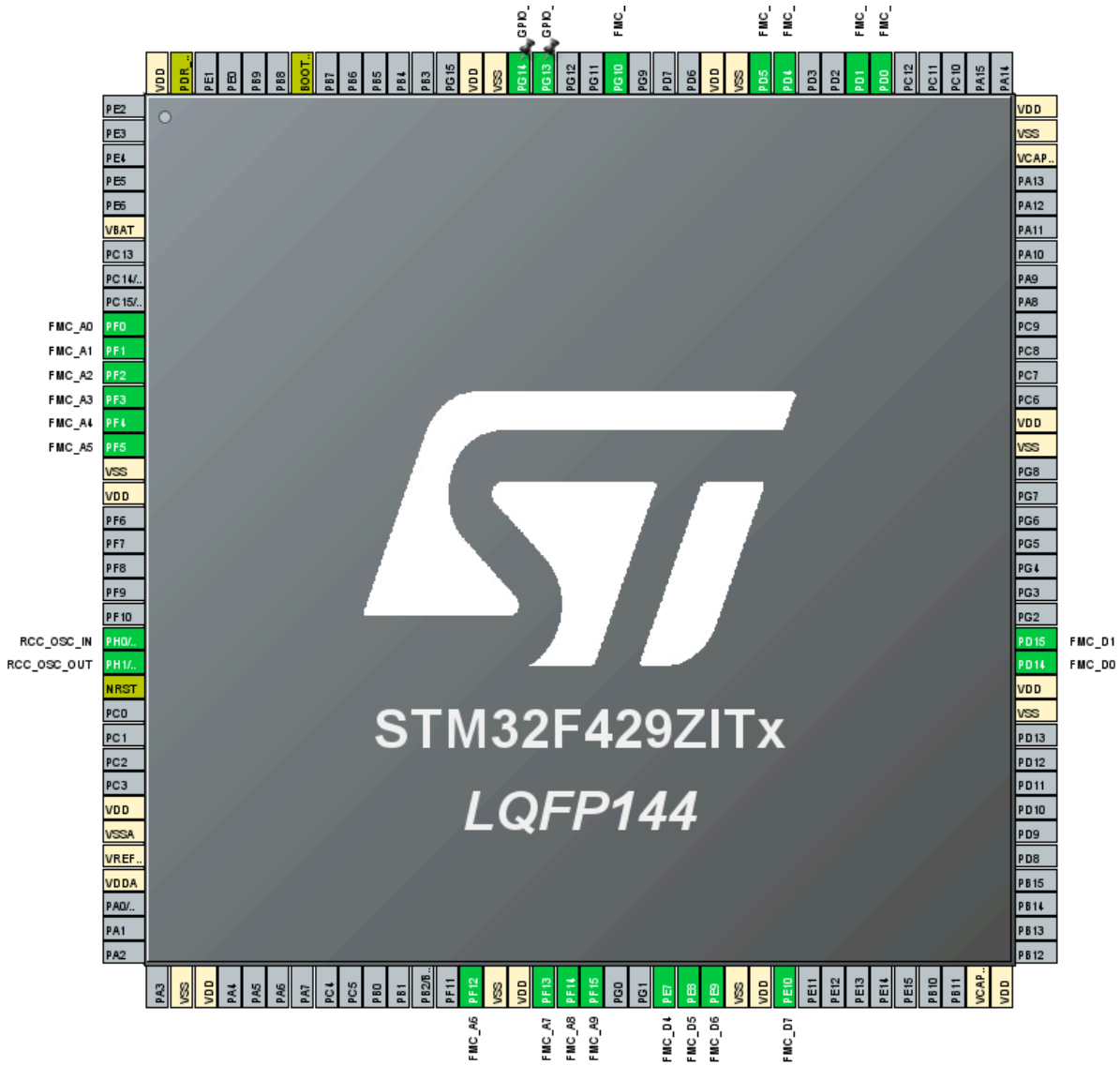

Figure 2: STM Cube configuration

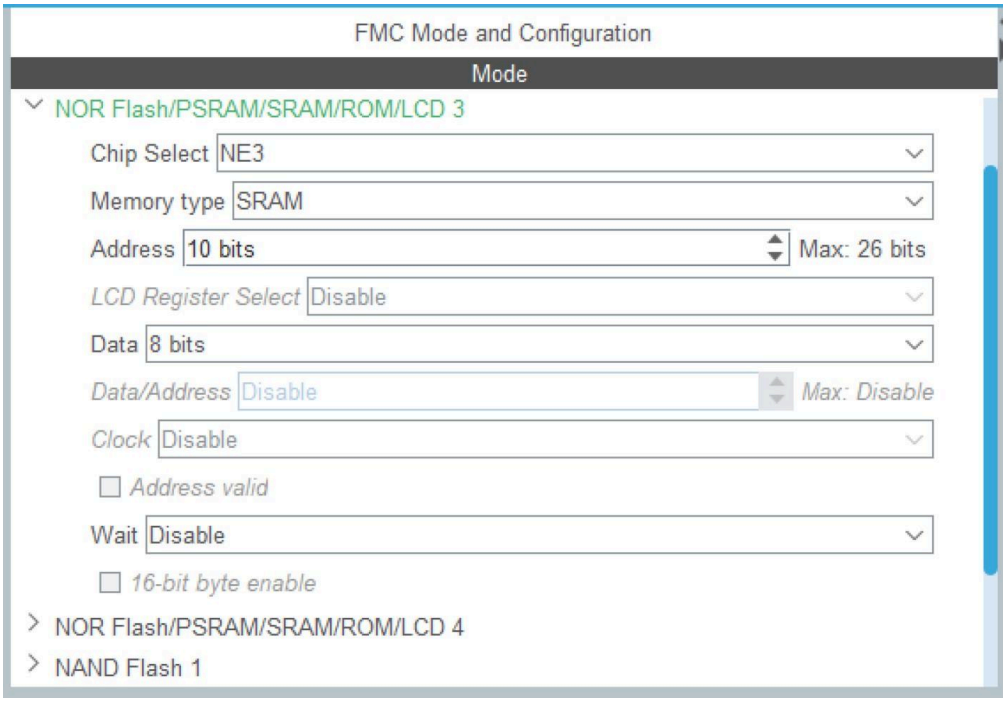

Figure 3: FMC Mode for SRAM3

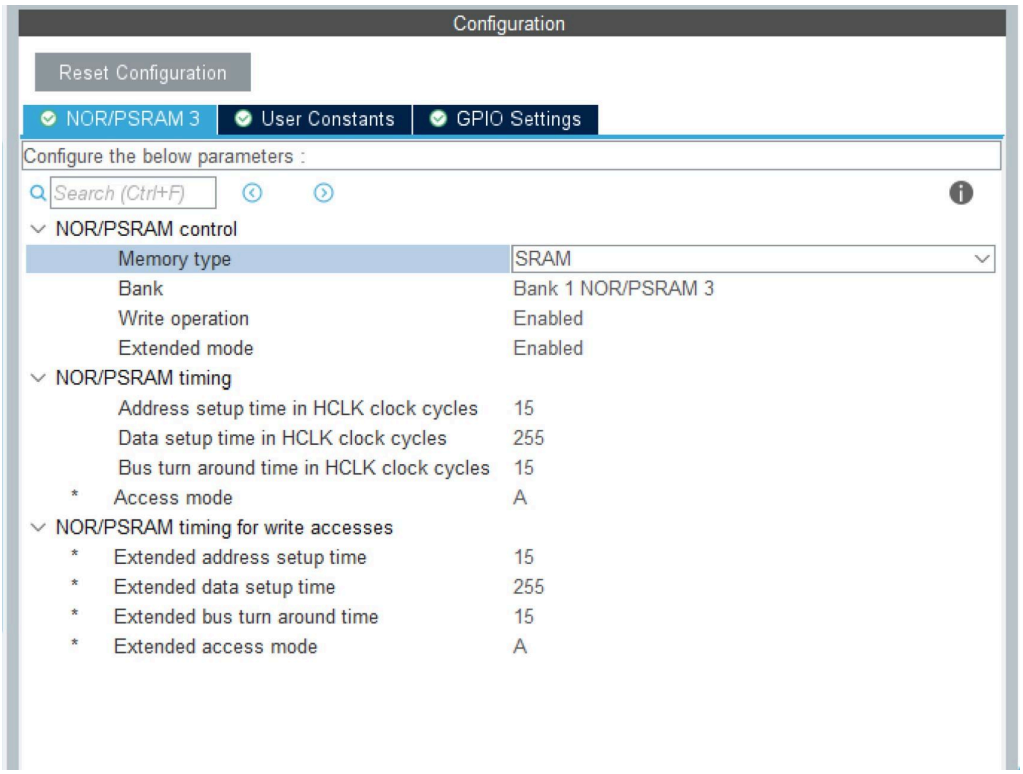

Figure 4: FMC Configuration for SRAM3

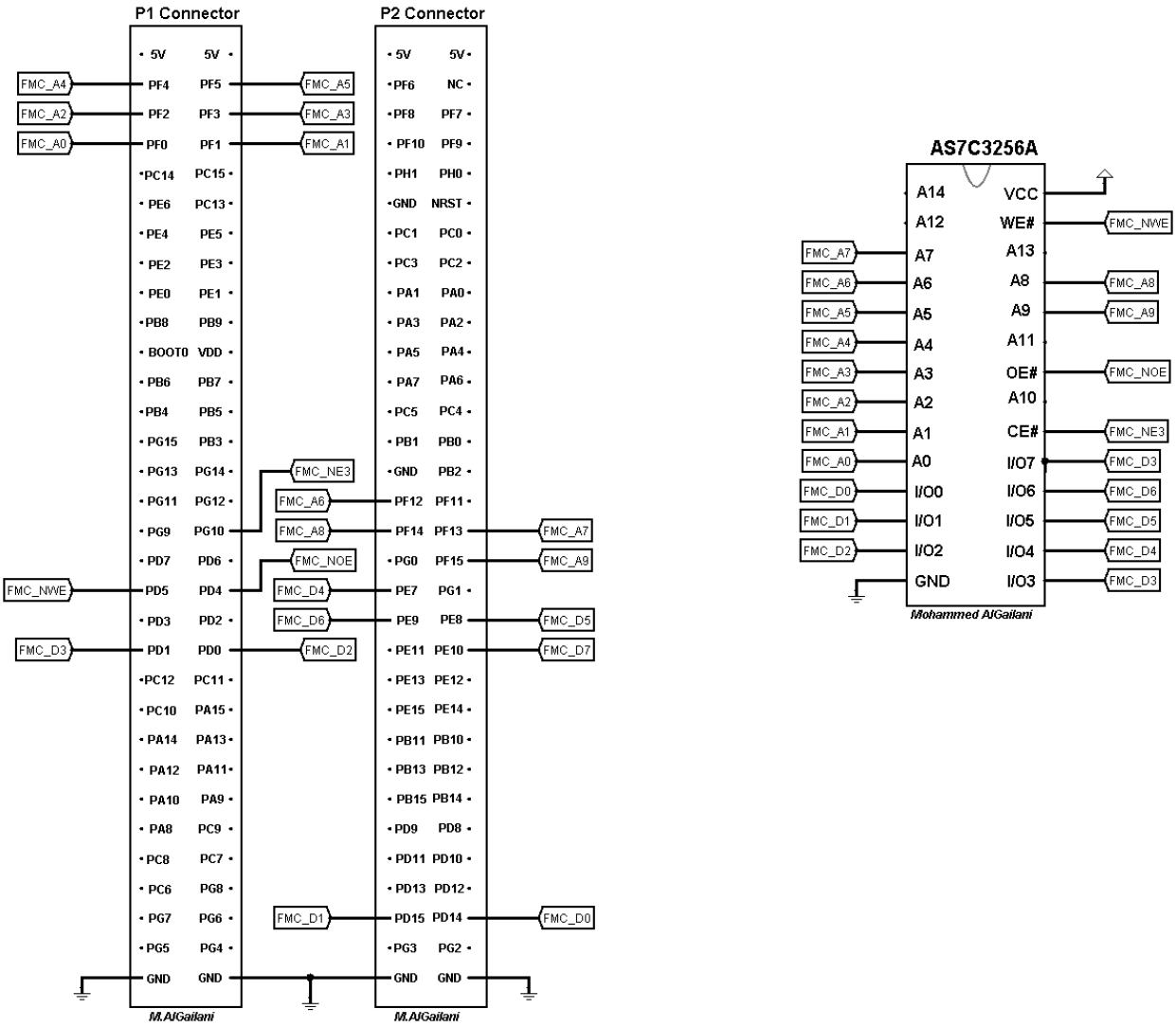

Figure 5: Schematic diagram for tasks 1 and 2

Task 2

As for this task, we were to develop a C program that would load the contents of a buffer area in memory into the contents of a block in the SRAM chip and then read those contents back and print them on the computer monitor. The data that is written in an infinite loop is 00,11,22,33,44,55,66,77,88,99,aa,bb,cc,dd,ee,ff, and repeating starting back at 00. The program would have to run in an infinite loop. In this task, we had to use the logic analyzer to verify the data.

