Electrical and Computer Engineering Department
ECE 4510 Microcontroller Applications

Lab 8

Temperature Measurement

Isaac Bagley and Rohullah Sah

April 6, 2023

Introduction

The purpose of this lab was to do work on three different tasks. The first task was to display decimal numbers in an incrementing fashion from 0 to 212 at the display rate of 1.0 KHz in an infinite loop using an LCD. As for the second task, we had to display the temperature in Celsius and Fahrenheit depending upon the status of the switch for which one is to be displayed using an LM35 temperature sensor. And for the last task, we had to develop a C program for the Microchip TCN75A temperature sensor. The temperature values are sampled at 250ms intervals with DIP-switch being the mode to modify between Celsius and Fahrenheit with Fahrenheit being high and Celsius vice-versa.

Procedure

Task 1

In this task, we were to develop a C program to display decimal numbers from 0 to 212 in an infinite loop using an LCD.

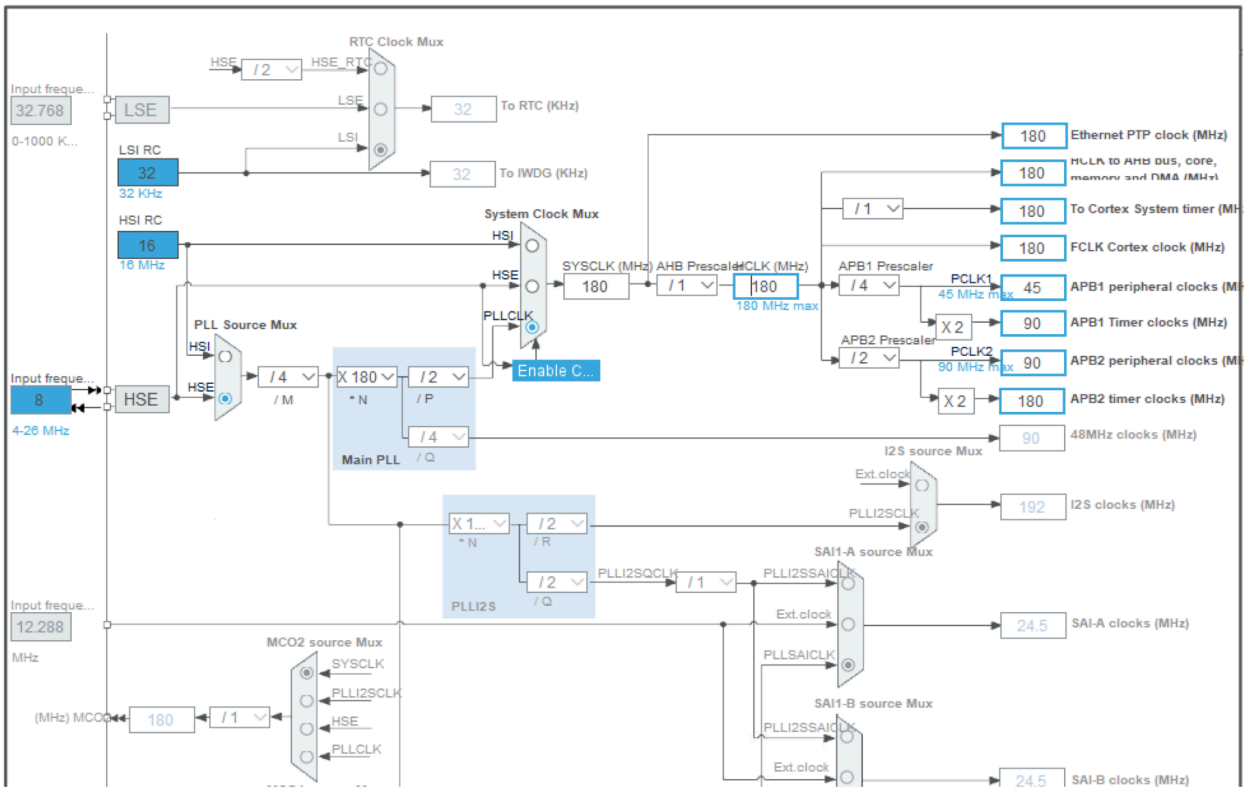

Figure 1: Clock Configuration

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

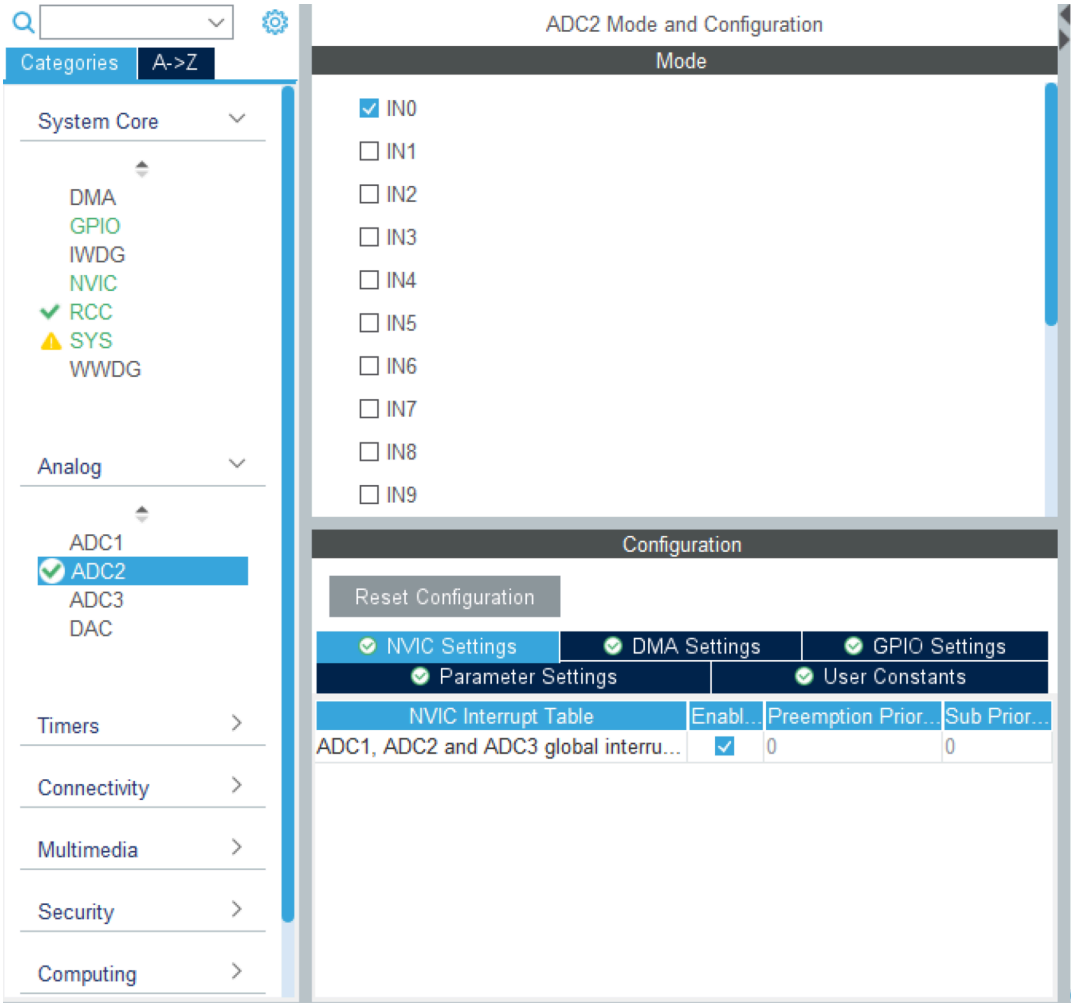

Figure 2: Enable ADC2_IN0 by ticking the box and enabling the global interrupt.

Figure 3: Pinout Configuration

By enabling ADC2_IN0, Port A Pin 0 turns on and we have PF11,PF12,PF13 and PD8 to PD15 to interface with the LCD module.

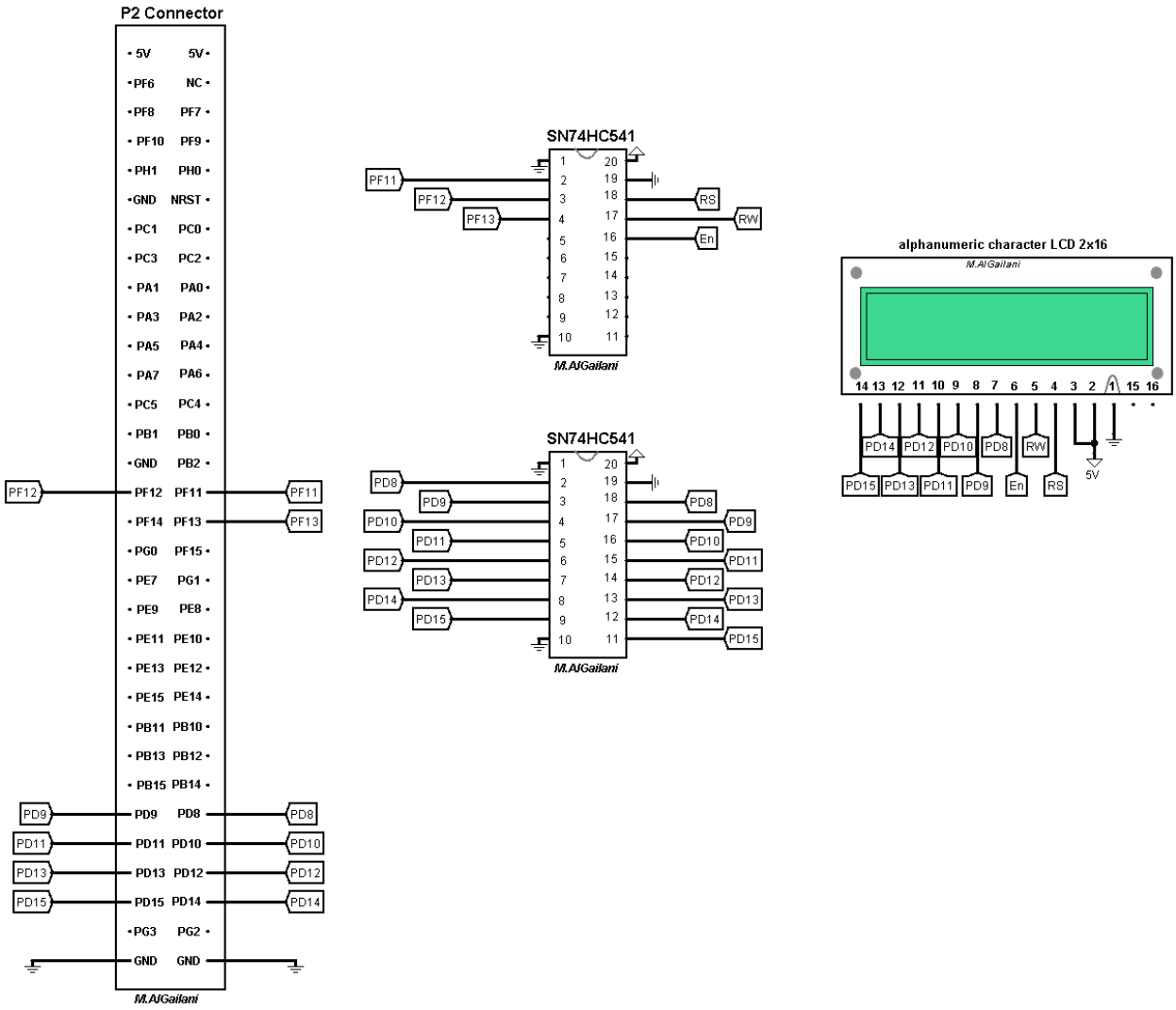

Figure 4: Schematic diagram of the design

Task 2

For this task, we had to develop a C program to display the temperature using an LM35 analog temperature sensor.