Results

Task 1

We were asked in task 1 to write 0x3c to memory address 0x68000000 after clearing all of the next kilobyte.

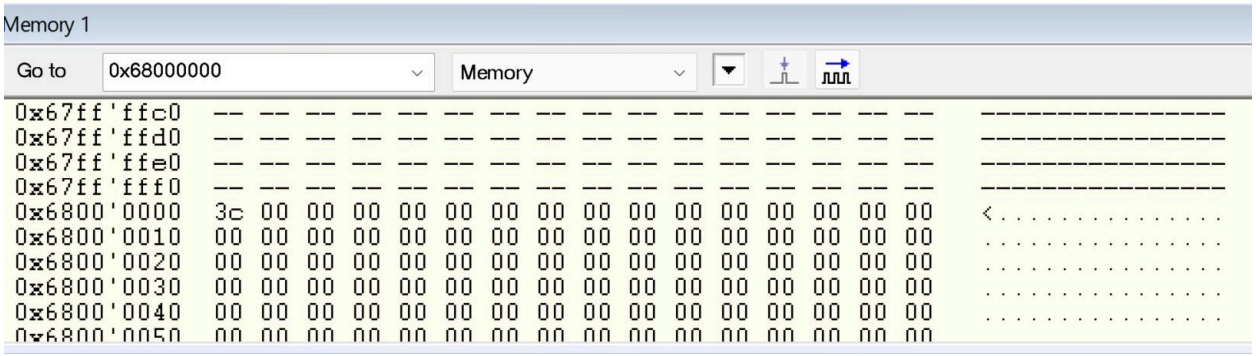

Figure 6: Memory window showing data byte 0x3C at address location 0x68000000 and the rest of the bits being cleared.

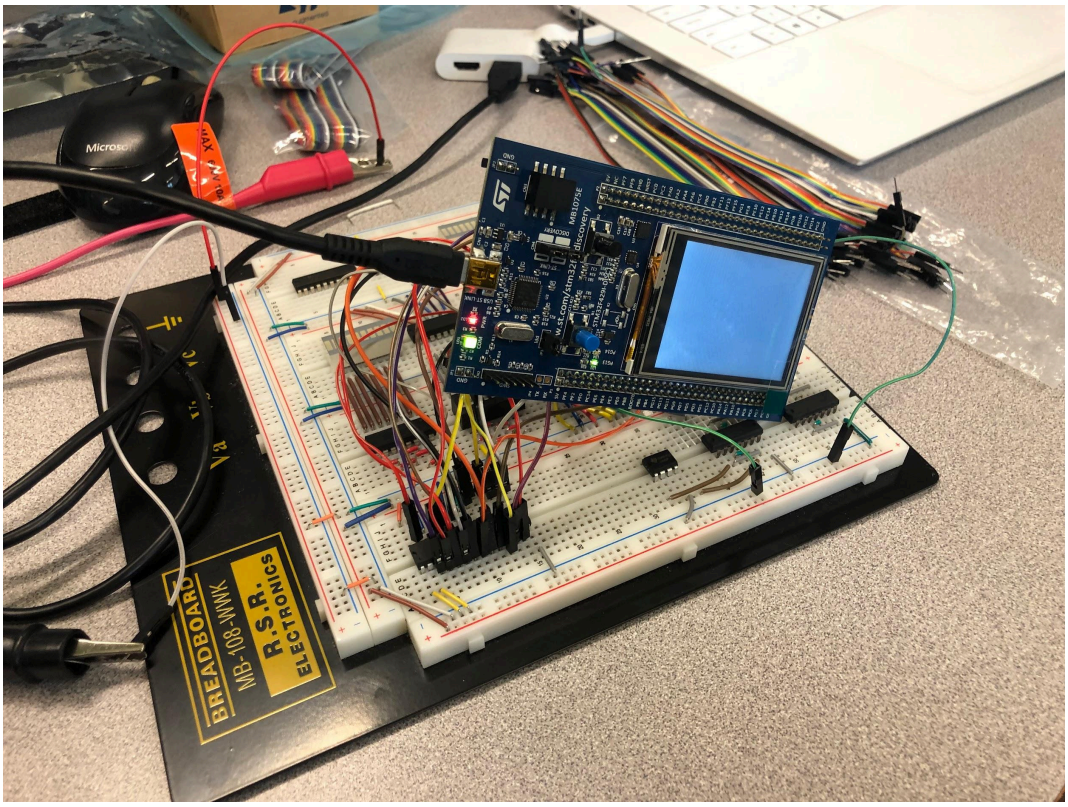

Figure 7: A picture of the setup showing Green LED light on the STM32 board indicating a correct read of the data.

Task 2

The goal for task 2 was to write a kilobyte of the repeating 0x00, 0x11, 0x22, … , 0xff pattern and read it back out using the logic analyzer and memory view.

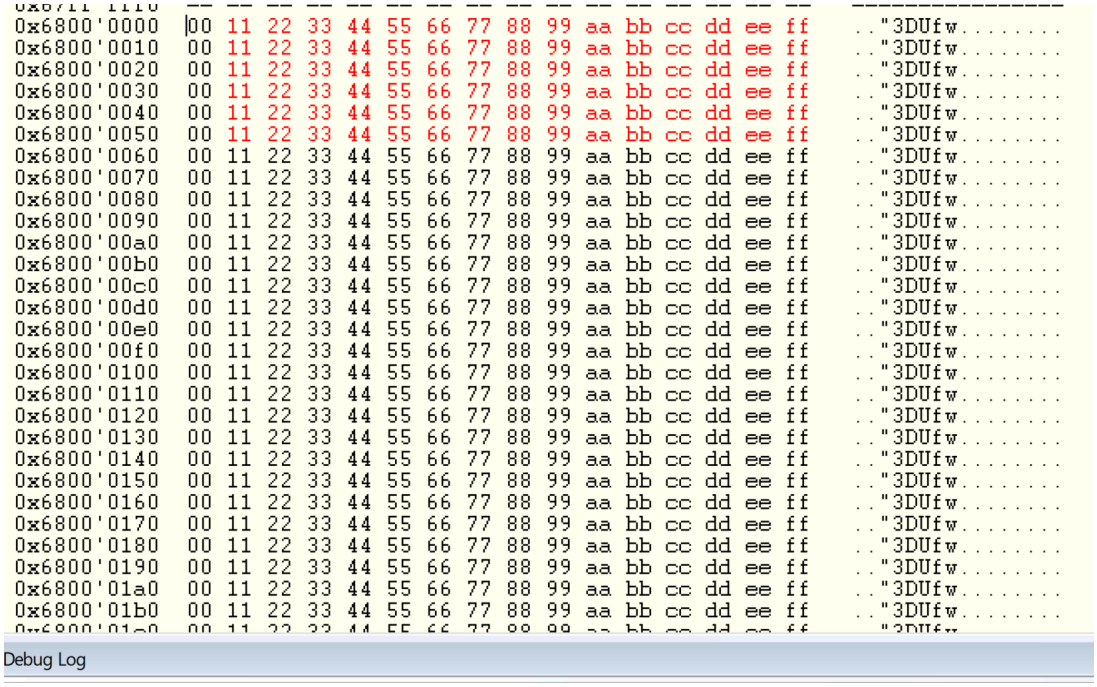

Figure 8: Window showing the program initializes a 16-byte write buffer in memory as: 0x00, 0x11, 0x22, …, 0xFF

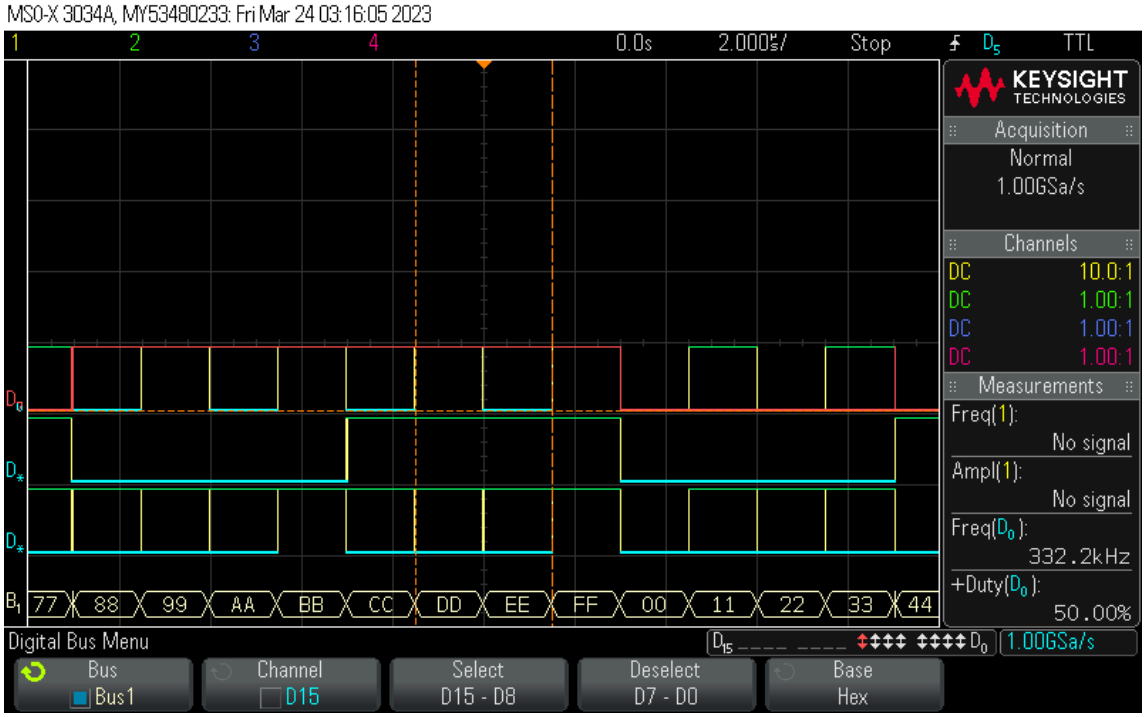

Figure 9: Logic Analyzer showing read data block at a point where the cycle repeats