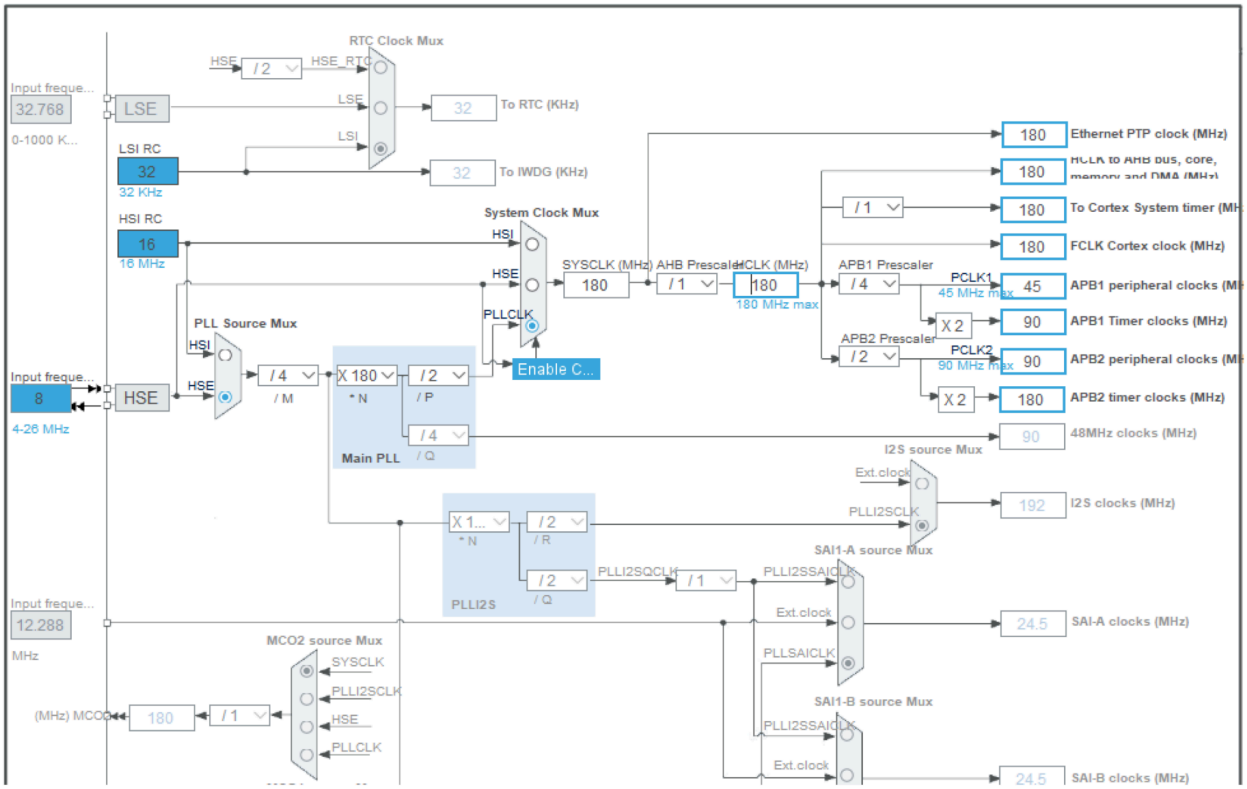

Figure 5: Clock Configuration

The clock configuration will be the same as the previous task.

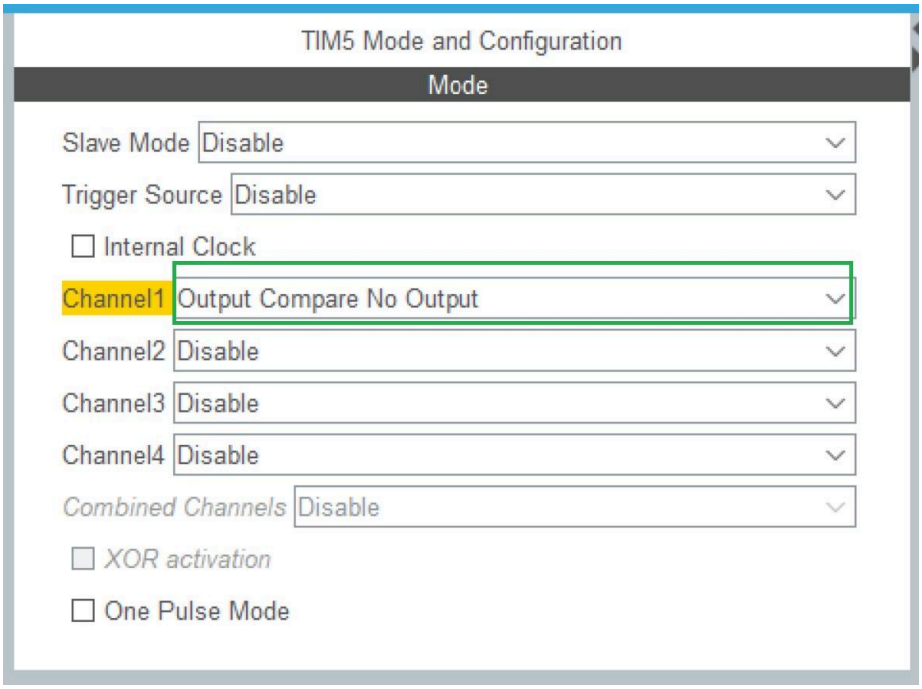

Figure 6: Enabling TIM5 Channel 1 as Output Compare No Output

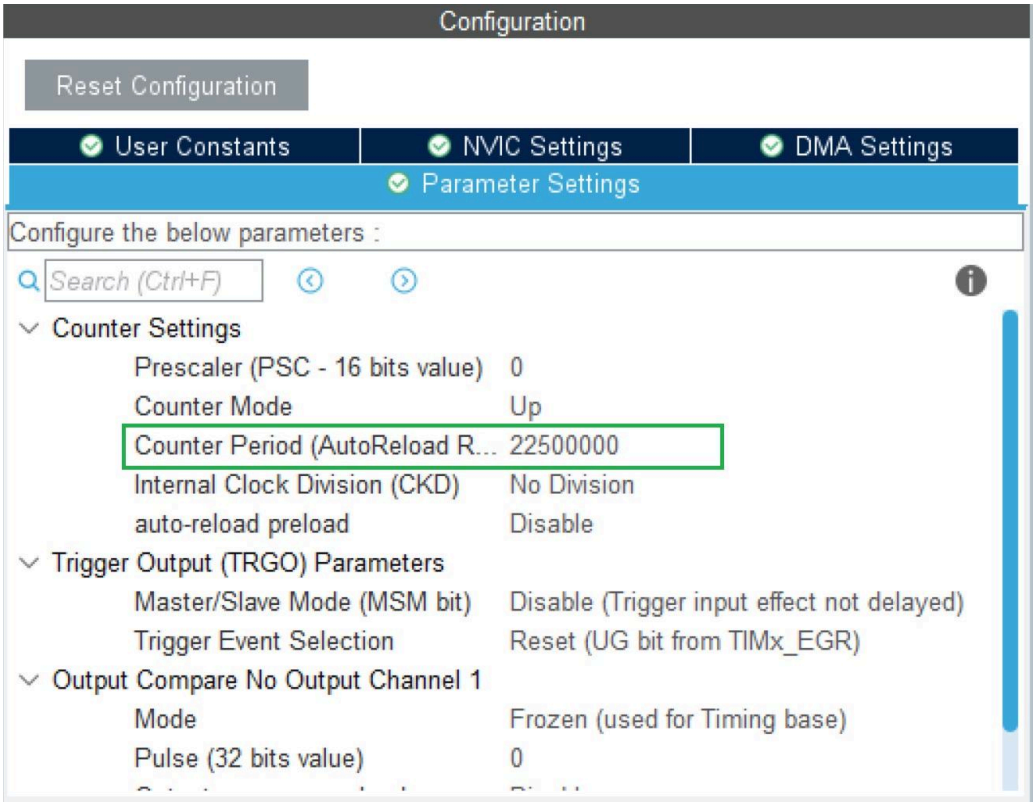

Figure 7: Changing counter period for TIM5 to occur every 250ms.

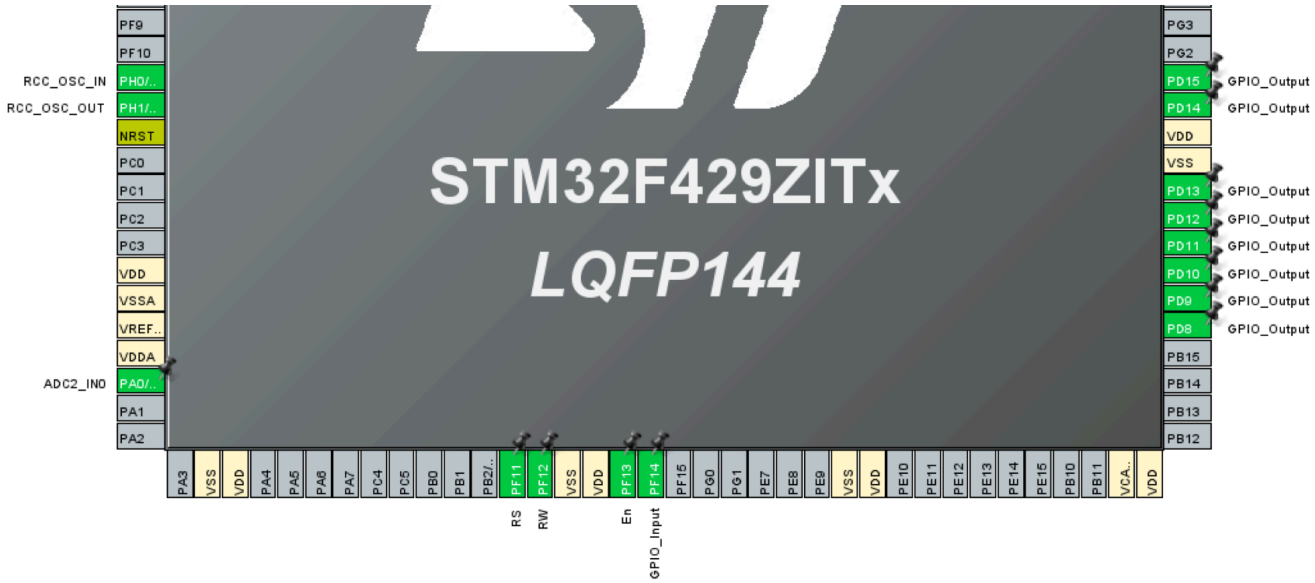

Figure 8: Pinout Configuration with extra input signal for the DIP-switch from PF14