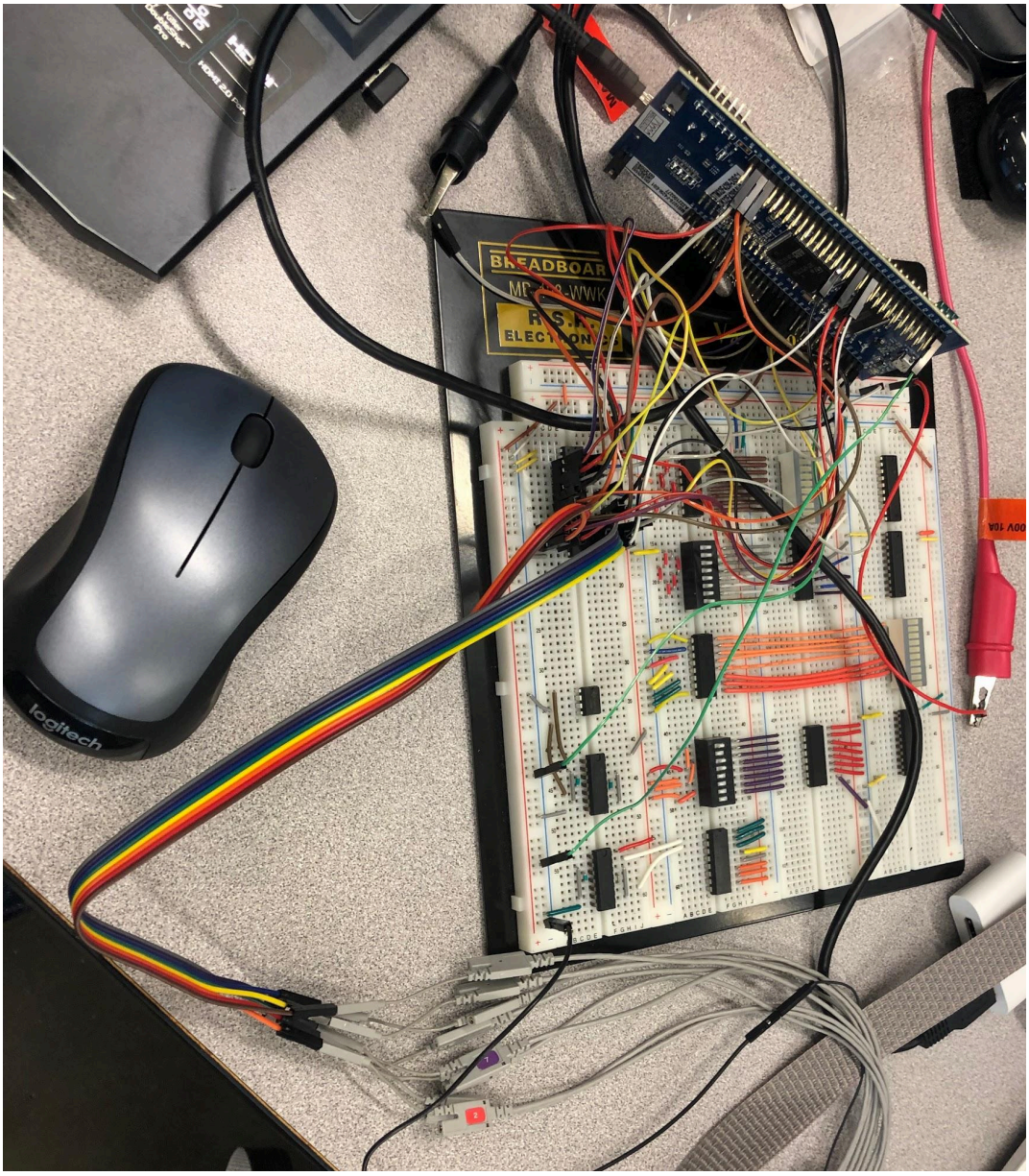

Figure 10: A picture of the setup connected to the logic analyzer

Conclusion

The lab helped us to learn and understand how to interface a microcontroller with external memory. The use of external memory is extremely important for the design of more complex systems. In all, we programmed the microcontroller to interface with an SRAM through the use of the Flexible Memory Controller (FMC) included on our STM32 board. Going forward, this will be an extremely useful skill in our careers in digital system design.

Appendix

C Code

Task 1

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 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 ---------------------------------------------------------*/
SRAM_HandleTypeDef hsram1;

/* USER CODE BEGIN PV */

uint32_t *address_type = (uint32_t*) 0x68000000;
uint8_t data_type = 0x3C;
uint32_t data_clear = 0x00000000;
uint8_t buffer_size;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_FMC_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_FMC_Init();
/* USER CODE BEGIN 2 */

HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_RESET);
HAL_SRAM_WriteOperation_Enable(&hsram1);
for(int i=0; i<0x100; i++){
    HAL_SRAM_Write_32b(&hsram1,address_type+i,&data_clear,1);
}
HAL_SRAM_Write_8b(&hsram1,address_type,&data_type,1);
//HAL_Delay(50);
HAL_SRAM_WriteOperation_Disable(&hsram1);
for(int i=0;i<10;i++){
   HAL_SRAM_Read_8b(&hsram1,address_type,&buffer_size,1);
   if(buffer_size==0x3C){
   HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_SET);
   }
   else{
   HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_SET);
}
}
/* 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};

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

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1
);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 180;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
```

```
    Error_Handler();
  }

  /** Activate the Over-Drive mode
  */
  if (HAL_PWREx_EnableOverDrive() != 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_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/* FMC initialization function */
static void MX_FMC_Init(void)
{

  /* USER CODE BEGIN FMC_Init 0 */

  /* USER CODE END FMC_Init 0 */

  FMC_NORSRAM_TimingTypeDef Timing = {0};
  FMC_NORSRAM_TimingTypeDef ExtTiming = {0};

  /* USER CODE BEGIN FMC_Init 1 */

  /* USER CODE END FMC_Init 1 */
```

```
/** Perform the SRAM1 memory initialization sequence
*/
hsram1.Instance = FMC_NORSRAM_DEVICE;
hsram1.Extended = FMC_NORSRAM_EXTENDED_DEVICE;
/* hsram1.Init */
hsram1.Init.NSBank = FMC_NORSRAM_BANK3;
hsram1.Init.DataAddressMux = FMC_DATA_ADDRESS_MUX_DISABLE;
hsram1.Init.MemoryType = FMC_MEMORY_TYPE_SRAM;
hsram1.Init.MemoryDataWidth = FMC_NORSRAM_MEM_BUS_WIDTH_8;
hsram1.Init.BurstAccessMode = FMC_BURST_ACCESS_MODE_DISABLE;
hsram1.Init.WaitSignalPolarity = FMC_WAIT_SIGNAL_POLARITY_LOW;
hsram1.Init.WrapMode = FMC_WRAP_MODE_DISABLE;
hsram1.Init.WaitSignalActive = FMC_WAIT_TIMING_BEFORE_WS;
hsram1.Init.WriteOperation = FMC_WRITE_OPERATION_ENABLE;
hsram1.Init.WaitSignal = FMC_WAIT_SIGNAL_DISABLE;
hsram1.Init.ExtendedMode = FMC_EXTENDED_MODE_ENABLE;
hsram1.Init.AsynchronousWait = FMC_ASYNCHRONOUS_WAIT_DISABLE;
hsram1.Init.WriteBurst = FMC_WRITE_BURST_DISABLE;
hsram1.Init.ContinuousClock = FMC_CONTINUOUS_CLOCK_SYNC_ONLY;
hsram1.Init.PageSize = FMC_PAGE_SIZE_NONE;
/* Timing */
Timing.AddressSetupTime = 15;
Timing.AddressHoldTime = 15;
Timing.DataSetupTime = 255;
Timing.BusTurnAroundDuration = 15;
Timing.CLKDivision = 16;
Timing.DataLatency = 17;
Timing.AccessMode = FMC_ACCESS_MODE_A;
/* ExtTiming */
ExtTiming.AddressSetupTime = 15;
ExtTiming.AddressHoldTime = 15;
ExtTiming.DataSetupTime = 255;
ExtTiming.BusTurnAroundDuration = 15;
ExtTiming.CLKDivision = 16;
ExtTiming.DataLatency = 17;
ExtTiming.AccessMode = FMC_ACCESS_MODE_A;

if (HAL_SRAM_Init(&hsram1, &Timing, &ExtTiming) != HAL_OK)
{
```

```
    Error_Handler( );
  }

  /* USER CODE BEGIN FMC_Init 2 */

  /* USER CODE END FMC_Init 2 */
}

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

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, Green_LED_Pin|Red_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : Green_LED_Pin Red_LED_Pin */
  GPIO_InitStruct.Pin = Green_LED_Pin|Red_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

}

/* 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 */
```

Task 2

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
```

```
  * @attention
  *
  * Copyright (c) 2023 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 ---------------------------------------------------------*/
SRAM_HandleTypeDef hsram1;

/* USER CODE BEGIN PV */

uint32_t *address_type = (uint32_t*) 0x68000000;
```

```
uint8_t data_type = 0x3C;
uint32_t data_clear = 0x00000000;
uint8_t buffer_size;
uint8_t data[16] = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa,
0xbb, 0xcc, 0xdd, 0xee, 0xff};

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_FMC_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_FMC_Init();
  /* USER CODE BEGIN 2 */

  HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_RESET);


  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {HAL_SRAM_WriteOperation_Enable(&hsram1);
  for (int j = 0; j<0x512; j+=4){
          HAL_SRAM_Write_8b(&hsram1, address_type+j, data, 16); //write the data until
the buffer is full
          }

  //HAL_Delay(50);
  HAL_SRAM_WriteOperation_Disable(&hsram1);
    /* 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_SCALE1
);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 180;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Activate the Over-Drive mode
  */
  if (HAL_PWREx_EnableOverDrive() != 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_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/* FMC initialization function */
static void MX_FMC_Init(void)
{

  /* USER CODE BEGIN FMC_Init 0 */

  /* USER CODE END FMC_Init 0 */

  FMC_NORSRAM_TimingTypeDef Timing = {0};
  FMC_NORSRAM_TimingTypeDef ExtTiming = {0};

  /* USER CODE BEGIN FMC_Init 1 */

  /* USER CODE END FMC_Init 1 */

  /** Perform the SRAM1 memory initialization sequence
  */
  hsram1.Instance = FMC_NORSRAM_DEVICE;
  hsram1.Extended = FMC_NORSRAM_EXTENDED_DEVICE;
  /* hsram1.Init */
  hsram1.Init.NSBank = FMC_NORSRAM_BANK3;
  hsram1.Init.DataAddressMux = FMC_DATA_ADDRESS_MUX_DISABLE;
  hsram1.Init.MemoryType = FMC_MEMORY_TYPE_SRAM;
  hsram1.Init.MemoryDataWidth = FMC_NORSRAM_MEM_BUS_WIDTH_8;
  hsram1.Init.BurstAccessMode = FMC_BURST_ACCESS_MODE_DISABLE;
  hsram1.Init.WaitSignalPolarity = FMC_WAIT_SIGNAL_POLARITY_LOW;
  hsram1.Init.WrapMode = FMC_WRAP_MODE_DISABLE;
  hsram1.Init.WaitSignalActive = FMC_WAIT_TIMING_BEFORE_WS;
  hsram1.Init.WriteOperation = FMC_WRITE_OPERATION_ENABLE;
```

```
  hsram1.Init.WaitSignal = FMC_WAIT_SIGNAL_DISABLE;
  hsram1.Init.ExtendedMode = FMC_EXTENDED_MODE_ENABLE;
  hsram1.Init.AsynchronousWait = FMC_ASYNCHRONOUS_WAIT_DISABLE;
  hsram1.Init.WriteBurst = FMC_WRITE_BURST_DISABLE;
  hsram1.Init.ContinuousClock = FMC_CONTINUOUS_CLOCK_SYNC_ONLY;
  hsram1.Init.PageSize = FMC_PAGE_SIZE_NONE;
  /* Timing */
  Timing.AddressSetupTime = 15;
  Timing.AddressHoldTime = 15;
  Timing.DataSetupTime = 255;
  Timing.BusTurnAroundDuration = 15;
  Timing.CLKDivision = 16;
  Timing.DataLatency = 17;
  Timing.AccessMode = FMC_ACCESS_MODE_A;
  /* ExtTiming */
  ExtTiming.AddressSetupTime = 15;
  ExtTiming.AddressHoldTime = 15;
  ExtTiming.DataSetupTime = 255;
  ExtTiming.BusTurnAroundDuration = 15;
  ExtTiming.CLKDivision = 16;
  ExtTiming.DataLatency = 17;
  ExtTiming.AccessMode = FMC_ACCESS_MODE_A;

  if (HAL_SRAM_Init(&hsram1, &Timing, &ExtTiming) != HAL_OK)
  {
    Error_Handler( );
  }

  /* USER CODE BEGIN FMC_Init 2 */

  /* USER CODE END FMC_Init 2 */
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
```

```
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, Green_LED_Pin|Red_LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : Green_LED_Pin Red_LED_Pin */
  GPIO_InitStruct.Pin = Green_LED_Pin|Red_LED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

}

/* 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 */
```