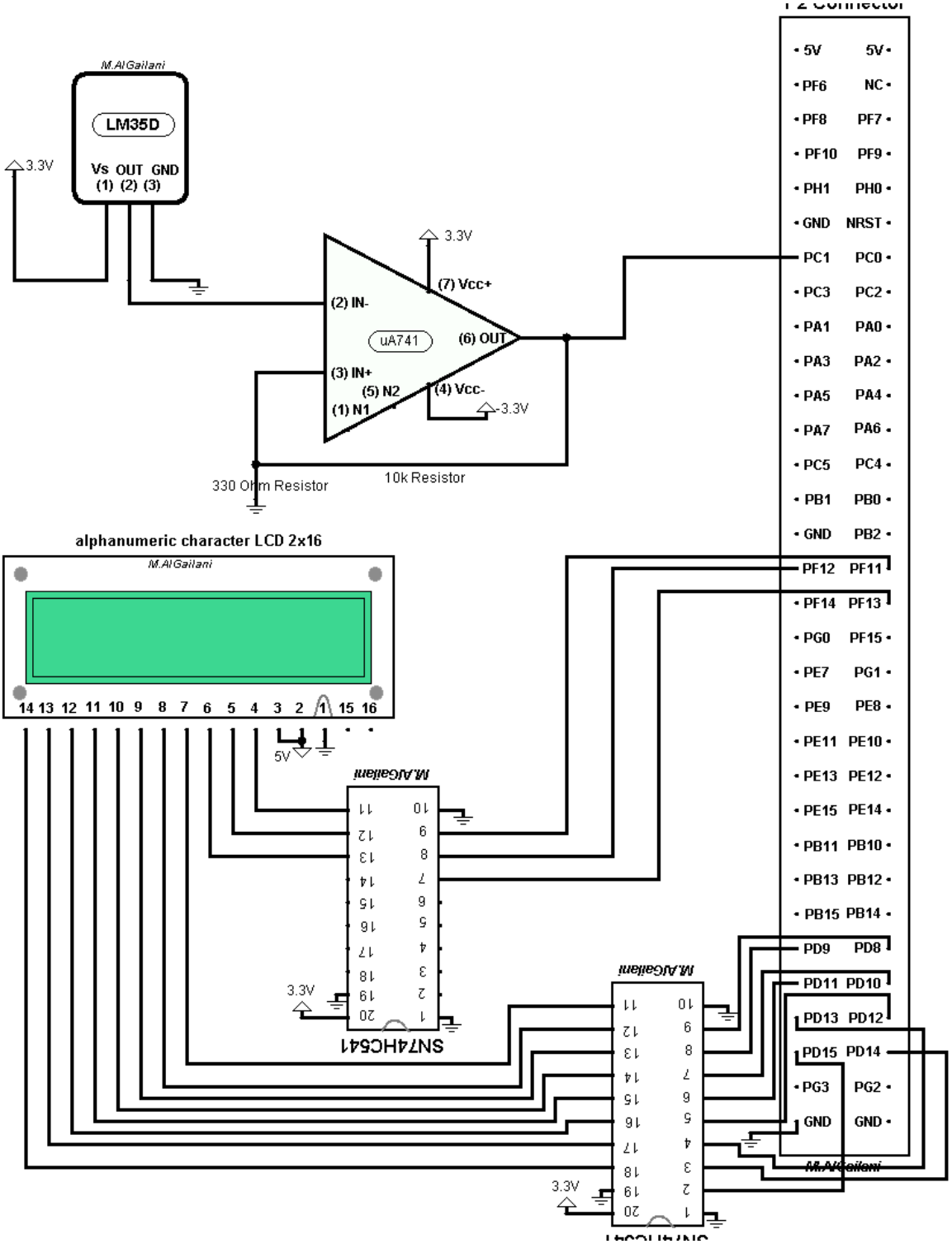

Figure 9: Schematic of the design using an op amp for a gain of 30

Task 3

For this task, we had to develop a C program to display the temperature either in Celsius or Fahrenheit using the digital temperature sensor.

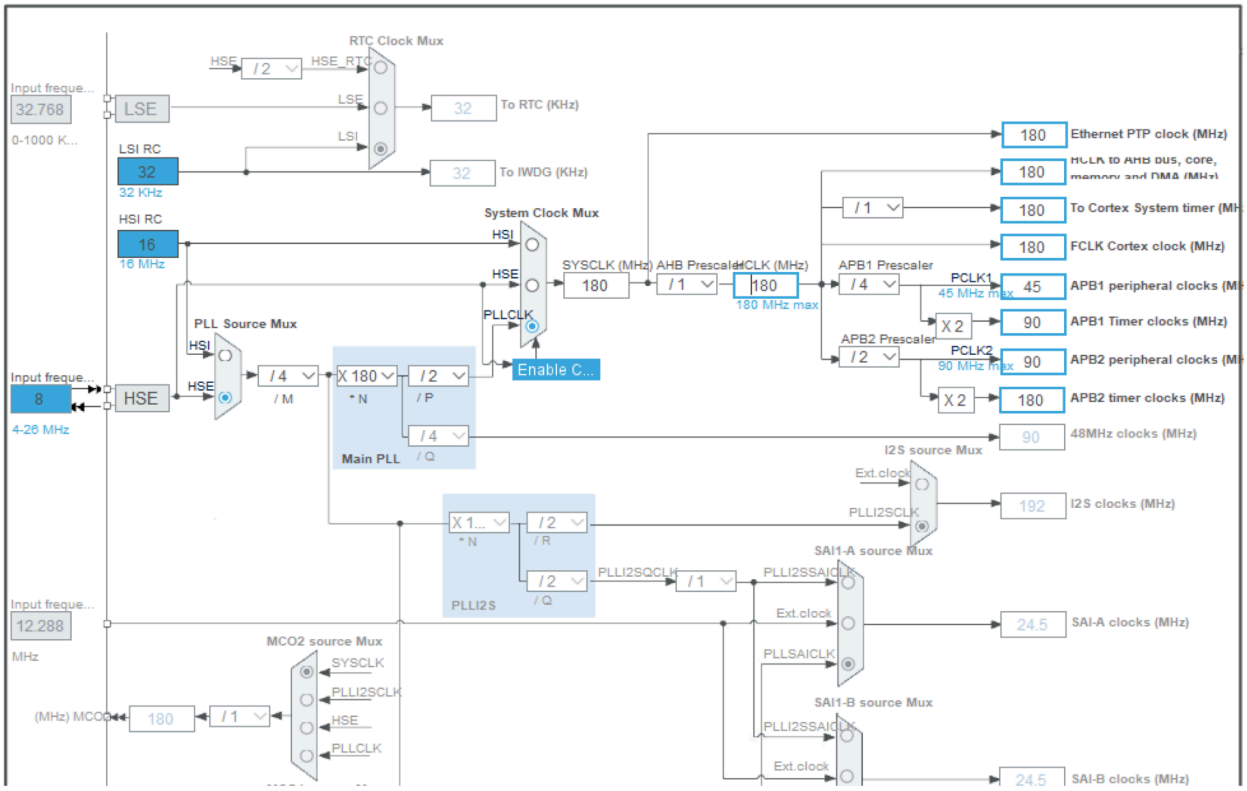

Figure 10: Clock Configuration

The clock configuration remains the same while we add a couple of pins for our new design.

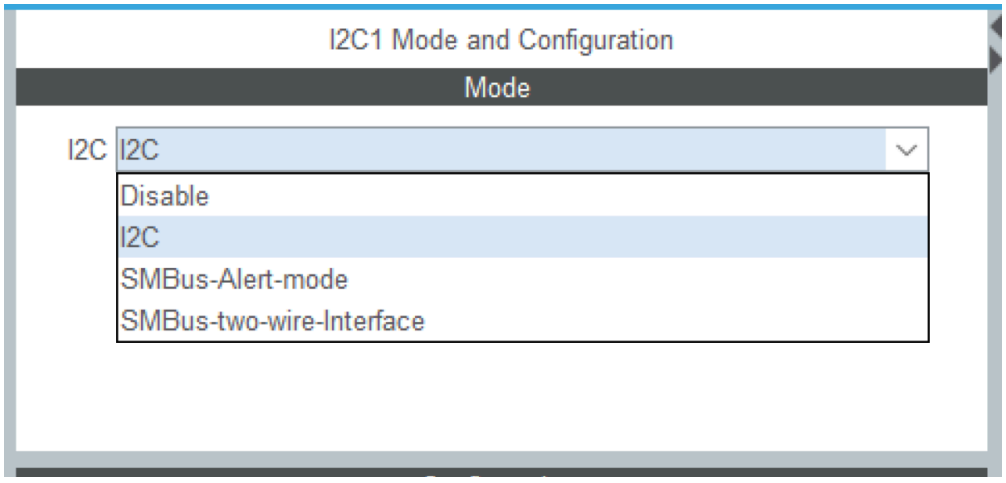

Figure 11: Enabling I2C mode for the digital sensor

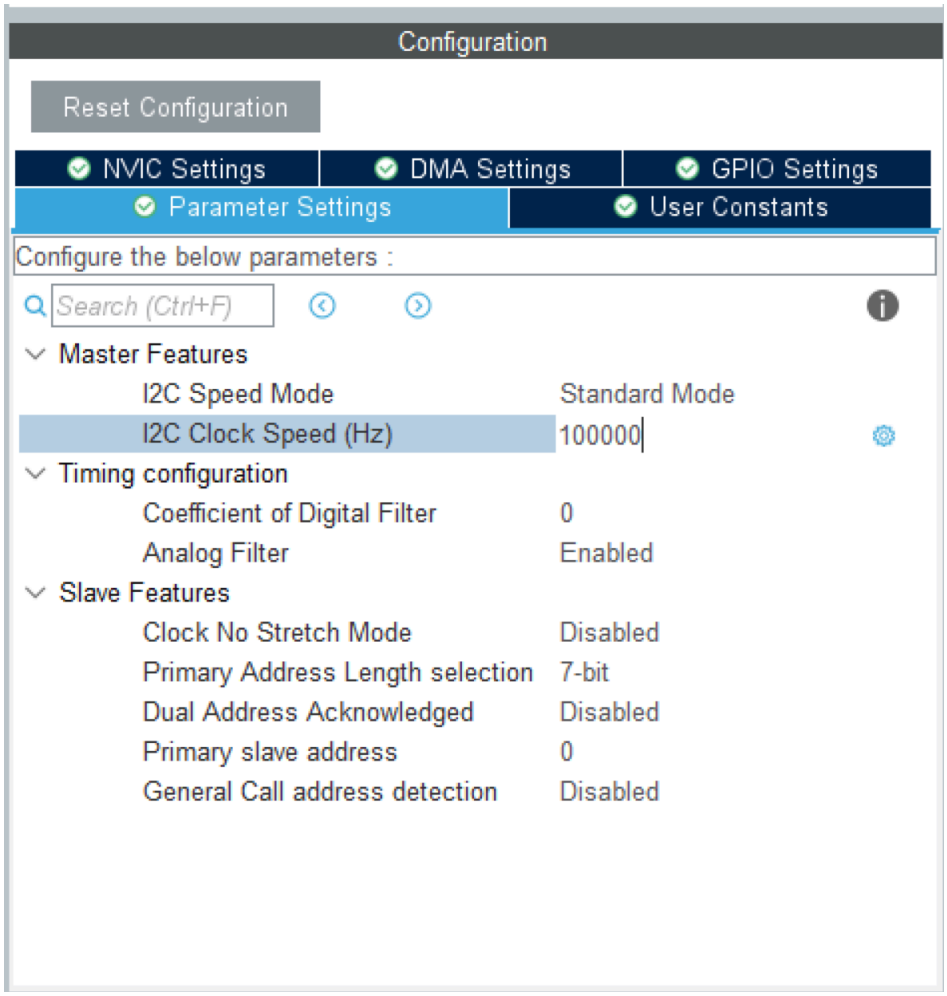

Figure 12: Modifying the settings to match the specifications in the prelab.

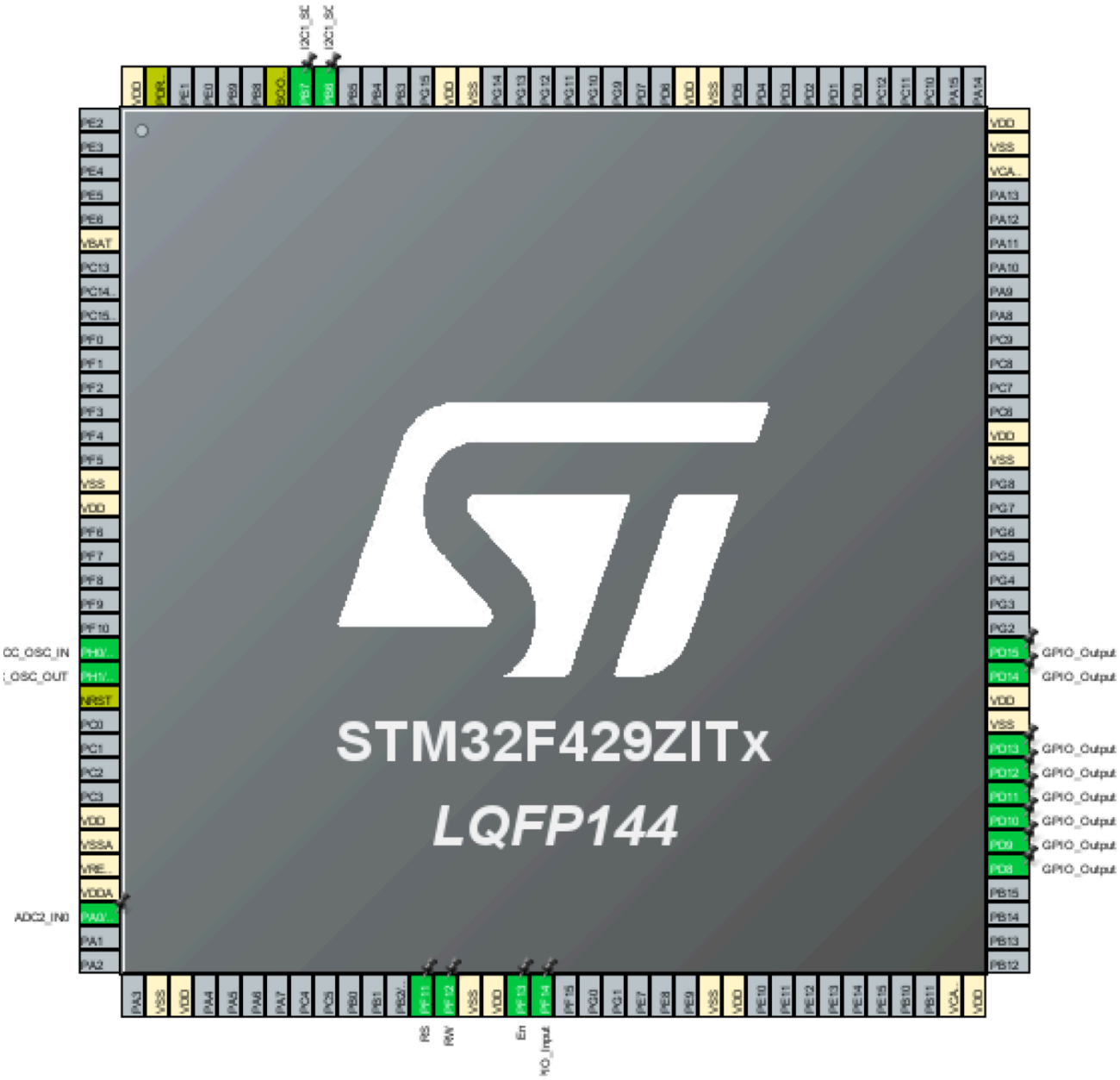

Figure 13: Pinout Configuration for the design

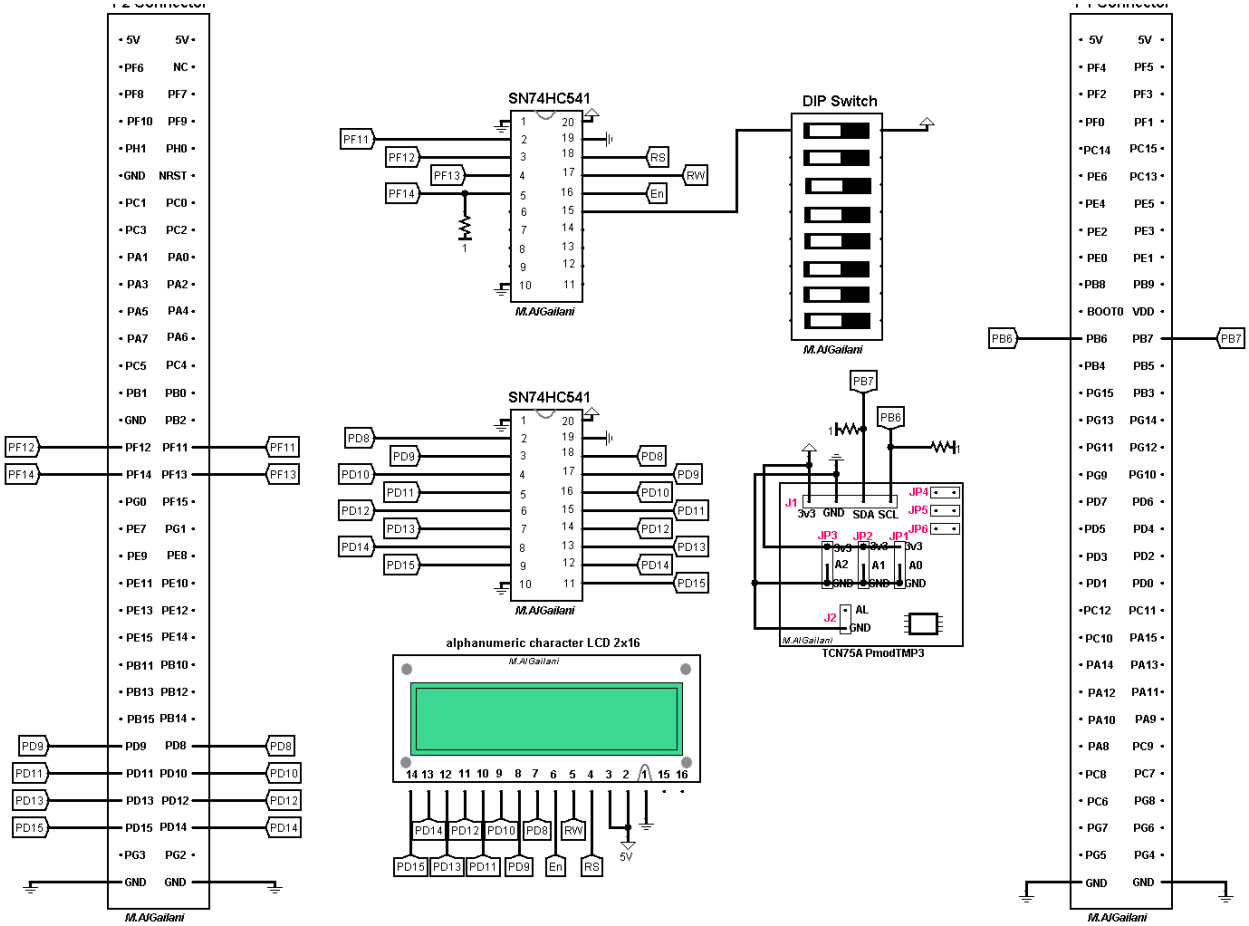

Figure 14: Schematic for the design.

Results

Task 1

Task 1 asked us to calibrate the LCD screen by counting up to 212 and restarting at 0 at a rate of 1Hz.

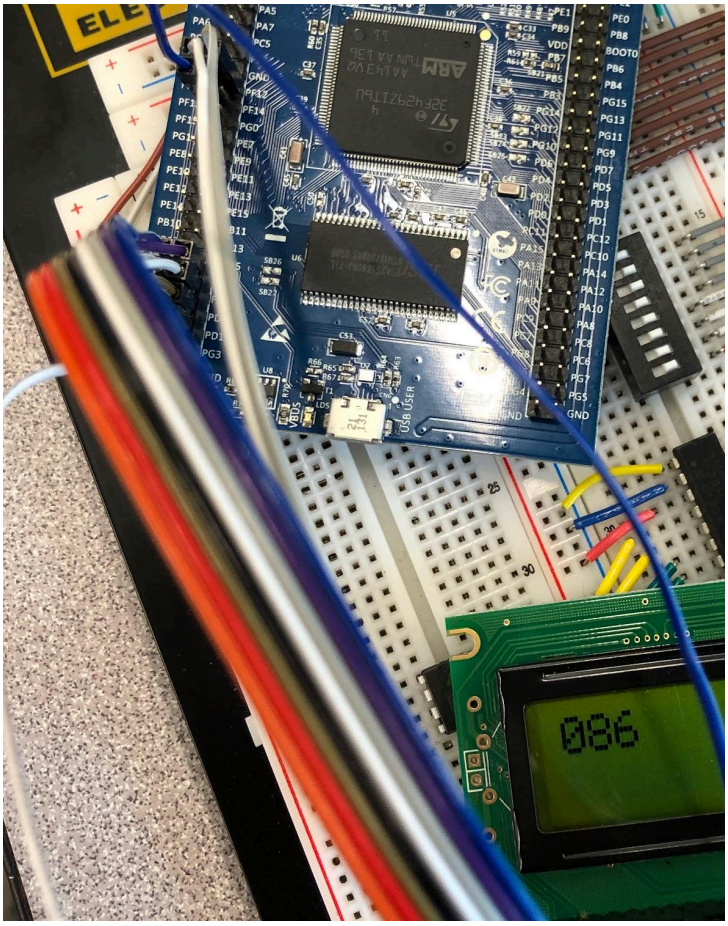

Figure 15: LCD displaying 086

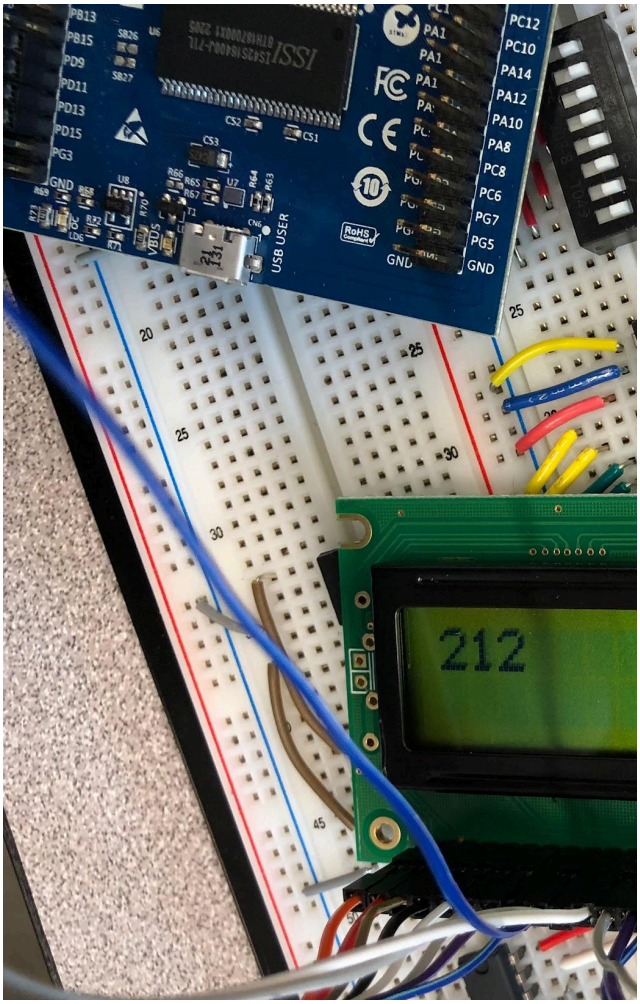

Figure 16: LCD displaying 212

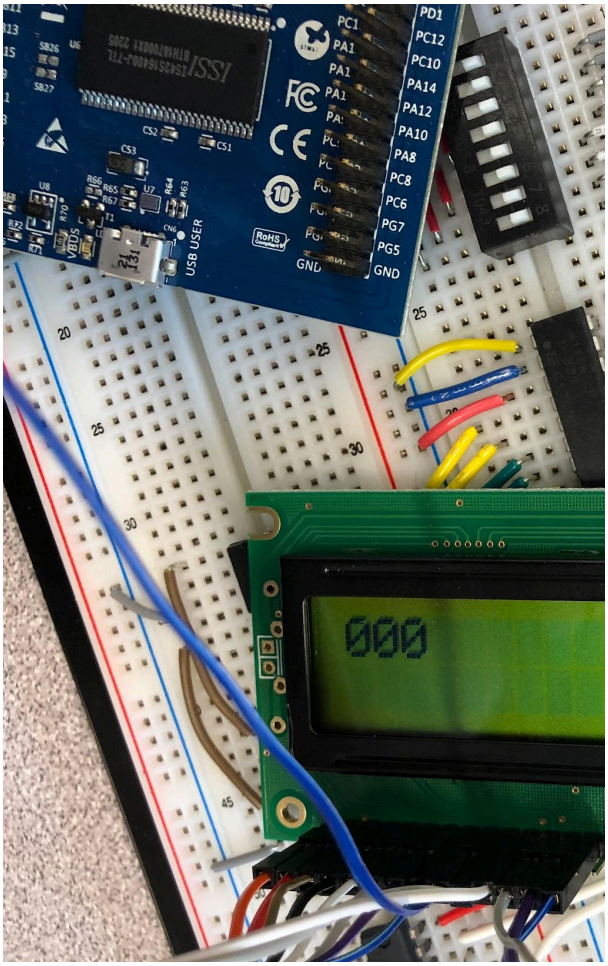

Figure 17: LCD displaying 000 after 212

Task 2

For task 2, we were asked to interface the analog temperature sensor with an ADC channel on our STM32 board to get accurate readings for room temperature.