.lst Files

Task 1

```
###############################################################################
#########
#
# IAR ANSI C/C++ Compiler V9.20.4.327/W64 for ARM         23/Mar/2023  13:57:22
# Copyright 1999-2022 IAR Systems AB.
#
#    Cpu mode          =  thumb
#    Endian            =  little
#    Source file       =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src\
main.c
#    Command line      =
#        -f
```

```
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o.rsp
#
(C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src
\main.c
#       -D USE_HAL_DRIVER -D STM32F429xx -lC
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\List\Application\User\Core
#       -o
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core
#       --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
#       --dlib_config "C:\Program Files\IAR Systems\Embedded Workbench
#       9.0\arm\inc\c\DLib_Config_Full.h" -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Core/Inc\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/STM32F4xx_HAL_Driver/Inc\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/CMSIS/Device/ST/STM32F4xx/Include\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/CMSIS/Include\
#       -Ohz) --dependencies=n
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o.d
```

```
#    Locale          =  C
#    List file       =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\List\Application\User\Core\main.lst
#    Object file     =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o
#    Runtime model:
#      __CPP_Runtime   =  1
#      __SystemLibrary =  DLib
#      __dlib_version  =  6
#      __size_limit    =  32768|ARM.EW.LINKER
#
###############################################################################
#########

C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src\
main.c
      1        /* USER CODE BEGIN Header */
      2        /**
      3          ******************************************************************************
      4          * @file           : main.c
      5          * @brief          : Main program body
      6          ******************************************************************************
      7          * @attention
      8          *
      9          * Copyright (c) 2023 STMicroelectronics.
     10          * All rights reserved.
     11          *
     12          * This software is licensed under terms that can be found in the LICENSE
file
     13          * in the root directory of this software component.
     14          * If no LICENSE file comes with this software, it is provided AS-IS.
     15          *
     16          ******************************************************************************
     17          */
     18        /* USER CODE END Header */
     19        /* Includes ------------------------------------------------------------------*/
```

```
 20         #include "main.h"
 21
 22         /* Private includes ----------------------------------------------------------*/
 23         /* USER CODE BEGIN Includes */
 24
 25         /* USER CODE END Includes */
 26
 27         /* Private typedef -----------------------------------------------------------*/
 28         /* USER CODE BEGIN PTD */
 29
 30         /* USER CODE END PTD */
 31
 32         /* Private define ------------------------------------------------------------*/
 33         /* USER CODE BEGIN PD */
 34         /* USER CODE END PD */
 35
 36         /* Private macro -------------------------------------------------------------*/
 37         /* USER CODE BEGIN PM */
 38
 39         /* USER CODE END PM */
 40
 41         /* Private variables ---------------------------------------------------------*/

\                           In section .data, align 4
 42         SRAM_HandleTypeDef hsram1;
 43
 44         /* USER CODE BEGIN PV */
 45
 46         uint32_t *address_type = (uint32_t*) 0x68000000;
 47         uint8_t data_type = 0x3C;
 48         uint32_t data_clear = 0x00000000;
 49         uint8_t buffer_size;
\                    buffer_size:
\        0x0                DS8 1
\        0x1   0x00 0x00      DC8 0, 0, 0

\              0x00
\                    hsram1:
\        0x4   0x0000'0000    DC32 0x0, 0x0
```

```
\            0x0000'0000
\       0xC                  DS8 68
\      0x50   0x0000'0000       DC32 0x0
\      0x54                   DS8 0
\                 address_type:
\      0x54   0x6800'0000       DC32 0x6800'0000
\                 data_type:
\      0x58   0x3C             DC8 60
\      0x59   0x00 0x00         DC8 0, 0, 0

\            0x00
\                 data_clear:
\      0x5C   0x0000'0000       DC32 0
  50
  51        /* USER CODE END PV */
  52
  53        /* Private function prototypes -----------------------------------------------*/
  54        void SystemClock_Config(void);
  55        static void MX_GPIO_Init(void);
  56        static void MX_FMC_Init(void);
  57        /* USER CODE BEGIN PFP */
  58
  59        /* USER CODE END PFP */
  60
  61        /* Private user code ---------------------------------------------------------*/
  62        /* USER CODE BEGIN 0 */
  63
  64        /* USER CODE END 0 */
  65
  66        /**
  67          * @brief  The application entry point.
  68          * @retval int
  69          */

\                           In section .text, align 2, keep-with-next
  70        int main(void)
  71        {
\                 main: (+1)
\       0x0   0xB570            PUSH     {R4-R6,LR}
\       0x2   0xB08E            SUB      SP,SP,#+56
```

```
  72          /* USER CODE BEGIN 1 */
  73
  74          /* USER CODE END 1 */
  75
  76          /* MCU Configuration--------------------------------------------------------*/
  77
  78          /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  79          HAL_Init();
\       0x4   0x.... 0x....      BL       HAL_Init
  80
  81          /* USER CODE BEGIN Init */
  82
  83          /* USER CODE END Init */
  84
  85          /* Configure the system clock */
  86          SystemClock_Config();
\       0x8   0x.... 0x....      BL       SystemClock_Config
  87
  88          /* USER CODE BEGIN SysInit */
  89
  90          /* USER CODE END SysInit */
  91
  92          /* Initialize all configured peripherals */
  93          MX_GPIO_Init();
\       0xC   0x.... 0x....      BL       ?Subroutine0
\                  ??CrossCallReturnLabel_0: (+1)
\      0x10   0x2000             MOVS     R0,#+0
\      0x12   0x9000             STR      R0,[SP, #+0]
\      0x14   0x....             LDR.N    R5,??DataTable1
\      0x16   0x....             LDR.N    R0,??DataTable1_1
\      0x18   0x....             LDR.N    R6,??DataTable1_2
\      0x1A   0x6801             LDR      R1,[R0, #+0]
\      0x1C   0xF041 0x0120      ORR      R1,R1,#0x20
\      0x20   0x6001             STR      R1,[R0, #+0]
\      0x22   0x6802             LDR      R2,[R0, #+0]
\      0x24   0xF002 0x0220      AND      R2,R2,#0x20
\      0x28   0x9200             STR      R2,[SP, #+0]
\      0x2A   0x2200             MOVS     R2,#+0
\      0x2C   0x9900             LDR      R1,[SP, #+0]
\      0x2E   0x9200             STR      R2,[SP, #+0]
```

```
\      0x30   0x6803           LDR      R3,[R0, #+0]
\      0x32   0xF043 0x0380    ORR      R3,R3,#0x80
\      0x36   0x6003           STR      R3,[R0, #+0]
\      0x38   0x6801           LDR      R1,[R0, #+0]
\      0x3A   0xF001 0x0180    AND      R1,R1,#0x80
\      0x3E   0x9100           STR      R1,[SP, #+0]
\      0x40   0x9900           LDR      R1,[SP, #+0]
\      0x42   0x9200           STR      R2,[SP, #+0]
\      0x44   0x6803           LDR      R3,[R0, #+0]
\      0x46   0xF043 0x0310    ORR      R3,R3,#0x10
\      0x4A   0x6003           STR      R3,[R0, #+0]
\      0x4C   0x6801           LDR      R1,[R0, #+0]
\      0x4E   0xF001 0x0110    AND      R1,R1,#0x10
\      0x52   0x9100           STR      R1,[SP, #+0]
\      0x54   0x9900           LDR      R1,[SP, #+0]
\      0x56   0x9200           STR      R2,[SP, #+0]
\      0x58   0x6803           LDR      R3,[R0, #+0]
\      0x5A   0xF043 0x0308    ORR      R3,R3,#0x8
\      0x5E   0x6003           STR      R3,[R0, #+0]
\      0x60   0x6801           LDR      R1,[R0, #+0]
\      0x62   0xF001 0x0108    AND      R1,R1,#0x8
\      0x66   0x9100           STR      R1,[SP, #+0]
\      0x68   0x9900           LDR      R1,[SP, #+0]
\      0x6A   0x9200           STR      R2,[SP, #+0]
\      0x6C   0xF44F 0x41C0    MOV      R1,#+24576
\      0x70   0x6803           LDR      R3,[R0, #+0]
\      0x72   0xF043 0x0340    ORR      R3,R3,#0x40
\      0x76   0x6003           STR      R3,[R0, #+0]
\      0x78   0x6800           LDR      R0,[R0, #+0]
\      0x7A   0xF000 0x0040    AND      R0,R0,#0x40
\      0x7E   0x9000           STR      R0,[SP, #+0]
\      0x80   0x9800           LDR      R0,[SP, #+0]
\      0x82   0x.... 0x....    BL       ?Subroutine1
\                ??CrossCallReturnLabel_5: (+1)
\      0x86   0xF44F 0x41C0    MOV      R1,#+24576
\      0x8A   0x9101           STR      R1,[SP, #+4]
\      0x8C   0x2201           MOVS     R2,#+1
\      0x8E   0x2100           MOVS     R1,#+0
\      0x90   0x9103           STR      R1,[SP, #+12]
\      0x92   0x9104           STR      R1,[SP, #+16]
```

```
\       0x94   0x9202           STR      R2,[SP, #+8]
\       0x96   0xA901           ADD      R1,SP,#+4
\       0x98   0x4628           MOV      R0,R5
\       0x9A   0x.... 0x....    BL       HAL_GPIO_Init
  94          MX_FMC_Init();
\       0x9E   0x221C           MOVS     R2,#+28
\       0xA0   0x2100           MOVS     R1,#+0
\       0xA2   0xA807           ADD      R0,SP,#+28
\       0xA4   0x.... 0x....    BL       memset
\       0xA8   0x221C           MOVS     R2,#+28
\       0xAA   0x2100           MOVS     R1,#+0
\       0xAC   0x4668           MOV      R0,SP
\       0xAE   0x.... 0x....    BL       memset
\       0xB2   0xF04F 0x4020    MOV      R0,#+2684354560
\       0xB6   0x....           LDR.N    R1,??DataTable1_3
\       0xB8   0x6070           STR      R0,[R6, #+4]
\       0xBA   0x60B1           STR      R1,[R6, #+8]
\       0xBC   0x2004           MOVS     R0,#+4