Expression	Value
num_out	<array>"...
[0]	'0' (0x30)
[1]	'2' (0x32)
[2]	'2' (0x32)
data	22
reading	1'024
<click to add>	

Figure 18 - Our live watch showing a room temperature of 22C. The output to the LCD did not work when the ADC channel was enabled, so the live watch is shown.

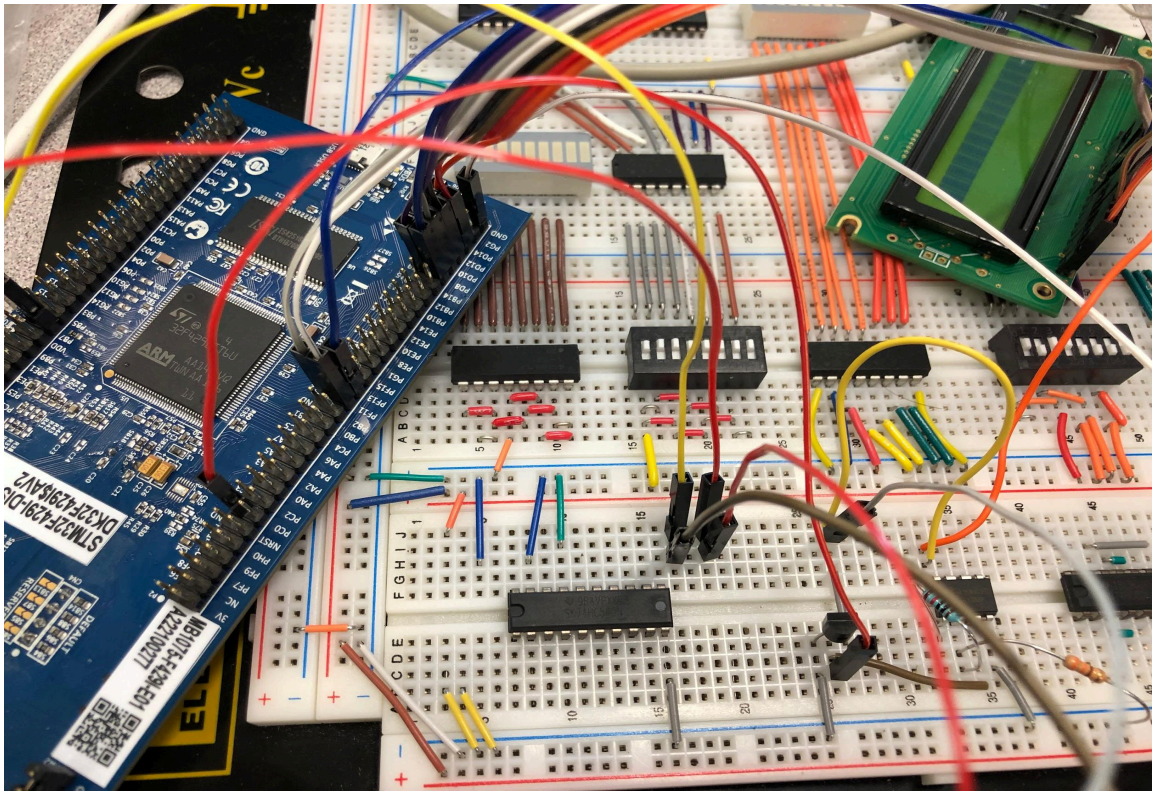

Figure 19: Setup for the Task 2

Task 3

For task 3 we were asked to interface the digital temperature sensor with our STM32 board.

Expression	Value
temperature	'.' (0x17)
val_in	<array>".€"
[0]	'.' (0x17)
[1]	'€' (0x80)
num_out	<array>"...
[0]	'0' (0x30)
[1]	'2' (0x32)
[2]	'3' (0x33)
buffer	0
<click to add>	

Figure 20 - the live watch for the digital sensor data at room temperature in Celsius stored in the num_out variable

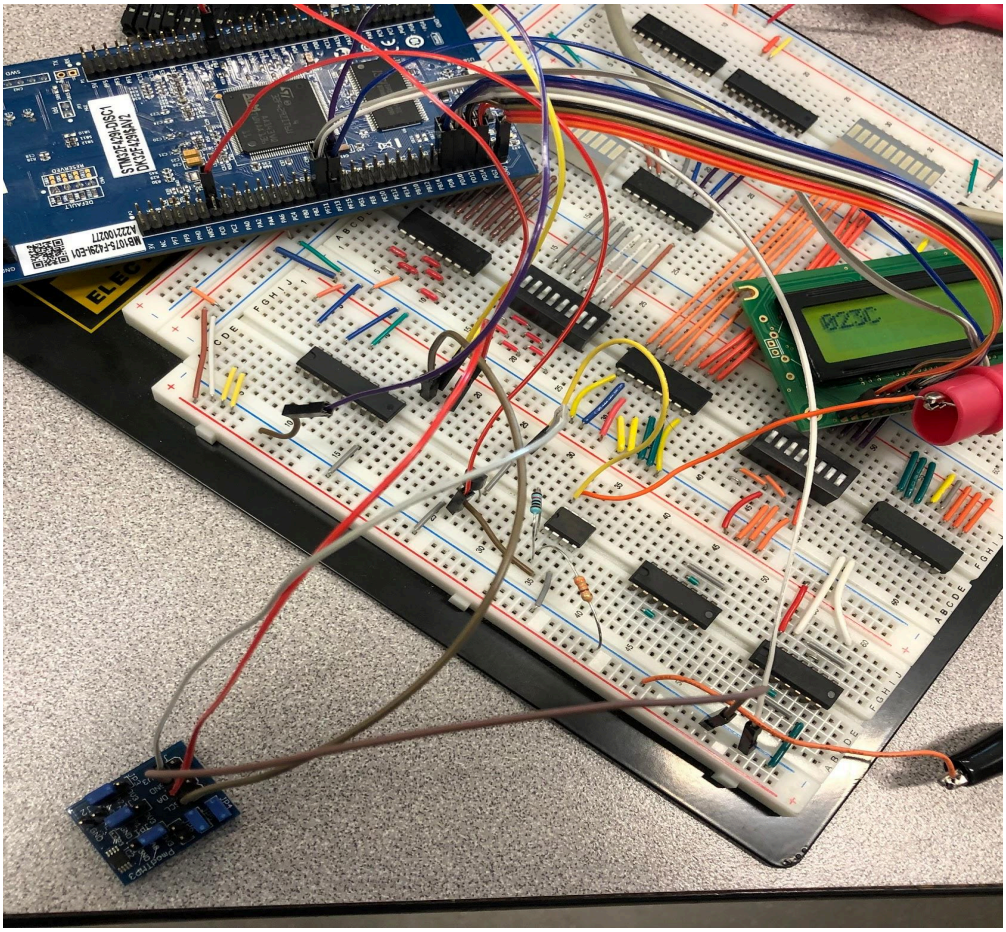

Figure 21: LCD screen showing temperature as 23 C

Expression	Value
temperature	'I' (0x49)
⊟ val_in	<array>"."
[0]	'.' (0x18)
[1]	'\0' (0x00)
⊟ num_out	<array>"...
[0]	'0' (0x30)
[1]	'7' (0x37)
[2]	'3' (0x33)
buffer	0
<click to add>	

Figure 21 - the live watch showing the digital temperature sensor output in fahrenheit stored in the num_out variable

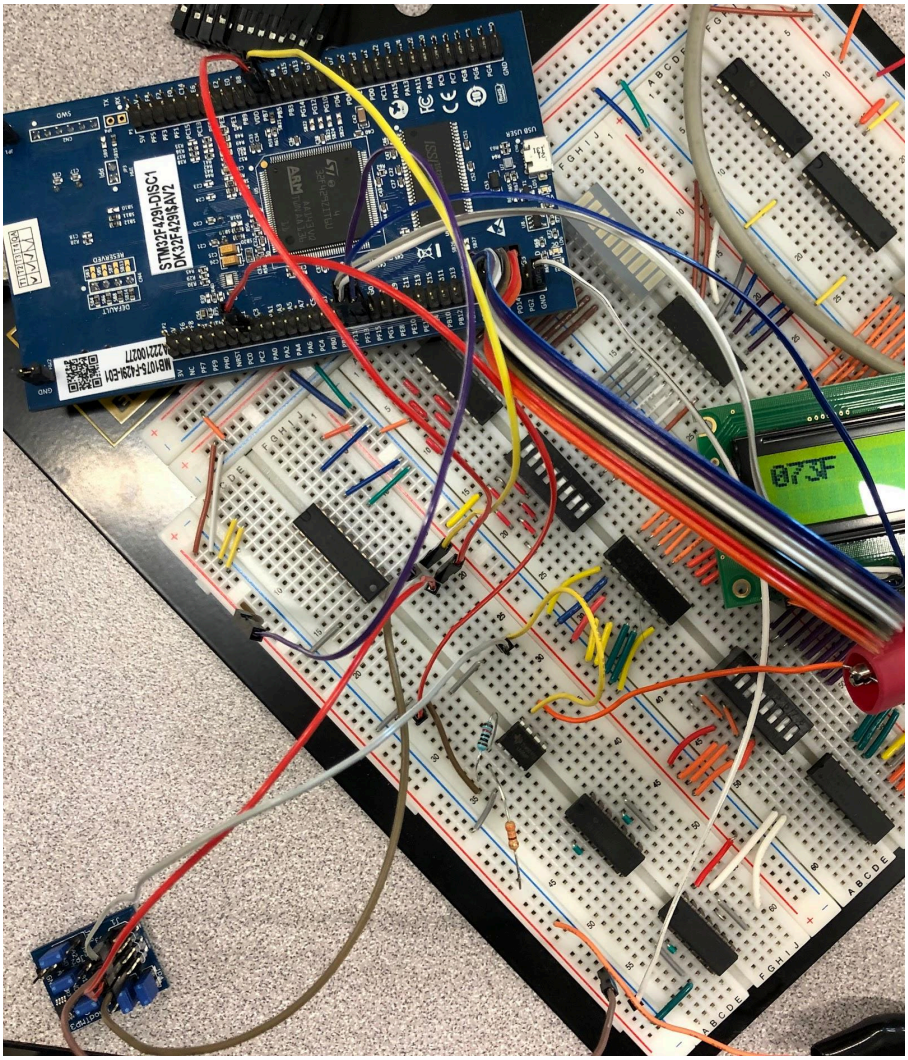

Figure 22: LCD showing the temperature in Fahrenheit.

Conclusion

The parallel communication protocol used for the LCD in this lab was more simple to troubleshoot than the SPI protocol used for other LCD interfacing in other labs, but issues were still encountered. In this lab, the main issue was with the analog temperature sensor not working properly. The sensor did not react to changes in temperature between room temperature and the temperature of a hand being placed on it, so it was only viable to be used at room temperature. The results reflect this use.

The digital temperature sensor was more robust than the analog temperature sensor and was able to be used to get better readings at multiple temperatures. The temperature outputs to the LCD were correct after some digital signal processing using values from the prelab and the digital temperature sensor datasheet. In all, the temperature sensors both yielded results close to the actual room temperature of 22 degrees Celsius.

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 */
void LCD_Init();
void LCD_SendCmd(uint8_t);
void LCD_SendData(uint8_t);
```

```
void int_to_str(uint8_t);
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim5;

/* USER CODE BEGIN PV */
uint8_t out = 0;
uint8_t num_out[3];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM5_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_TIM5_Init();
/* USER CODE BEGIN 2 */
LCD_Init();
//HAL_TIM_OC_Start_IT(&htim5, TIM_CHANNEL_1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{if (out < 213){
        LCD_SendCmd(0x01);
        int_to_str(out);
        for (int i=0; i<3; i++){
        LCD_SendData(num_out[i]);
        }
        LCD_SendData('C');
        out++;
 }
 else{
        out = 0; // reset out once 213 is reached
 }
 HAL_Delay(1000);
        /* 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();
  }
}

/**
  * @brief TIM5 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM5_Init(void)
{

  /* USER CODE BEGIN TIM5_Init 0 */

  /* USER CODE END TIM5_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM5_Init 1 */

  /* USER CODE END TIM5_Init 1 */
  htim5.Instance = TIM5;
```

```
  htim5.Init.Prescaler = 9000;
  htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim5.Init.Period = 10000;
  htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_OC_Init(&htim5) != HAL_OK)
  {
        Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) != HAL_OK)
  {
        Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_TIMING;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_OC_ConfigChannel(&htim5, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
        Error_Handler();
  }
  /* USER CODE BEGIN TIM5_Init 2 */

  /* USER CODE END TIM5_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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,
GPIO_PIN_RESET);

  /*Configure GPIO pins : PF11 PF12 PF13 */
  GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : PD8 PD9 PD10 PD11
                          PD12 PD13 PD14 PD15 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void LCD_Init(){  // from datasheet
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET); // set E = 0
  HAL_Delay(100);  // setup time
  LCD_SendCmd(0x30);
  HAL_Delay(30);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
```

```
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x38);
  LCD_SendCmd(0x80);
  LCD_SendCmd(0x0c);
  LCD_SendCmd(0x06);
}

void LCD_SendCmd(uint8_t cmd){
  GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the correct portion of
the output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET); // set to command
mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void LCD_SendData(uint8_t i){
  GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the correct portion of the
output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); // set to data mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void int_to_str(uint8_t number) {
  int Hex_offset = 0x30;
  int first_dig = (number % 1000)/100;  //separate the hundreds digit
  first_dig = first_dig + Hex_offset;
  num_out[0] = first_dig;
  int second_dig = (number % 100)/10 ; //separate the tens digit
  second_dig = second_dig + Hex_offset;
  num_out[1] = second_dig;
  int third_dig = number % 10;  // separate the ones place
  third_dig = third_dig + Hex_offset;
  num_out[2] = third_dig;
```

```
}
/* 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 */
void LCD_init();
void LCD_SendCmd(uint8_t);
void LCD_SendData(uint8_t);
void int_to_str(uint8_t);
/* USER CODE END PD */

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

/* USER CODE END PM */
```

```
/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc2;

/* USER CODE BEGIN PV */
uint8_t data = 0;
uint16_t reading = 0;
uint8_t num_out[3];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC2_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_ADC2_Init();
  /* USER CODE BEGIN 2 */
// PD8...PD15 is data out to display
// PF11 = RS, PF12 = R/W, PF13 = E for display
// ADC2 channel 11 is the ADC channel used (port PC1)
  LCD_init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
          HAL_ADC_Start(&hadc2);
          reading = HAL_ADC_GetValue(&hadc2);
          data = (reading / 45); // convert from ADC value to temp in C
          int_to_str(data);
          LCD_SendCmd(0x01);
          for (int i=0; i<3; i++){
          LCD_SendData(num_out[i]);
          }
          //LCD_SendData('C');
          HAL_Delay(250); // delay for 250ms
          /* 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();
  }
}

/**
  * @brief ADC2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC2_Init(void)
{

  /* USER CODE BEGIN ADC2_Init 0 */

  /* USER CODE END ADC2_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC2_Init 1 */

  /* USER CODE END ADC2_Init 1 */

  /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and
number of conversion)
  */
  hadc2.Instance = ADC2;
  hadc2.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc2.Init.Resolution = ADC_RESOLUTION_12B;
  hadc2.Init.ScanConvMode = DISABLE;
```

```
  hadc2.Init.ContinuousConvMode = DISABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  hadc2.Init.DMAContinuousRequests = DISABLE;
  hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
          Error_Handler();
  }

  /** Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_11;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
          Error_Handler();
  }
  /* USER CODE BEGIN ADC2_Init 2 */

  /* USER CODE END ADC2_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_GPIOH_CLK_ENABLE();
```

```
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOF, Register_Select_Pin|R_W_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,
GPIO_PIN_RESET);

  /*Configure GPIO pins : Register_Select_Pin R_W_Pin */
  GPIO_InitStruct.Pin = Register_Select_Pin|R_W_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : E_Pin PF14 */
  GPIO_InitStruct.Pin = E_Pin|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : PD8 PD9 PD10 PD11
                          PD12 PD13 PD14 PD15 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void LCD_init(){  // from datasheet
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET); // set E = 0
```

```
  HAL_Delay(100);  // setup time
  LCD_SendCmd(0x30);
  HAL_Delay(30);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x38);
  LCD_SendCmd(0x80);
  LCD_SendCmd(0x0c);
  LCD_SendCmd(0x06);
}

void LCD_SendCmd(uint8_t cmd){
  GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the correct portion of
the output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET); // set to command
mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void LCD_SendData(uint8_t i){
  GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the correct portion of the
output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); // set to data mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void int_to_str(uint8_t number) {
  int Hex_offset = 0x30;
  int first_dig = (number % 1000)/100;  //separate the hundreds digit
  first_dig = first_dig + Hex_offset;
  num_out[0] = first_dig;
  int second_dig = (number % 100)/10 ; //separate the tens digit
```

```
  second_dig = second_dig + Hex_offset;
  num_out[1] = second_dig;
  int third_dig = number % 10;  // separate the ones place
  third_dig = third_dig + Hex_offset;
  num_out[2] = third_dig;
}
/* 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 3

```
/* 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 */
#define CONFIG 0x0001;
#define TEMP_REG 0b00000000;
void LCD_Init();
void LCD_SendCmd(uint8_t);
void LCD_SendData(uint8_t);
```

```
void int_to_str(uint8_t);
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

/* USER CODE BEGIN PV */
uint8_t temperature = 0;
uint16_t buffer = 0;
uint8_t val_in[2];
uint8_t config = CONFIG;
uint8_t temp_reg = TEMP_REG;
int unit = 0;
uint8_t num_out[3];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_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_I2C1_Init();
  /* USER CODE BEGIN 2 */
  // start the data read
  HAL_I2C_Init(&hi2c1);
  HAL_Delay(1); // wait for setup time
  HAL_I2C_Master_Transmit(&hi2c1, 0b1001001 << 1, &config, 1, 1000); // configure
register 1001001 to be used
  HAL_Delay(1); // wait for setup time
  HAL_I2C_Master_Transmit(&hi2c1, 0b1001001 << 1, &temp_reg, 1, 1000); // turn on
the temperature read function
  HAL_I2C_Master_Receive(&hi2c1, 0b1001001 << 1, &val_in[0], 2, 1000); // read the
output temperature, 12 bit resolution requires 2 byte read
  LCD_Init();
  /* USER CODE END 2 */

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

```
  { LCD_SendCmd(0x01);
          unit = HAL_GPIO_ReadPin(GPIOF, GPIO_PIN_14); // read the unit select pin
          buffer = (val_in[1]<<8 && val_in[0]) >> 4;
          temperature = val_in[0];
          if (unit == 0) {
          int_to_str(temperature);
          for (int i=0; i<3; i++){
          LCD_SendData(num_out[i]);
          }
          LCD_SendData('C');}
  else {
          temperature = temperature *(2) +25; // convert to fahrenheit
          int_to_str(temperature);
          for (int i=0; i<3; i++){
          LCD_SendData(num_out[i]);
          }
          LCD_SendData('F');
  }
          HAL_Delay(250); //wait to read the temp again
          HAL_I2C_Master_Receive(&hi2c1, 0b1001001, &val_in[0], 2, 100000); // read
the output temperature, 12 bit resolution requires 2 byte read

          /* 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();
  }
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
        Error_Handler();
  }

  /** Configure Analogue filter
  */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) !=
HAL_OK)
  {
        Error_Handler();
  }
```

```
  /** Configure Digital filter
  */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  {
          Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,
GPIO_PIN_RESET);

  /*Configure GPIO pins : PF11 PF12 PF13 */
  GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : PD8 PD9 PD10 PD11
                           PD12 PD13 PD14 PD15 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                           |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void LCD_Init(){  // from datasheet
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET); // set E = 0
  HAL_Delay(100);  // setup time
  LCD_SendCmd(0x30);
  HAL_Delay(30);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x38);
  LCD_SendCmd(0x10);
  LCD_SendCmd(0x0c);
  LCD_SendCmd(0x06);
}

void LCD_SendCmd(uint8_t cmd){
  GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the correct portion of
the output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET); // set to command
mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
```

```
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void LCD_SendData(uint8_t i){
  GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the correct portion of the
output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); // set to data mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void int_to_str(uint8_t number) {
  int Hex_offset = 0x30;
  int first_dig = (number % 1000)/100;  //separate the hundreds digit
  first_dig = first_dig + Hex_offset;
  num_out[0] = first_dig;
  int second_dig = (number % 100)/10 ; //separate the tens digit
  second_dig = second_dig + Hex_offset;
  num_out[1] = second_dig;
  int third_dig = number % 10;  // separate the ones place
  third_dig = third_dig + Hex_offset;
  num_out[2] = third_dig;
}
/* 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         06/Apr/2023  17:11:08
# Copyright 1999-2022 IAR Systems AB.
#
#       Cpu mode             =  thumb
#       Endian          =  little
#       Source file     =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\Core
\Src\main.c
#       Command line         =
#       -f
```

```
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\Obj\Application\User\Core\main.o.rsp
#
(S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\Cor
e\Src\main.c
#       -D USE_HAL_DRIVER -D STM32F429xx -lC
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\List\Application\User\Core
#       -o
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\Obj\Application\User\Core
#       --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
#       --dlib_config S:\School_Work\arm\inc\c\DLib_Config_Full.h -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM/../Core/Inc\
#       -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM/../Drivers/STM32F4xx_HAL_Driver/Inc\
#       -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\
#       -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\
#       -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM/../Drivers/CMSIS/Include\
#       -Ohz) --dependencies=n
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\Obj\Application\User\Core\main.o.d
#       Locale          =  C
```

```
#       List file         =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\List\Application\User\Core\main.lst
#       Object file     =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\EWA
RM\prelab8_task1\Obj\Application\User\Core\main.o
#       Runtime model:
#       __CPP_Runtime   =  1
#       __SystemLibrary =  DLib
#       __dlib_version  =  6
#       __size_limit   =  32768|ARM.EW.LINKER
#
###############################################################################
#########

S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\prelab8_task1\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      void LCD_Init();
    35      void LCD_SendCmd(uint8_t);
    36      void LCD_SendData(uint8_t);
    37      void int_to_str(uint8_t);
    38      /* USER CODE END PD */
    39
    40      /* Private macro -------------------------------------------------------------*/
    41      /* USER CODE BEGIN PM */
    42
    43      /* USER CODE END PM */
    44
    45      /* Private variables ---------------------------------------------------------*/

\                           In section .bss, align 4
    46      TIM_HandleTypeDef htim5;
    47
    48      /* USER CODE BEGIN PV */
    49      uint8_t out = 0;
    50      uint8_t num_out[3];
\           num_out:
\   0x0                     DS8 4
\           out:
\   0x4                     DS8 1
\   0x5                     DS8 3
\           htim5:
\   0x8                     DS8 72
```