\       0xBE   0x2100           MOVS     R1,#+0
\       0xC0   0x220F           MOVS     R2,#+15
\       0xC2   0x60F0           STR      R0,[R6, #+12]
\       0xC4   0x6131           STR      R1,[R6, #+16]
\       0xC6   0x6171           STR      R1,[R6, #+20]
\       0xC8   0x61B1           STR      R1,[R6, #+24]
\       0xCA   0x61F1           STR      R1,[R6, #+28]
\       0xCC   0x6231           STR      R1,[R6, #+32]
\       0xCE   0x6271           STR      R1,[R6, #+36]
\       0xD0   0x62B1           STR      R1,[R6, #+40]
\       0xD2   0x6331           STR      R1,[R6, #+48]
\       0xD4   0x63B1           STR      R1,[R6, #+56]
\       0xD6   0x6431           STR      R1,[R6, #+64]
\       0xD8   0x9208           STR      R2,[SP, #+32]
\       0xDA   0x920A           STR      R2,[SP, #+40]
\       0xDC   0xF44F 0x5080    MOV      R0,#+4096
\       0xE0   0x210F           MOVS     R1,#+15
\       0xE2   0x2211           MOVS     R2,#+17
\       0xE4   0x62F0           STR      R0,[R6, #+44]
\       0xE6   0x9107           STR      R1,[SP, #+28]
\       0xE8   0x920C           STR      R2,[SP, #+48]
\       0xEA   0xF44F 0x4080    MOV      R0,#+16384
```

```
\      0xEE   0x21FF           MOVS     R1,#+255
\      0xF0   0x220F           MOVS     R2,#+15
\      0xF2   0x6370           STR      R0,[R6, #+52]
\      0xF4   0x9109           STR      R1,[SP, #+36]
\      0xF6   0x9200           STR      R2,[SP, #+0]
\      0xF8   0x9201           STR      R2,[SP, #+4]
\      0xFA   0x9102           STR      R1,[SP, #+8]
\      0xFC   0x9203           STR      R2,[SP, #+12]
\      0xFE   0x2000           MOVS     R0,#+0
\     0x100   0x2210           MOVS     R2,#+16
\     0x102   0x2111           MOVS     R1,#+17
\     0x104   0x63F0           STR      R0,[R6, #+60]
\     0x106   0x64B0           STR      R0,[R6, #+72]
\     0x108   0x2310           MOVS     R3,#+16
\     0x10A   0x900D           STR      R0,[SP, #+52]
\     0x10C   0x9204           STR      R2,[SP, #+16]
\     0x10E   0x9105           STR      R1,[SP, #+20]
\     0x110   0x9006           STR      R0,[SP, #+24]
\     0x112   0x930B           STR      R3,[SP, #+44]
\     0x114   0x466A           MOV      R2,SP
\     0x116   0xA907           ADD      R1,SP,#+28
\     0x118   0x1D30           ADDS     R0,R6,#+4
\     0x11A   0x.... 0x....    BL       HAL_SRAM_Init
\     0x11E   0xB108           CBZ.N    R0,??main_0
\     0x120   0x.... 0x....    BL       Error_Handler
  95          /* USER CODE BEGIN 2 */
  96
  97          HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_RESET);
\               ??main_0: (+1)
\     0x124   0x2200           MOVS     R2,#+0
\     0x126   0xF44F 0x5100    MOV      R1,#+8192
\     0x12A   0x.... 0x....    BL       ?Subroutine1
  98          HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_RESET);
\               ??CrossCallReturnLabel_4: (+1)
\     0x12E   0x2200           MOVS     R2,#+0
\     0x130   0xF44F 0x4180    MOV      R1,#+16384
\     0x134   0x.... 0x....    BL       ?Subroutine1
  99          HAL_SRAM_WriteOperation_Enable(&hsram1);
\               ??CrossCallReturnLabel_3: (+1)
\     0x138   0x1D30           ADDS     R0,R6,#+4
```

```
\     0x13A   0x.... 0x....      BL       HAL_SRAM_WriteOperation_Enable
 100          for(int i=0; i<0x100; i++){
\     0x13E   0x2400             MOVS     R4,#+0
 101            HAL_SRAM_Write_32b(&hsram1,address_type+i,&data_clear,1);
\                ??main_1: (+1)
\     0x140   0x6D70             LDR      R0,[R6, #+84]
\     0x142   0xEB00 0x0184      ADD      R1,R0,R4, LSL #+2
\     0x146   0x2301             MOVS     R3,#+1
\     0x148   0xF106 0x025C      ADD      R2,R6,#+92
\     0x14C   0x1D30             ADDS     R0,R6,#+4
\     0x14E   0x.... 0x....      BL       HAL_SRAM_Write_32b
 102          }
\     0x152   0x1C64             ADDS     R4,R4,#+1
\     0x154   0xF5B4 0x7F80      CMP      R4,#+256
\     0x158   0xDBF2             BLT.N    ??main_1
 103          HAL_SRAM_Write_8b(&hsram1,address_type,&data_type,1);
\     0x15A   0x6D71             LDR      R1,[R6, #+84]
\     0x15C   0x2301             MOVS     R3,#+1
\     0x15E   0xF106 0x0258      ADD      R2,R6,#+88
\     0x162   0x1D30             ADDS     R0,R6,#+4
\     0x164   0x.... 0x....      BL       HAL_SRAM_Write_8b
 104          //HAL_Delay(50);
 105          HAL_SRAM_WriteOperation_Disable(&hsram1);
\     0x168   0x1D30             ADDS     R0,R6,#+4
\     0x16A   0x.... 0x....      BL       HAL_SRAM_WriteOperation_Disable
 106          for(int i=0;i<10;i++){
\     0x16E   0x240A             MOVS     R4,#+10
 107            HAL_SRAM_Read_8b(&hsram1,address_type,&buffer_size,1);
\                ??main_2: (+1)
\     0x170   0x6D71             LDR      R1,[R6, #+84]
\     0x172   0x2301             MOVS     R3,#+1
\     0x174   0x4632             MOV      R2,R6
\     0x176   0x1D30             ADDS     R0,R6,#+4
\     0x178   0x.... 0x....      BL       HAL_SRAM_Read_8b
 108            if(buffer_size==0x3C){
\     0x17C   0x7830             LDRB     R0,[R6, #+0]
\     0x17E   0x283C             CMP      R0,#+60
\     0x180   0xBF07             ITTEE    EQ
\     0x182   0x2201             MOVEQ    R2,#+1
\     0x184   0xF44F 0x5100      MOVEQ    R1,#+8192
```

```
\     0x188   0x2201           MOVNE    R2,#+1
\     0x18A   0xF44F 0x4180    MOVNE    R1,#+16384
 109            HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_SET);
 110            }
 111            else{
 112            HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_SET);
\     0x18E   0x.... 0x....    BL       ?Subroutine1
 113          }
 114          }
\               ??CrossCallReturnLabel_2: (+1)
\     0x192   0x1E64           SUBS     R4,R4,#+1
\     0x194   0xD1EC           BNE.N    ??main_2
 115          /* USER CODE END 2 */
 116
 117          /* Infinite loop */
 118          /* USER CODE BEGIN WHILE */
 119          while (1)
\               ??main_3: (+1)
\     0x196   0xE7FE           B.N      ??main_3
 120          {
 121            /* USER CODE END WHILE */
 122
 123            /* USER CODE BEGIN 3 */
 124          }
 125          /* USER CODE END 3 */
 126        }

\                       In section .text, align 2, keep-with-next
\               ?Subroutine1: (+1)
\       0x0   0x4628           MOV      R0,R5
\       0x2   0x.... 0x....    B.W      HAL_GPIO_WritePin
 127
 128        /**
 129          * @brief System Clock Configuration
 130          * @retval None
 131          */

\                       In section .text, align 2, keep-with-next
 132        void SystemClock_Config(void)
 133        {
```

```
  \                 SystemClock_Config: (+1)
  \       0x0   0xB580            PUSH     {R7,LR}
  \       0x2   0xB092            SUB      SP,SP,#+72
  \       0x4   0x2230            MOVS     R2,#+48
  \       0x6   0x2100            MOVS     R1,#+0
  \       0x8   0xA806            ADD      R0,SP,#+24
  \       0xA   0x.... 0x....     BL       memset
  \       0xE   0x.... 0x....     BL       ?Subroutine0
   134          RCC_OscInitTypeDef RCC_OscInitStruct = {0};
   135          RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
   136
   137          /** Configure the main internal regulator output voltage
   138          */
   139          __HAL_RCC_PWR_CLK_ENABLE();
  \                 ??CrossCallReturnLabel_1: (+1)
  \      0x12   0x2000            MOVS     R0,#+0
  \      0x14   0x9000            STR      R0,[SP, #+0]
   140
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1
);
   141
   142          /** Initializes the RCC Oscillators according to the specified parameters
   143          * in the RCC_OscInitTypeDef structure.
   144          */
   145          RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
   146          RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  \      0x16   0xF44F 0x3380      MOV      R3,#+65536
  \      0x1A   0x....             LDR.N    R0,??DataTable1_4
  \      0x1C   0x6801             LDR      R1,[R0, #+0]
  \      0x1E   0xF041 0x5180      ORR      R1,R1,#0x10000000
  \      0x22   0x6001             STR      R1,[R0, #+0]
  \      0x24   0x2100             MOVS     R1,#+0
  \      0x26   0x6800             LDR      R0,[R0, #+0]
  \      0x28   0xF000 0x5080      AND      R0,R0,#0x10000000
  \      0x2C   0x9000             STR      R0,[SP, #+0]
  \      0x2E   0x9800             LDR      R0,[SP, #+0]
  \      0x30   0x....             LDR.N    R0,??DataTable1_5
  \      0x32   0x9100             STR      R1,[SP, #+0]
  \      0x34   0x6802             LDR      R2,[R0, #+0]
  \      0x36   0xF442 0x4240      ORR      R2,R2,#0xC000
```

```
\      0x3A   0x6002           STR      R2,[R0, #+0]
\      0x3C   0x2201           MOVS     R2,#+1
\      0x3E   0x6800           LDR      R0,[R0, #+0]
\      0x40   0xF400 0x4040    AND      R0,R0,#0xC000
\      0x44   0x9000           STR      R0,[SP, #+0]
 147          RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 148          RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
 149          RCC_OscInitStruct.PLL.PLLM = 4;
 150          RCC_OscInitStruct.PLL.PLLN = 180;
 151          RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
 152          RCC_OscInitStruct.PLL.PLLQ = 4;
 153          if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
\      0x46   0xA806           ADD      R0,SP,#+24
\      0x48   0x9900           LDR      R1,[SP, #+0]
\      0x4A   0x9206           STR      R2,[SP, #+24]
\      0x4C   0x2102           MOVS     R1,#+2
\      0x4E   0xF44F 0x0280    MOV      R2,#+4194304
\      0x52   0x910C           STR      R1,[SP, #+48]
\      0x54   0x920D           STR      R2,[SP, #+52]
\      0x56   0x2104           MOVS     R1,#+4
\      0x58   0x22B4           MOVS     R2,#+180
\      0x5A   0x910E           STR      R1,[SP, #+56]
\      0x5C   0x920F           STR      R2,[SP, #+60]
\      0x5E   0x2102           MOVS     R1,#+2
\      0x60   0x2204           MOVS     R2,#+4
\      0x62   0x9307           STR      R3,[SP, #+28]
\      0x64   0x9110           STR      R1,[SP, #+64]
\      0x66   0x9211           STR      R2,[SP, #+68]