```
        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_TIM5_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   0xB530          PUSH          {R4,R5,LR}
  \     0x2   0xB089          SUB    SP,SP,#+36
        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      ?Subroutine2
\             ??CrossCallReturnLabel_2: (+1)
\       0x10  0x2000        MOVS        R0,#+0
\       0x12  0x9000        STR   R0,[SP, #+0]
\       0x14  0x....        LDR.N       R5,??DataTable4
\       0x16  0x....        LDR.N       R0,??DataTable4_1
\       0x18  0x....        LDR.N       R4,??DataTable4_2
\       0x1A  0x6801        LDR   R1,[R0, #+0]
\       0x1C  0xF041 0x0180     ORR   R1,R1,#0x80
\       0x20  0x6001        STR   R1,[R0, #+0]
\       0x22  0x6802        LDR   R2,[R0, #+0]
\       0x24  0xF002 0x0280     AND   R2,R2,#0x80
\       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 0x0320     ORR   R3,R3,#0x20
\       0x36  0x6003        STR   R3,[R0, #+0]
\       0x38  0x6801        LDR   R1,[R0, #+0]
\       0x3A  0xF001 0x0120     AND   R1,R1,#0x20
\       0x3E  0x9100        STR   R1,[SP, #+0]
\       0x40  0x9900        LDR   R1,[SP, #+0]
\       0x42  0x9200        STR   R2,[SP, #+0]
\       0x44  0xF44F 0x5160     MOV   R1,#+14336
\       0x48  0x6803        LDR   R3,[R0, #+0]
\       0x4A  0xF043 0x0308     ORR   R3,R3,#0x8
\       0x4E  0x6003        STR   R3,[R0, #+0]
\       0x50  0x6800        LDR   R0,[R0, #+0]
```

```
\      0x52   0xF000 0x0008       AND   R0,R0,#0x8
\      0x56   0x9000         STR   R0,[SP, #+0]
\      0x58   0x9800         LDR   R0,[SP, #+0]
\      0x5A   0x4628         MOV   R0,R5
\      0x5C   0x.... 0x....  BL      HAL_GPIO_WritePin
\      0x60   0x2200         MOVS          R2,#+0
\      0x62   0xF44F 0x417F       MOV   R1,#+65280
\      0x66   0x.... 0x....  BL      ?Subroutine3
\             ??CrossCallReturnLabel_7: (+1)
\      0x6A   0xF44F 0x5160       MOV   R1,#+14336
\      0x6E   0x.... 0x....  BL      ?Subroutine1
\             ??CrossCallReturnLabel_0: (+1)
\      0x72   0x4628         MOV   R0,R5
\      0x74   0x.... 0x....  BL      HAL_GPIO_Init
\      0x78   0xF44F 0x417F       MOV   R1,#+65280
\      0x7C   0x.... 0x....  BL      ?Subroutine1
\             ??CrossCallReturnLabel_1: (+1)
\      0x80   0x4620         MOV   R0,R4
\      0x82   0x.... 0x....  BL      HAL_GPIO_Init
       94               MX_TIM5_Init();
\      0x86   0x2208         MOVS          R2,#+8
\      0x88   0x2100         MOVS          R1,#+0
\      0x8A   0x4668         MOV   R0,SP
\      0x8C   0x.... 0x....  BL      memset
\      0x90   0x221C         MOVS          R2,#+28
\      0x92   0x2100         MOVS          R1,#+0
\      0x94   0xA802         ADD   R0,SP,#+8
\      0x96   0x.... 0x....  BL      memset
\      0x9A   0x....         LDR.N         R4,??DataTable4_3
\      0x9C   0x....         LDR.N         R0,??DataTable4_4
\      0x9E   0x60A0                STR   R0,[R4, #+8]
\      0xA0   0xF242 0x3128       MOVW          R1,#+9000
\      0xA4   0x60E1                STR   R1,[R4, #+12]
\      0xA6   0x2000         MOVS          R0,#+0
\      0xA8   0x6120         STR   R0,[R4, #+16]
\      0xAA   0xF242 0x7110       MOVW          R1,#+10000
\      0xAE   0x61A0                STR   R0,[R4, #+24]
\      0xB0   0x6220         STR   R0,[R4, #+32]
\      0xB2   0x6161         STR   R1,[R4, #+20]
\      0xB4   0xF104 0x0008       ADD   R0,R4,#+8
```

```
\     0xB8   0x.... 0x....    BL      HAL_TIM_OC_Init
\     0xBC   0xB108           CBZ.N   R0,??main_0
\     0xBE   0x.... 0x....    BL      Error_Handler
\            ??main_0: (+1)
\     0xC2   0x2100           MOVS    R1,#+0
\     0xC4   0x9100           STR     R1,[SP, #+0]
\     0xC6   0x9101           STR     R1,[SP, #+4]
\     0xC8   0xF104 0x0008    ADD     R0,R4,#+8
\     0xCC   0x4669           MOV     R1,SP
\     0xCE   0x.... 0x....    BL      HAL_TIMEx_MasterConfigSynchronization
\     0xD2   0xB108           CBZ.N   R0,??main_1
\     0xD4   0x.... 0x....    BL      Error_Handler
\            ??main_1: (+1)
\     0xD8   0x2100           MOVS    R1,#+0
\     0xDA   0x9102           STR     R1,[SP, #+8]
\     0xDC   0x2200           MOVS    R2,#+0
\     0xDE   0x9104           STR     R1,[SP, #+16]
\     0xE0   0x9203           STR     R2,[SP, #+12]
\     0xE2   0x9206           STR     R2,[SP, #+24]
\     0xE4   0xA902           ADD     R1,SP,#+8
\     0xE6   0xF104 0x0008    ADD     R0,R4,#+8
\     0xEA   0x.... 0x....    BL      HAL_TIM_OC_ConfigChannel
\     0xEE   0xB108           CBZ.N   R0,??main_2
\     0xF0   0x.... 0x....    BL      Error_Handler
      95            /* USER CODE BEGIN 2 */
      96            LCD_Init();
\            ??main_2: (+1)
\     0xF4   0x.... 0x....    BL      LCD_Init
\     0xF8   0xE001           B.N     ??main_3
      97            //HAL_TIM_OC_Start_IT(&htim5, TIM_CHANNEL_1);
      98            /* USER CODE END 2 */
      99
      100           /* Infinite loop */
      101           /* USER CODE BEGIN WHILE */
      102           while (1)
      103           {if (out < 213){
      104           LCD_SendCmd(0x01);
      105           int_to_str(out);
      106           for (int i=0; i<3; i++){
      107           LCD_SendData(num_out[i]);
```

```
      108             }
      109             LCD_SendData('C');
      110             out++;
      111             }
      112             else{
      113             out = 0; // reset out once 213 is reached
\               ??main_4: (+1)
\     0xFA  0x2000      MOVS      R0,#+0
\     0xFC  0xE014      B.N       ??main_5
      114             }
\               ??main_3: (+1)
\     0xFE  0x7920      LDRB R0,[R4, #+4]
\     0x100  0x28D5     CMP  R0,#+213
\     0x102  0xDAFA     BGE.N     ??main_4
\     0x104  0x2001     MOVS      R0,#+1
\     0x106  0x.... 0x.... BL     LCD_SendCmd
\     0x10A  0x7920     LDRB R0,[R4, #+4]
\     0x10C  0x.... 0x.... BL     int_to_str
\     0x110  0x2500     MOVS      R5,#+0
\               ??main_6: (+1)
\     0x112  0x5D60     LDRB R0,[R4, R5]
\     0x114  0x.... 0x.... BL     LCD_SendData
\     0x118  0x1C6D     ADDS      R5,R5,#+1
\     0x11A  0x2D03     CMP  R5,#+3
\     0x11C  0xDBF9     BLT.N ??main_6
\     0x11E  0x2043     MOVS      R0,#+67
\     0x120  0x.... 0x.... BL     LCD_SendData
\     0x124  0x7920     LDRB R0,[R4, #+4]
\     0x126  0x1C40     ADDS      R0,R0,#+1
      115             HAL_Delay(1000);
\               ??main_5: (+1)
\     0x128  0x7120     STRB R0,[R4, #+4]
\     0x12A  0xF44F 0x707A MOV  R0,#+1000
\     0x12E  0x.... 0x.... BL     HAL_Delay
\     0x132  0xE7E4     B.N       ??main_3
      116             /* USER CODE END WHILE */
      117
      118             /* USER CODE BEGIN 3 */
      119             }
      120             /* USER CODE END 3 */
```

```
        121             }
        122
        123             /**
        124             * @brief System Clock Configuration
        125             * @retval None
        126             */

  \                             In section .text, align 2, keep-with-next
        127             void SystemClock_Config(void)
        128             {
  \             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     ?Subroutine2
        129             RCC_OscInitTypeDef RCC_OscInitStruct = {0};
        130             RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
        131
        132             /** Configure the main internal regulator output voltage
        133             */
        134             __HAL_RCC_PWR_CLK_ENABLE();
  \             ??CrossCallReturnLabel_3: (+1)
  \      0x12  0x2000          MOVS         R0,#+0
  \      0x14  0x9000          STR    R0,[SP, #+0]
        135
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1
);
        136
        137             /** Initializes the RCC Oscillators according to the specified
parameters
        138             * in the RCC_OscInitTypeDef structure.
        139             */
        140             RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSE;
        141             RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  \      0x16  0xF44F 0x3380       MOV  R3,#+65536
  \      0x1A  0x....              LDR.N        R0,??DataTable4_5
```

```
\     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,??DataTable4_6
\     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]
      142             RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
      143             RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
      144             RCC_OscInitStruct.PLL.PLLM = 4;
      145             RCC_OscInitStruct.PLL.PLLN = 180;
      146             RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
      147             RCC_OscInitStruct.PLL.PLLQ = 4;
      148             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
         149            {
         150            Error_Handler();
  \      0x6E   0x.... 0x....     BL      Error_Handler
         151            }
         152
         153            /** Activate the Over-Drive mode
         154            */
         155            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
         156            {
         157            Error_Handler();
  \      0x78   0xB672        CPSID         I
  \              ??SystemClock_Config_2: (+1)
  \      0x7A   0xE7FE            B.N     ??SystemClock_Config_2
         158            }
         159
         160            /** Initializes the CPU, AHB and APB buses clocks
         161            */
         162            RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
         163
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
         164            RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  \              ??SystemClock_Config_1: (+1)
  \      0x7C   0x2102        MOVS          R1,#+2
  \      0x7E   0x9102        STR    R1,[SP, #+8]
         165            RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  \      0x80   0x2200        MOVS          R2,#+0
  \      0x82   0x9203        STR    R2,[SP, #+12]
  \      0x84   0x200F        MOVS          R0,#+15
         166            RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  \      0x86   0xF44F 0x51A0      MOV   R1,#+5120
  \      0x8A   0x9001        STR    R0,[SP, #+4]
  \      0x8C   0x9104        STR    R1,[SP, #+16]
         167            RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
  \      0x8E   0xF44F 0x5280      MOV   R2,#+4096
  \      0x92   0x9205        STR   R2,[SP, #+20]
         168
         169            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
         170            {
         171            Error_Handler();
  \      0x9E   0xB672                CPSID         I
  \             ??SystemClock_Config_4: (+1)
  \      0xA0   0xE7FE                B.N     ??SystemClock_Config_4
         172            }
         173            }
  \             ??SystemClock_Config_3: (+1)
  \      0xA2   0xB013                ADD   SP,SP,#+76
  \      0xA4   0xBD00                POP   {PC}