\      0x68   0x.... 0x....    BL       HAL_RCC_OscConfig
\      0x6C   0xB108           CBZ.N    R0,??SystemClock_Config_0
 154          {
 155            Error_Handler();
\      0x6E   0x.... 0x....    BL       Error_Handler
 156          }
 157
 158          /** Activate the Over-Drive mode
 159          */
 160          if (HAL_PWREx_EnableOverDrive() != HAL_OK)
\                ??SystemClock_Config_0: (+1)
\      0x72   0x.... 0x....    BL       HAL_PWREx_EnableOverDrive
```

```
  \      0x76   0xB108           CBZ.N    R0,??SystemClock_Config_1
   161          {
   162            Error_Handler();
  \      0x78   0xB672           CPSID    I
  \                ??SystemClock_Config_2: (+1)
  \      0x7A   0xE7FE           B.N      ??SystemClock_Config_2
   163          }
   164
   165          /** Initializes the CPU, AHB and APB buses clocks
   166          */
   167          RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
   168
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
   169          RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  \                ??SystemClock_Config_1: (+1)
  \      0x7C   0x2102           MOVS     R1,#+2
  \      0x7E   0x9102           STR      R1,[SP, #+8]
   170          RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  \      0x80   0x2200           MOVS     R2,#+0
  \      0x82   0x9203           STR      R2,[SP, #+12]
  \      0x84   0x200F           MOVS     R0,#+15
   171          RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  \      0x86   0xF44F 0x51A0    MOV      R1,#+5120
  \      0x8A   0x9001           STR      R0,[SP, #+4]
  \      0x8C   0x9104           STR      R1,[SP, #+16]
   172          RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
  \      0x8E   0xF44F 0x5280    MOV      R2,#+4096
  \      0x92   0x9205           STR      R2,[SP, #+20]
   173
   174          if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) !=
HAL_OK)
  \      0x94   0x2105           MOVS     R1,#+5
  \      0x96   0xA801           ADD      R0,SP,#+4
  \      0x98   0x.... 0x....    BL       HAL_RCC_ClockConfig
  \      0x9C   0xB108           CBZ.N    R0,??SystemClock_Config_3
   175          {
   176            Error_Handler();
  \      0x9E   0xB672           CPSID    I
  \                ??SystemClock_Config_4: (+1)
```

```
\      0xA0   0xE7FE            B.N      ??SystemClock_Config_4
 177          }
 178        }
\                ??SystemClock_Config_3: (+1)
\      0xA2   0xB013            ADD      SP,SP,#+76
\      0xA4   0xBD00            POP      {PC}

\                           In section .text, align 2, keep-with-next
\                ?Subroutine0: (+1)
\       0x0   0x2214            MOVS     R2,#+20
\       0x2   0x2100            MOVS     R1,#+0
\       0x4   0xA801            ADD      R0,SP,#+4
\       0x6   0x.... 0x....     B.W      memset
 179
 180        /* FMC initialization function */
 181        static void MX_FMC_Init(void)
 182        {
 183
 184          /* USER CODE BEGIN FMC_Init 0 */
 185
 186          /* USER CODE END FMC_Init 0 */
 187
 188          FMC_NORSRAM_TimingTypeDef Timing = {0};
 189          FMC_NORSRAM_TimingTypeDef ExtTiming = {0};
 190
 191          /* USER CODE BEGIN FMC_Init 1 */
 192
 193          /* USER CODE END FMC_Init 1 */
 194
 195          /** Perform the SRAM1 memory initialization sequence
 196          */
 197          hsram1.Instance = FMC_NORSRAM_DEVICE;
 198          hsram1.Extended = FMC_NORSRAM_EXTENDED_DEVICE;
 199          /* hsram1.Init */
 200          hsram1.Init.NSBank = FMC_NORSRAM_BANK3;
 201          hsram1.Init.DataAddressMux = FMC_DATA_ADDRESS_MUX_DISABLE;
 202          hsram1.Init.MemoryType = FMC_MEMORY_TYPE_SRAM;
 203          hsram1.Init.MemoryDataWidth =
FMC_NORSRAM_MEM_BUS_WIDTH_8;
```

```
  204         hsram1.Init.BurstAccessMode =
FMC_BURST_ACCESS_MODE_DISABLE;
  205         hsram1.Init.WaitSignalPolarity = FMC_WAIT_SIGNAL_POLARITY_LOW;
  206         hsram1.Init.WrapMode = FMC_WRAP_MODE_DISABLE;
  207         hsram1.Init.WaitSignalActive = FMC_WAIT_TIMING_BEFORE_WS;
  208         hsram1.Init.WriteOperation = FMC_WRITE_OPERATION_ENABLE;
  209         hsram1.Init.WaitSignal = FMC_WAIT_SIGNAL_DISABLE;
  210         hsram1.Init.ExtendedMode = FMC_EXTENDED_MODE_ENABLE;
  211         hsram1.Init.AsynchronousWait =
FMC_ASYNCHRONOUS_WAIT_DISABLE;
  212         hsram1.Init.WriteBurst = FMC_WRITE_BURST_DISABLE;
  213         hsram1.Init.ContinuousClock =
FMC_CONTINUOUS_CLOCK_SYNC_ONLY;
  214         hsram1.Init.PageSize = FMC_PAGE_SIZE_NONE;
  215         /* Timing */
  216         Timing.AddressSetupTime = 15;
  217         Timing.AddressHoldTime = 15;
  218         Timing.DataSetupTime = 255;
  219         Timing.BusTurnAroundDuration = 15;
  220         Timing.CLKDivision = 16;
  221         Timing.DataLatency = 17;
  222         Timing.AccessMode = FMC_ACCESS_MODE_A;
  223         /* ExtTiming */
  224         ExtTiming.AddressSetupTime = 15;
  225         ExtTiming.AddressHoldTime = 15;
  226         ExtTiming.DataSetupTime = 255;
  227         ExtTiming.BusTurnAroundDuration = 15;
  228         ExtTiming.CLKDivision = 16;
  229         ExtTiming.DataLatency = 17;
  230         ExtTiming.AccessMode = FMC_ACCESS_MODE_A;
  231
  232         if (HAL_SRAM_Init(&hsram1, &Timing, &ExtTiming) != HAL_OK)
  233         {
  234           Error_Handler( );
  235         }
  236
  237         /* USER CODE BEGIN FMC_Init 2 */
  238
  239         /* USER CODE END FMC_Init 2 */
  240       }
```

```
  241
  242       /**
  243         * @brief GPIO Initialization Function
  244         * @param None
  245         * @retval None
  246         */
  247       static void MX_GPIO_Init(void)
  248       {
  249         GPIO_InitTypeDef GPIO_InitStruct = {0};
  250
  251         /* GPIO Ports Clock Enable */
  252         __HAL_RCC_GPIOF_CLK_ENABLE();
  253         __HAL_RCC_GPIOH_CLK_ENABLE();
  254         __HAL_RCC_GPIOE_CLK_ENABLE();
  255         __HAL_RCC_GPIOD_CLK_ENABLE();
  256         __HAL_RCC_GPIOG_CLK_ENABLE();
  257
  258         /*Configure GPIO pin Output Level */
  259         HAL_GPIO_WritePin(GPIOG, Green_LED_Pin|Red_LED_Pin,
GPIO_PIN_RESET);
  260
  261         /*Configure GPIO pins : Green_LED_Pin Red_LED_Pin */
  262         GPIO_InitStruct.Pin = Green_LED_Pin|Red_LED_Pin;
  263         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  264         GPIO_InitStruct.Pull = GPIO_NOPULL;
  265         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  266         HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
  267
  268       }
  269
  270       /* USER CODE BEGIN 4 */
  271
  272       /* USER CODE END 4 */
  273
  274       /**
  275         * @brief  This function is executed in case of error occurrence.
  276         * @retval None
  277         */

  \                        In section .text, align 2, keep-with-next
```

```
  278        void Error_Handler(void)
  279        {
  280          /* USER CODE BEGIN Error_Handler_Debug */
  281          /* User can add his own implementation to report the HAL error return
state */
  282          __disable_irq();
  \                 Error_Handler: (+1)
  \       0x0   0xB672            CPSID    I
  283          while (1)
  \                 ??Error_Handler_0: (+1)
  \       0x2   0xE7FE            B.N      ??Error_Handler_0
  284          {
  285          }
  286          /* USER CODE END Error_Handler_Debug */
  287        }

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1:
  \       0x0   0x4002'1800       DC32     0x40021800

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1_1:
  \       0x0   0x4002'3830       DC32     0x40023830

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1_2:
  \       0x0   0x....'....       DC32     buffer_size

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1_3:
  \       0x0   0xA000'0104       DC32     0xa0000104

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1_4:
  \       0x0   0x4002'3840       DC32     0x40023840

  \                         In section .text, align 4, keep-with-next
  \                 ??DataTable1_5:
  \       0x0   0x4000'7000       DC32     0x40007000
  288
```

```
  289        #ifdef  USE_FULL_ASSERT
  290        /**
  291          * @brief  Reports the name of the source file and the source line number
  292          *         where the assert_param error has occurred.
  293          * @param  file: pointer to the source file name
  294          * @param  line: assert_param error line source number
  295          * @retval None
  296          */
  297        void assert_failed(uint8_t *file, uint32_t line)
  298        {
  299          /* USER CODE BEGIN 6 */
  300          /* User can add his own implementation to report the file name and line
number,
  301             ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  302          /* USER CODE END 6 */
  303        }
  304        #endif /* USE_FULL_ASSERT */

  Maximum stack usage in bytes:

  .cstack Function
  ------- --------
     0   Error_Handler
    80   SystemClock_Config
      80   -> Error_Handler
      80   -> HAL_PWREx_EnableOverDrive
      80   -> HAL_RCC_ClockConfig
      80   -> HAL_RCC_OscConfig
      80   -> memset
    72   main
      72   -> Error_Handler
      72   -> HAL_GPIO_Init
      72   -> HAL_GPIO_WritePin
      72   -> HAL_Init
      72   -> HAL_SRAM_Init
      72   -> HAL_SRAM_Read_8b
      72   -> HAL_SRAM_WriteOperation_Disable
      72   -> HAL_SRAM_WriteOperation_Enable
      72   -> HAL_SRAM_Write_32b
      72   -> HAL_SRAM_Write_8b
```

```
      72   -> SystemClock_Config
      72   -> memset


  Section sizes:

  Bytes  Function/Label
  -----  --------------
      4  ??DataTable1
      4  ??DataTable1_1
      4  ??DataTable1_2
      4  ??DataTable1_3
      4  ??DataTable1_4
      4  ??DataTable1_5
     10  ?Subroutine0
      6  ?Subroutine1
      4  Error_Handler
    166  SystemClock_Config
     96  buffer_size
         hsram1
         address_type
         data_type
         data_clear
    408  main


   96 bytes in section .data
  618 bytes in section .text

  618 bytes of CODE memory
   96 bytes of DATA memory

Errors: none
Warnings: none
```

Task 2

```
#############################################################################
#########
#
# IAR ANSI C/C++ Compiler V9.20.4.327/W64 for ARM         23/Mar/2023  14:21:04
# Copyright 1999-2022 IAR Systems AB.
#
#    Cpu mode          =  thumb
#    Endian            =  little
#    Source file       =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src\
main.c
#    Command line      =
#        -f
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o.rsp
#
(C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src
\main.c
#        -D USE_HAL_DRIVER -D STM32F429xx -lC
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\List\Application\User\Core
#        -o
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core
#        --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
#        --dlib_config "C:\Program Files\IAR Systems\Embedded Workbench
#        9.0\arm\inc\c\DLib_Config_Full.h" -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Core/Inc\
#        -I
```

```
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/STM32F4xx_HAL_Driver/Inc\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/CMSIS/Device/ST/STM32F4xx/Include\
#       -I
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM/
../Drivers/CMSIS/Include\
#       -Ohz) --dependencies=n
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o.d
#    Locale           =  C
#    List file        =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\List\Application\User\Core\main.lst
#    Object file      =
#
C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\EWARM\
Lab10_Configuration\Obj\Application\User\Core\main.o
#    Runtime model:
#      __CPP_Runtime   =  1
#      __SystemLibrary =  DLib
#      __dlib_version  =  6
#      __size_limit    =  32768|ARM.EW.LINKER
#
################################################################################
#########

C:\Users\azeR\Documents\ECE4510_Lab\Lab10\Codes\Lab10_Configuration\Core\Src\
main.c
      1         /* USER CODE BEGIN Header */
```

```
  2 /**
  3   ******************************************************************************
  4   * @file           : main.c
  5   * @brief          : Main program body
  6   ******************************************************************************
  7   * @attention
  8   *
  9   * Copyright (c) 2023 STMicroelectronics.
 10   * All rights reserved.
 11   *
 12   * This software is licensed under terms that can be found in the LICENSE
file
 13   * in the root directory of this software component.
 14   * If no LICENSE file comes with this software, it is provided AS-IS.
 15   *
 16   ******************************************************************************
 17   */
 18 /* USER CODE END Header */
 19 /* Includes ------------------------------------------------------------------*/
 20 #include "main.h"
 21
 22 /* Private includes ----------------------------------------------------------*/
 23 /* USER CODE BEGIN Includes */
 24
 25 /* USER CODE END Includes */
 26
 27 /* Private typedef -----------------------------------------------------------*/
 28 /* USER CODE BEGIN PTD */
 29
 30 /* USER CODE END PTD */
 31
 32 /* Private define ------------------------------------------------------------*/
 33 /* USER CODE BEGIN PD */
 34 /* USER CODE END PD */
 35
 36 /* Private macro -------------------------------------------------------------*/
 37 /* USER CODE BEGIN PM */
 38
 39 /* USER CODE END PM */
 40
```

```
  41        /* Private variables ---------------------------------------------------------*/
  42        SRAM_HandleTypeDef hsram1;
  43
  44        /* USER CODE BEGIN PV */
  45
  46        uint32_t *address_type = (uint32_t*) 0x68000000;
  47        uint8_t data_type = 0x3C;

\                           In section .bss, align 4
  48        uint32_t data_clear = 0x00000000;
\               data_clear:
\       0x0                 DS8 4

\                           In section .bss, align 1
  49        uint8_t buffer_size;
\               buffer_size:
\       0x0                 DS8 1

\                           In section .data, align 4
\               hsram1:
\       0x0   0x0000'0000       DC32 0x0, 0x0

\           0x0000'0000
\       0x8                 DS8 68
\      0x4C   0x0000'0000       DC32 0x0
\      0x50                 DS8 0
\               address_type:
\      0x50   0x6800'0000       DC32 0x6800'0000
  50        uint8_t data[16] = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88,
0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff};
\               `data`:
\      0x54   0x00 0x11         DC8 0, 17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187,
204, 221, 238

\           0x22 0x33

\           0x44 0x55

\           0x66 0x77
```

```
\            0x88 0x99

\            0xAA 0xBB

\            0xCC 0xDD

\            0xEE
\      0x63   0xFF              DC8 255

\                           In section .data, align 1
\                 data_type:
\       0x0   0x3C              DC8 60
  51
  52          /* USER CODE END PV */
  53
  54          /* Private function prototypes -----------------------------------------------*/
  55          void SystemClock_Config(void);
  56          static void MX_GPIO_Init(void);
  57          static void MX_FMC_Init(void);
  58          /* USER CODE BEGIN PFP */
  59
  60          /* USER CODE END PFP */
  61
  62          /* Private user code ---------------------------------------------------------*/
  63          /* USER CODE BEGIN 0 */
  64
  65          /* USER CODE END 0 */
  66
  67          /**
  68            * @brief  The application entry point.
  69            * @retval int
  70            */

\                           In section .text, align 2, keep-with-next
  71          int main(void)
  72          {
\                 main: (+1)
\       0x0   0xB538            PUSH     {R3-R5,LR}
\       0x2   0xB08E            SUB      SP,SP,#+56
  73            /* USER CODE BEGIN 1 */
```

```
  74
  75          /* USER CODE END 1 */
  76
  77          /* MCU Configuration--------------------------------------------------------*/
  78
  79          /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  80          HAL_Init();
\       0x4   0x.... 0x....      BL       HAL_Init
  81
  82          /* USER CODE BEGIN Init */
  83
  84          /* USER CODE END Init */
  85
  86          /* Configure the system clock */
  87          SystemClock_Config();
\       0x8   0x.... 0x....      BL       SystemClock_Config
  88
  89          /* USER CODE BEGIN SysInit */
  90
  91          /* USER CODE END SysInit */
  92
  93          /* Initialize all configured peripherals */
  94          MX_GPIO_Init();
\       0xC   0x.... 0x....      BL       ?Subroutine0
\                  ??CrossCallReturnLabel_0: (+1)
\      0x10   0x2000             MOVS     R0,#+0
\      0x12   0x9000             STR      R0,[SP, #+0]
\      0x14   0x....             LDR.N    R4,??DataTable1
\      0x16   0x....             LDR.N    R0,??DataTable1_1
\      0x18   0x....             LDR.N    R5,??DataTable1_2
\      0x1A   0x6801             LDR      R1,[R0, #+0]
\      0x1C   0xF041 0x0120      ORR      R1,R1,#0x20
\      0x20   0x6001             STR      R1,[R0, #+0]
\      0x22   0x6802             LDR      R2,[R0, #+0]
\      0x24   0xF002 0x0220      AND      R2,R2,#0x20
\      0x28   0x9200             STR      R2,[SP, #+0]
\      0x2A   0x2200             MOVS     R2,#+0
\      0x2C   0x9900             LDR      R1,[SP, #+0]
\      0x2E   0x9200             STR      R2,[SP, #+0]
\      0x30   0x6803             LDR      R3,[R0, #+0]
```

```
\       0x32   0xF043 0x0380      ORR      R3,R3,#0x80
\       0x36   0x6003             STR      R3,[R0, #+0]
\       0x38   0x6801             LDR      R1,[R0, #+0]
\       0x3A   0xF001 0x0180      AND      R1,R1,#0x80
\       0x3E   0x9100             STR      R1,[SP, #+0]
\       0x40   0x9900             LDR      R1,[SP, #+0]
\       0x42   0x9200             STR      R2,[SP, #+0]
\       0x44   0x6803             LDR      R3,[R0, #+0]
\       0x46   0xF043 0x0310      ORR      R3,R3,#0x10
\       0x4A   0x6003             STR      R3,[R0, #+0]
\       0x4C   0x6801             LDR      R1,[R0, #+0]
\       0x4E   0xF001 0x0110      AND      R1,R1,#0x10
\       0x52   0x9100             STR      R1,[SP, #+0]
\       0x54   0x9900             LDR      R1,[SP, #+0]
\       0x56   0x9200             STR      R2,[SP, #+0]
\       0x58   0x6803             LDR      R3,[R0, #+0]
\       0x5A   0xF043 0x0308      ORR      R3,R3,#0x8
\       0x5E   0x6003             STR      R3,[R0, #+0]
\       0x60   0x6801             LDR      R1,[R0, #+0]
\       0x62   0xF001 0x0108      AND      R1,R1,#0x8
\       0x66   0x9100             STR      R1,[SP, #+0]
\       0x68   0x9900             LDR      R1,[SP, #+0]
\       0x6A   0x9200             STR      R2,[SP, #+0]
\       0x6C   0xF44F 0x41C0      MOV      R1,#+24576
\       0x70   0x6803             LDR      R3,[R0, #+0]
\       0x72   0xF043 0x0340      ORR      R3,R3,#0x40
\       0x76   0x6003             STR      R3,[R0, #+0]
\       0x78   0x6800             LDR      R0,[R0, #+0]
\       0x7A   0xF000 0x0040      AND      R0,R0,#0x40
\       0x7E   0x9000             STR      R0,[SP, #+0]
\       0x80   0x9800             LDR      R0,[SP, #+0]
\       0x82   0x4620             MOV      R0,R4
\       0x84   0x.... 0x....      BL       HAL_GPIO_WritePin
\       0x88   0xF44F 0x41C0      MOV      R1,#+24576
\       0x8C   0x9101             STR      R1,[SP, #+4]
\       0x8E   0x2201             MOVS     R2,#+1
\       0x90   0x2100             MOVS     R1,#+0
\       0x92   0x9103             STR      R1,[SP, #+12]
\       0x94   0x9104             STR      R1,[SP, #+16]
\       0x96   0x9202             STR      R2,[SP, #+8]
```

```
\      0x98   0xA901            ADD      R1,SP,#+4
\      0x9A   0x4620            MOV      R0,R4
\      0x9C   0x.... 0x....     BL       HAL_GPIO_Init
  95          MX_FMC_Init();
\      0xA0   0x221C            MOVS     R2,#+28
\      0xA2   0x2100            MOVS     R1,#+0
\      0xA4   0xA807            ADD      R0,SP,#+28
\      0xA6   0x.... 0x....     BL       memset
\      0xAA   0x221C            MOVS     R2,#+28
\      0xAC   0x2100            MOVS     R1,#+0
\      0xAE   0x4668            MOV      R0,SP
\      0xB0   0x.... 0x....     BL       memset
\      0xB4   0x....            LDR.N    R1,??DataTable1_3
\      0xB6   0x6069            STR      R1,[R5, #+4]
\      0xB8   0x2204            MOVS     R2,#+4
\      0xBA   0x60AA            STR      R2,[R5, #+8]
\      0xBC   0xF44F 0x5180     MOV      R1,#+4096
\      0xC0   0x220F            MOVS     R2,#+15
\      0xC2   0x62A9            STR      R1,[R5, #+40]
\      0xC4   0x9207            STR      R2,[SP, #+28]
\      0xC6   0x9208            STR      R2,[SP, #+32]
\      0xC8   0x920A            STR      R2,[SP, #+40]
\      0xCA   0xF44F 0x4180     MOV      R1,#+16384
\      0xCE   0x2211            MOVS     R2,#+17
\      0xD0   0x6329            STR      R1,[R5, #+48]
\      0xD2   0x920C            STR      R2,[SP, #+48]
\      0xD4   0xF04F 0x4020     MOV      R0,#+2684354560
\      0xD8   0x21FF            MOVS     R1,#+255
\      0xDA   0x220F            MOVS     R2,#+15
\      0xDC   0x6028            STR      R0,[R5, #+0]
\      0xDE   0x9109            STR      R1,[SP, #+36]
\      0xE0   0x9200            STR      R2,[SP, #+0]
\      0xE2   0x9201            STR      R2,[SP, #+4]
\      0xE4   0x9102            STR      R1,[SP, #+8]
\      0xE6   0x9203            STR      R2,[SP, #+12]
\      0xE8   0x2000            MOVS     R0,#+0
\      0xEA   0x2210            MOVS     R2,#+16
\      0xEC   0x2111            MOVS     R1,#+17
\      0xEE   0x60E8            STR      R0,[R5, #+12]
\      0xF0   0x6128            STR      R0,[R5, #+16]
```

```
\      0xF2   0x6168           STR      R0,[R5, #+20]
\      0xF4   0x61A8           STR      R0,[R5, #+24]
\      0xF6   0x61E8           STR      R0,[R5, #+28]
\      0xF8   0x6228           STR      R0,[R5, #+32]
\      0xFA   0x6268           STR      R0,[R5, #+36]
\      0xFC   0x62E8           STR      R0,[R5, #+44]
\      0xFE   0x6368           STR      R0,[R5, #+52]
\     0x100   0x63A8           STR      R0,[R5, #+56]
\     0x102   0x63E8           STR      R0,[R5, #+60]
\     0x104   0x6468           STR      R0,[R5, #+68]
\     0x106   0x2310           MOVS     R3,#+16
\     0x108   0x900D           STR      R0,[SP, #+52]
\     0x10A   0x9204           STR      R2,[SP, #+16]
\     0x10C   0x9105           STR      R1,[SP, #+20]
\     0x10E   0x9006           STR      R0,[SP, #+24]
\     0x110   0x930B           STR      R3,[SP, #+44]
\     0x112   0x466A           MOV      R2,SP
\     0x114   0xA907           ADD      R1,SP,#+28
\     0x116   0x4628           MOV      R0,R5
\     0x118   0x.... 0x....    BL       HAL_SRAM_Init
\     0x11C   0xB108           CBZ.N    R0,??main_0
\     0x11E   0x.... 0x....    BL       Error_Handler
  96          /* USER CODE BEGIN 2 */
  97
  98          HAL_GPIO_WritePin(GPIOG,GPIO_PIN_13,GPIO_PIN_RESET);
\               ??main_0: (+1)
\     0x122   0x2200           MOVS     R2,#+0
\     0x124   0xF44F 0x5100    MOV      R1,#+8192
\     0x128   0x4620           MOV      R0,R4
\     0x12A   0x.... 0x....    BL       HAL_GPIO_WritePin
  99          HAL_GPIO_WritePin(GPIOG,GPIO_PIN_14,GPIO_PIN_RESET);
\     0x12E   0x2200           MOVS     R2,#+0
\     0x130   0xF44F 0x4180    MOV      R1,#+16384
\     0x134   0x4620           MOV      R0,R4
\     0x136   0x.... 0x....    BL       HAL_GPIO_WritePin
 100
 101
 102          /* USER CODE END 2 */
 103
 104          /* Infinite loop */
```

```
  105           /* USER CODE BEGIN WHILE */
  106           while (1)
  107           {HAL_SRAM_WriteOperation_Enable(&hsram1);
  \                 ??main_1: (+1)
  \     0x13A   0x4628            MOV      R0,R5
  \     0x13C   0x.... 0x....     BL       HAL_SRAM_WriteOperation_Enable
  108           for (int j = 0; j<0x512; j+=4){
  \     0x140   0x2400            MOVS     R4,#+0
  109                   HAL_SRAM_Write_8b(&hsram1, address_type+j, data, 16); //write
the data until the buffer is full
  \                 ??main_2: (+1)
  \     0x142   0x6D28            LDR      R0,[R5, #+80]
  \     0x144   0xEB00 0x0184     ADD      R1,R0,R4, LSL #+2
  \     0x148   0x2310            MOVS     R3,#+16
  \     0x14A   0xF105 0x0254     ADD      R2,R5,#+84
  \     0x14E   0x4628            MOV      R0,R5
  \     0x150   0x.... 0x....     BL       HAL_SRAM_Write_8b
  110           }
  \     0x154   0x1D24            ADDS     R4,R4,#+4
  \     0x156   0xF240 0x5012     MOVW     R0,#+1298
  \     0x15A   0x4284            CMP      R4,R0
  \     0x15C   0xDBF1            BLT.N    ??main_2
  111
  112           //HAL_Delay(50);
  113           HAL_SRAM_WriteOperation_Disable(&hsram1);
  \     0x15E   0x4628            MOV      R0,R5
  \     0x160   0x.... 0x....     BL       HAL_SRAM_WriteOperation_Disable
  \     0x164   0xE7E9            B.N      ??main_1
  114           /* USER CODE END WHILE */
  115
  116           /* USER CODE BEGIN 3 */
  117           }
  118           /* USER CODE END 3 */
  119         }
  120
  121         /**
  122           * @brief System Clock Configuration
  123           * @retval None
  124           */
```

```
\                          In section .text, align 2, keep-with-next
 125        void SystemClock_Config(void)
 126        {
\               SystemClock_Config: (+1)
\       0x0  0xB580           PUSH     {R7,LR}
\       0x2  0xB092           SUB      SP,SP,#+72
\       0x4  0x2230           MOVS     R2,#+48
\       0x6  0x2100           MOVS     R1,#+0
\       0x8  0xA806           ADD      R0,SP,#+24
\       0xA  0x.... 0x....    BL       memset
\       0xE  0x.... 0x....    BL       ?Subroutine0
 127          RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 128          RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 129
 130          /** Configure the main internal regulator output voltage
 131          */
 132          __HAL_RCC_PWR_CLK_ENABLE();
\               ??CrossCallReturnLabel_1: (+1)
\      0x12  0x2000           MOVS     R0,#+0
\      0x14  0x9000           STR      R0,[SP, #+0]
 133
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1
);
 134
 135          /** Initializes the RCC Oscillators according to the specified parameters
 136          * in the RCC_OscInitTypeDef structure.
 137          */
 138          RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
 139          RCC_OscInitStruct.HSEState = RCC_HSE_ON;
\      0x16  0xF44F 0x3380     MOV      R3,#+65536
\      0x1A  0x....            LDR.N    R0,??DataTable1_4
\      0x1C  0x6801            LDR      R1,[R0, #+0]
\      0x1E  0xF041 0x5180     ORR      R1,R1,#0x10000000
\      0x22  0x6001            STR      R1,[R0, #+0]
\      0x24  0x2100            MOVS     R1,#+0
\      0x26  0x6800            LDR      R0,[R0, #+0]
\      0x28  0xF000 0x5080     AND      R0,R0,#0x10000000
\      0x2C  0x9000            STR      R0,[SP, #+0]
\      0x2E  0x9800            LDR      R0,[SP, #+0]
\      0x30  0x....            LDR.N    R0,??DataTable1_5
```

```
\      0x32   0x9100           STR      R1,[SP, #+0]
\      0x34   0x6802           LDR      R2,[R0, #+0]
\      0x36   0xF442 0x4240    ORR      R2,R2,#0xC000
\      0x3A   0x6002           STR      R2,[R0, #+0]
\      0x3C   0x2201           MOVS     R2,#+1
\      0x3E   0x6800           LDR      R0,[R0, #+0]
\      0x40   0xF400 0x4040    AND      R0,R0,#0xC000
\      0x44   0x9000           STR      R0,[SP, #+0]
 140          RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 141          RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
 142          RCC_OscInitStruct.PLL.PLLM = 4;
 143          RCC_OscInitStruct.PLL.PLLN = 180;
 144          RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
 145          RCC_OscInitStruct.PLL.PLLQ = 4;
 146          if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
\      0x46   0xA806           ADD      R0,SP,#+24
\      0x48   0x9900           LDR      R1,[SP, #+0]
\      0x4A   0x9206           STR      R2,[SP, #+24]
\      0x4C   0x2102           MOVS     R1,#+2
\      0x4E   0xF44F 0x0280    MOV      R2,#+4194304
\      0x52   0x910C           STR      R1,[SP, #+48]
\      0x54   0x920D           STR      R2,[SP, #+52]
\      0x56   0x2104           MOVS     R1,#+4
\      0x58   0x22B4           MOVS     R2,#+180
\      0x5A   0x910E           STR      R1,[SP, #+56]
\      0x5C   0x920F           STR      R2,[SP, #+60]
\      0x5E   0x2102           MOVS     R1,#+2
\      0x60   0x2204           MOVS     R2,#+4
\      0x62   0x9307           STR      R3,[SP, #+28]
\      0x64   0x9110           STR      R1,[SP, #+64]
\      0x66   0x9211           STR      R2,[SP, #+68]
\      0x68   0x.... 0x....    BL       HAL_RCC_OscConfig
\      0x6C   0xB108           CBZ.N    R0,??SystemClock_Config_0
 147          {
 148            Error_Handler();
\      0x6E   0x.... 0x....    BL       Error_Handler
 149          }
 150
 151          /** Activate the Over-Drive mode
 152          */
```

```
  153          if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  \                ??SystemClock_Config_0: (+1)
  \      0x72   0x.... 0x....      BL       HAL_PWREx_EnableOverDrive
  \      0x76   0xB108             CBZ.N    R0,??SystemClock_Config_1
  154          {
  155            Error_Handler();
  \      0x78   0xB672             CPSID    I
  \                ??SystemClock_Config_2: (+1)
  \      0x7A   0xE7FE             B.N      ??SystemClock_Config_2
  156          }
  157
  158          /** Initializes the CPU, AHB and APB buses clocks
  159          */
  160          RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
  161
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  162          RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  \                ??SystemClock_Config_1: (+1)
  \      0x7C   0x2102             MOVS     R1,#+2
  \      0x7E   0x9102             STR      R1,[SP, #+8]
  163          RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  \      0x80   0x2200             MOVS     R2,#+0
  \      0x82   0x9203             STR      R2,[SP, #+12]
  \      0x84   0x200F             MOVS     R0,#+15
  164          RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  \      0x86   0xF44F 0x51A0      MOV      R1,#+5120
  \      0x8A   0x9001             STR      R0,[SP, #+4]
  \      0x8C   0x9104             STR      R1,[SP, #+16]
  165          RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
  \      0x8E   0xF44F 0x5280      MOV      R2,#+4096
  \      0x92   0x9205             STR      R2,[SP, #+20]
  166
  167          if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) !=
HAL_OK)
  \      0x94   0x2105             MOVS     R1,#+5
  \      0x96   0xA801             ADD      R0,SP,#+4
  \      0x98   0x.... 0x....      BL       HAL_RCC_ClockConfig
  \      0x9C   0xB108             CBZ.N    R0,??SystemClock_Config_3
  168          {
```

```
 169            Error_Handler();
\      0x9E   0xB672            CPSID    I
\                 ??SystemClock_Config_4: (+1)
\      0xA0   0xE7FE            B.N      ??SystemClock_Config_4
 170          }
 171        }
\                 ??SystemClock_Config_3: (+1)
\      0xA2   0xB013            ADD      SP,SP,#+76
\      0xA4   0xBD00            POP      {PC}

\                           In section .text, align 2, keep-with-next
\                 ?Subroutine0: (+1)
\       0x0   0x2214            MOVS     R2,#+20
\       0x2   0x2100            MOVS     R1,#+0
\       0x4   0xA801            ADD      R0,SP,#+4
\       0x6   0x.... 0x....     B.W      memset
 172
 173        /* FMC initialization function */
 174        static void MX_FMC_Init(void)
 175        {
 176
 177          /* USER CODE BEGIN FMC_Init 0 */
 178
 179          /* USER CODE END FMC_Init 0 */
 180
 181          FMC_NORSRAM_TimingTypeDef Timing = {0};
 182          FMC_NORSRAM_TimingTypeDef ExtTiming = {0};
 183
 184          /* USER CODE BEGIN FMC_Init 1 */
 185
 186          /* USER CODE END FMC_Init 1 */
 187
 188          /** Perform the SRAM1 memory initialization sequence
 189          */
 190          hsram1.Instance = FMC_NORSRAM_DEVICE;
 191          hsram1.Extended = FMC_NORSRAM_EXTENDED_DEVICE;
 192          /* hsram1.Init */
 193          hsram1.Init.NSBank = FMC_NORSRAM_BANK3;
 194          hsram1.Init.DataAddressMux = FMC_DATA_ADDRESS_MUX_DISABLE;
 195          hsram1.Init.MemoryType = FMC_MEMORY_TYPE_SRAM;
```

```
  196         hsram1.Init.MemoryDataWidth =
FMC_NORSRAM_MEM_BUS_WIDTH_8;
  197         hsram1.Init.BurstAccessMode =
FMC_BURST_ACCESS_MODE_DISABLE;
  198         hsram1.Init.WaitSignalPolarity = FMC_WAIT_SIGNAL_POLARITY_LOW;
  199         hsram1.Init.WrapMode = FMC_WRAP_MODE_DISABLE;
  200         hsram1.Init.WaitSignalActive = FMC_WAIT_TIMING_BEFORE_WS;
  201         hsram1.Init.WriteOperation = FMC_WRITE_OPERATION_ENABLE;
  202         hsram1.Init.WaitSignal = FMC_WAIT_SIGNAL_DISABLE;
  203         hsram1.Init.ExtendedMode = FMC_EXTENDED_MODE_ENABLE;
  204         hsram1.Init.AsynchronousWait =
FMC_ASYNCHRONOUS_WAIT_DISABLE;
  205         hsram1.Init.WriteBurst = FMC_WRITE_BURST_DISABLE;
  206         hsram1.Init.ContinuousClock =
FMC_CONTINUOUS_CLOCK_SYNC_ONLY;
  207         hsram1.Init.PageSize = FMC_PAGE_SIZE_NONE;
  208         /* Timing */
  209         Timing.AddressSetupTime = 15;
  210         Timing.AddressHoldTime = 15;
  211         Timing.DataSetupTime = 255;
  212         Timing.BusTurnAroundDuration = 15;
  213         Timing.CLKDivision = 16;
  214         Timing.DataLatency = 17;
  215         Timing.AccessMode = FMC_ACCESS_MODE_A;
  216         /* ExtTiming */
  217         ExtTiming.AddressSetupTime = 15;
  218         ExtTiming.AddressHoldTime = 15;
  219         ExtTiming.DataSetupTime = 255;
  220         ExtTiming.BusTurnAroundDuration = 15;
  221         ExtTiming.CLKDivision = 16;
  222         ExtTiming.DataLatency = 17;
  223         ExtTiming.AccessMode = FMC_ACCESS_MODE_A;
  224
  225         if (HAL_SRAM_Init(&hsram1, &Timing, &ExtTiming) != HAL_OK)
  226         {
  227           Error_Handler( );
  228         }
  229
  230         /* USER CODE BEGIN FMC_Init 2 */
  231
```

```
  232         /* USER CODE END FMC_Init 2 */
  233       }
  234
  235       /**
  236         * @brief GPIO Initialization Function
  237         * @param None
  238         * @retval None
  239         */
  240       static void MX_GPIO_Init(void)
  241       {
  242         GPIO_InitTypeDef GPIO_InitStruct = {0};
  243
  244         /* GPIO Ports Clock Enable */
  245         __HAL_RCC_GPIOF_CLK_ENABLE();
  246         __HAL_RCC_GPIOH_CLK_ENABLE();
  247         __HAL_RCC_GPIOE_CLK_ENABLE();
  248         __HAL_RCC_GPIOD_CLK_ENABLE();
  249         __HAL_RCC_GPIOG_CLK_ENABLE();
  250
  251         /*Configure GPIO pin Output Level */
  252         HAL_GPIO_WritePin(GPIOG, Green_LED_Pin|Red_LED_Pin,
GPIO_PIN_RESET);
  253
  254         /*Configure GPIO pins : Green_LED_Pin Red_LED_Pin */
  255         GPIO_InitStruct.Pin = Green_LED_Pin|Red_LED_Pin;
  256         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  257         GPIO_InitStruct.Pull = GPIO_NOPULL;
  258         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  259         HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
  260
  261       }
  262
  263       /* USER CODE BEGIN 4 */
  264
  265       /* USER CODE END 4 */
  266
  267       /**
  268         * @brief  This function is executed in case of error occurrence.
  269         * @retval None
  270         */
```

```
  \                        In section .text, align 2, keep-with-next
   271        void Error_Handler(void)
   272        {
   273          /* USER CODE BEGIN Error_Handler_Debug */
   274          /* User can add his own implementation to report the HAL error return
state */
   275          __disable_irq();
  \                 Error_Handler: (+1)
  \       0x0   0xB672           CPSID    I
   276          while (1)
  \                 ??Error_Handler_0: (+1)
  \       0x2   0xE7FE           B.N      ??Error_Handler_0
   277          {
   278          }
   279          /* USER CODE END Error_Handler_Debug */
   280        }

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1:
  \       0x0   0x4002'1800       DC32     0x40021800

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1_1:
  \       0x0   0x4002'3830       DC32     0x40023830

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1_2:
  \       0x0   0x....'....       DC32     hsram1

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1_3:
  \       0x0   0xA000'0104       DC32     0xa0000104

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1_4:
  \       0x0   0x4002'3840       DC32     0x40023840

  \                        In section .text, align 4, keep-with-next
  \                 ??DataTable1_5:
```

```
  \        0x0   0x4000'7000        DC32     0x40007000
   281
   282          #ifdef  USE_FULL_ASSERT
   283          /**
   284            * @brief  Reports the name of the source file and the source line number
   285            *         where the assert_param error has occurred.
   286            * @param  file: pointer to the source file name
   287            * @param  line: assert_param error line source number
   288            * @retval None
   289            */
   290          void assert_failed(uint8_t *file, uint32_t line)
   291          {
   292            /* USER CODE BEGIN 6 */
   293            /* User can add his own implementation to report the file name and line
number,
   294               ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
   295            /* USER CODE END 6 */
   296          }
   297          #endif /* USE_FULL_ASSERT */

  Maximum stack usage in bytes:

  .cstack Function
  ------- --------
     0   Error_Handler
    80   SystemClock_Config
      80   -> Error_Handler
      80   -> HAL_PWREx_EnableOverDrive
      80   -> HAL_RCC_ClockConfig
      80   -> HAL_RCC_OscConfig
      80   -> memset
    72   main
      72   -> Error_Handler
      72   -> HAL_GPIO_Init
      72   -> HAL_GPIO_WritePin
      72   -> HAL_Init
      72   -> HAL_SRAM_Init
      72   -> HAL_SRAM_WriteOperation_Disable
      72   -> HAL_SRAM_WriteOperation_Enable
      72   -> HAL_SRAM_Write_8b
```

```
    72   -> SystemClock_Config
    72   -> memset


  Section sizes:

  Bytes  Function/Label
  -----  --------------
     4  ??DataTable1
     4  ??DataTable1_1
     4  ??DataTable1_2
     4  ??DataTable1_3
     4  ??DataTable1_4
     4  ??DataTable1_5
    10  ?Subroutine0
     4  Error_Handler
   166  SystemClock_Config
     1  buffer_size
     4  data_clear
     1  data_type
   100  hsram1
          address_type
          data
   358  main


   5 bytes in section .bss
 101 bytes in section .data
 562 bytes in section .text

 562 bytes of CODE memory
 106 bytes of DATA memory

Errors: none
Warnings: none
```