  \                            In section .text, align 2, keep-with-next
  \             ?Subroutine2: (+1)
  \      0x0    0x2214        MOVS        R2,#+20
  \      0x2    0x2100        MOVS        R1,#+0
  \      0x4    0xA801        ADD   R0,SP,#+4
  \      0x6    0x.... 0x....     B.W     memset
         174
         175            /**
         176            * @brief TIM5 Initialization Function
         177            * @param None
         178            * @retval None
         179            */
         180            static void MX_TIM5_Init(void)
         181            {
         182
         183            /* USER CODE BEGIN TIM5_Init 0 */
         184
         185            /* USER CODE END TIM5_Init 0 */
         186
         187            TIM_MasterConfigTypeDef sMasterConfig = {0};
```

```
188         TIM_OC_InitTypeDef sConfigOC = {0};
189
190         /* USER CODE BEGIN TIM5_Init 1 */
191
192         /* USER CODE END TIM5_Init 1 */
193         htim5.Instance = TIM5;
194         htim5.Init.Prescaler = 9000;
195         htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
196         htim5.Init.Period = 10000;
197         htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
198         htim5.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
199         if (HAL_TIM_OC_Init(&htim5) != HAL_OK)
200         {
201         Error_Handler();
202         }
203         sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
204         sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
205         if (HAL_TIMEx_MasterConfigSynchronization(&htim5,
&sMasterConfig) != HAL_OK)
206         {
207         Error_Handler();
208         }
209         sConfigOC.OCMode = TIM_OCMODE_TIMING;
210         sConfigOC.Pulse = 0;
211         sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
212         sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
213         if (HAL_TIM_OC_ConfigChannel(&htim5, &sConfigOC,
TIM_CHANNEL_1) != HAL_OK)
214         {
215         Error_Handler();
216         }
217         /* USER CODE BEGIN TIM5_Init 2 */
218
219         /* USER CODE END TIM5_Init 2 */
220
221         }
222
223         /**
```

```
224         * @brief GPIO Initialization Function
225         * @param None
226         * @retval None
227         */
228         static void MX_GPIO_Init(void)
229         {
230         GPIO_InitTypeDef GPIO_InitStruct = {0};
231
232         /* GPIO Ports Clock Enable */
233         __HAL_RCC_GPIOH_CLK_ENABLE();
234         __HAL_RCC_GPIOF_CLK_ENABLE();
235         __HAL_RCC_GPIOD_CLK_ENABLE();
236
237         /*Configure GPIO pin Output Level */
238         HAL_GPIO_WritePin(GPIOF,
GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13, GPIO_PIN_RESET);
239
240         /*Configure GPIO pin Output Level */
241         HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
242
|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
243
244         /*Configure GPIO pins : PF11 PF12 PF13 */
245         GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13;
246         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
247         GPIO_InitStruct.Pull = GPIO_NOPULL;
248         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
249         HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
250
251         /*Configure GPIO pins : PD8 PD9 PD10 PD11
252                                 PD12 PD13 PD14 PD15 */
253         GPIO_InitStruct.Pin =
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
254
|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
255         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
256         GPIO_InitStruct.Pull = GPIO_NOPULL;
257         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
258         HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
```

```
        259
        260             }
        261
        262             /* USER CODE BEGIN 4 */

  \                             In section .text, align 2, keep-with-next
        263             void LCD_Init(){  // from datasheet
  \             LCD_Init: (+1)
  \     0x0   0xB580            PUSH            {R7,LR}
        264             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
// set E = 0
  \     0x2   0x2200            MOVS            R2,#+0
  \     0x4   0xF44F 0x5100     MOV             R1,#+8192
  \     0x8   0x....            LDR.N           R0,??DataTable4
  \     0xA   0x.... 0x....     BL              HAL_GPIO_WritePin
        265             HAL_Delay(100);  // setup time
  \     0xE   0x2064            MOVS            R0,#+100
  \     0x10  0x.... 0x....     BL              HAL_Delay
        266             LCD_SendCmd(0x30);
  \     0x14  0x2030            MOVS            R0,#+48
  \     0x16  0x.... 0x....     BL              LCD_SendCmd
        267             HAL_Delay(30);
  \     0x1A  0x201E                    MOVS            R0,#+30
  \     0x1C  0x.... 0x....     BL              HAL_Delay
        268             LCD_SendCmd(0x30);
  \     0x20  0x2030            MOVS            R0,#+48
  \     0x22  0x.... 0x....     BL              LCD_SendCmd
        269             HAL_Delay(10);
  \     0x26  0x200A            MOVS            R0,#+10
  \     0x28  0x.... 0x....     BL              HAL_Delay
        270             LCD_SendCmd(0x30);
  \     0x2C  0x2030            MOVS            R0,#+48
  \     0x2E  0x.... 0x....     BL              LCD_SendCmd
        271             HAL_Delay(10);
  \     0x32  0x200A            MOVS            R0,#+10
  \     0x34  0x.... 0x....     BL              HAL_Delay
        272             LCD_SendCmd(0x38);
  \     0x38  0x2038            MOVS            R0,#+56
  \     0x3A  0x.... 0x....     BL              LCD_SendCmd
        273             LCD_SendCmd(0x80);
```

```
  \      0x3E   0x2080         MOVS     R0,#+128
  \      0x40   0x.... 0x....  BL       LCD_SendCmd
         274              LCD_SendCmd(0x0c);
  \      0x44   0x200C         MOVS     R0,#+12
  \      0x46   0x.... 0x....  BL       LCD_SendCmd
         275              LCD_SendCmd(0x06);
  \      0x4A   0xE8BD 0x4002  POP      {R1,LR}
  \      0x4E   0x2006         MOVS     R0,#+6
  \      0x50                  REQUIRE LCD_SendCmd
  \      0x50                  ;; // Fall through to label LCD_SendCmd
         276              }
         277

  \                            In section .text, align 2, keep-with-next
         278              void LCD_SendCmd(uint8_t cmd){
  \              LCD_SendCmd: (+1)
  \      0x0   0xB510          PUSH     {R4,LR}
         279              GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the
correct portion of the output to write to the data port of LCD
  \      0x2   0x.... 0x....   BL       ?Subroutine4
         280              HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET);
// set to command mode
  \              ??CrossCallReturnLabel_8: (+1)
  \      0x6   0x2200          MOVS     R2,#+0
  \      0x8   0x....          B.N      ?Subroutine0
         281              HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET);
// set to write mode
         282              HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); //
give an E pulse
         283              HAL_Delay(1);
         284              HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
         285              }

  \                            In section .text, align 2, keep-with-next
  \              ?Subroutine0: (+1)
  \      0x0   0xF44F 0x6100   MOV    R1,#+2048
  \      0x4   0x.... 0x....   BL       ?Subroutine3
  \              ??CrossCallReturnLabel_6: (+1)
  \      0x8   0x2200          MOVS     R2,#+0
  \      0xA   0xF44F 0x5180   MOV    R1,#+4096
```

```
\      0xE   0x.... 0x....      BL      ?Subroutine3
\            ??CrossCallReturnLabel_5: (+1)
\      0x12  0x2201       MOVS       R2,#+1
\      0x14  0xF44F 0x5100      MOV   R1,#+8192
\      0x18  0x.... 0x....      BL      ?Subroutine3
\            ??CrossCallReturnLabel_4: (+1)
\      0x1C  0x2001       MOVS       R0,#+1
\      0x1E  0x.... 0x....      BL      HAL_Delay
\      0x22  0x4620       MOV   R0,R4
\      0x24  0xE8BD 0x4010      POP   {R4,LR}
\      0x28  0x2200       MOVS       R2,#+0
\      0x2A  0xF44F 0x5100      MOV   R1,#+8192
\      0x2E  0x.... 0x....      B.W     HAL_GPIO_WritePin

\                                 In section .text, align 2, keep-with-next
\            ?Subroutine4: (+1)
\      0x0   0x....             LDR.N       R3,??DataTable4_7
\      0x2   0x....             LDR.N       R4,??DataTable4
\      0x4   0x0200             LSLS  R0,R0,#+8
\      0x6   0x6018             STR   R0,[R3, #+0]
\      0x8   0x4770             BX      LR

\                                 In section .text, align 2, keep-with-next
\            ?Subroutine3: (+1)
\      0x0   0x4620             MOV   R0,R4
\      0x2   0x.... 0x....      B.W     HAL_GPIO_WritePin
       286

\                                 In section .text, align 2, keep-with-next
       287            void LCD_SendData(uint8_t i){
\            LCD_SendData: (+1)
\      0x0   0xB510             PUSH        {R4,LR}
       288            GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the
correct portion of the output to write to the data port of LCD
\      0x2   0x.... 0x....      BL      ?Subroutine4
       289            HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); //
set to data mode
\            ??CrossCallReturnLabel_9: (+1)
\      0x6   0x2201             MOVS        R2,#+1
\      0x8                      REQUIRE ?Subroutine0
```

```
\      0x8                     ;; // Fall through to label ?Subroutine0
       290             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET);
// set to write mode
       291             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); //
give an E pulse
       292             HAL_Delay(1);
       293             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
       294             }
       295

\                              In section .text, align 2, keep-with-next
       296             void int_to_str(uint8_t number) {
\              int_to_str: (+1)
\      0x0   0xB530            PUSH          {R4,R5,LR}
       297             int Hex_offset = 0x30;
       298             int first_dig = (number % 1000)/100;  //separate the hundreds digit
       299             first_dig = first_dig + Hex_offset;
       300             num_out[0] = first_dig;
\      0x2   0x2264            MOVS          R2,#+100
\      0x4   0xFB90 0xF2F2     SDIV  R2,R0,R2
\      0x8   0x....            LDR.N         R3,??DataTable4_3
\      0xA   0xF102 0x0430     ADD   R4,R2,#+48
\      0xE   0x701C            STRB R4,[R3, #+0]
       301             int second_dig = (number % 100)/10 ; //separate the tens digit
       302             second_dig = second_dig + Hex_offset;
       303             num_out[1] = second_dig;
\      0x10   0x2164           MOVS          R1,#+100
       304             int third_dig = number % 10;  // separate the ones place
       305             third_dig = third_dig + Hex_offset;
       306             num_out[2] = third_dig;
\      0x12   0x240A           MOVS          R4,#+10
\      0x14   0xFB01 0x0212    MLS   R2,R1,R2,R0
\      0x18   0x250A           MOVS          R5,#+10
\      0x1A   0xFB90 0xF4F4    SDIV  R4,R0,R4
\      0x1E   0xFB92 0xF2F5    SDIV  R2,R2,R5
\      0x22   0xFB05 0x0014    MLS   R0,R5,R4,R0
\      0x26   0x3230           ADDS          R2,R2,#+48
\      0x28   0x3030           ADDS          R0,R0,#+48
\      0x2A   0x705A                   STRB R2,[R3, #+1]
\      0x2C   0x7098           STRB R0,[R3, #+2]
```

```
        307             }
  \     0x2E   0xBD30           POP   {R4,R5,PC}
        308             /* USER CODE END 4 */
        309
        310             /**
        311             * @brief  This function is executed in case of error occurrence.
        312             * @retval None
        313             */

  \                             In section .text, align 2, keep-with-next
        314             void Error_Handler(void)
        315             {
        316             /* USER CODE BEGIN Error_Handler_Debug */
        317             /* User can add his own implementation to report the HAL error
return state */
        318               __disable_irq();
  \               Error_Handler: (+1)
  \     0x0   0xB672          CPSID         I
        319               while (1)
  \               ??Error_Handler_0: (+1)
  \     0x2   0xE7FE          B.N     ??Error_Handler_0
        320             {
        321             }
        322             /* USER CODE END Error_Handler_Debug */
        323             }

  \                             In section .text, align 2, keep-with-next
  \               ?Subroutine1: (+1)
  \     0x0   0x9101          STR   R1,[SP, #+4]
  \     0x2   0x2201          MOVS          R2,#+1
  \     0x4   0x2100          MOVS          R1,#+0
  \     0x6   0x9103          STR   R1,[SP, #+12]
  \     0x8   0x9104          STR   R1,[SP, #+16]
  \     0xA   0x9202          STR   R2,[SP, #+8]
  \     0xC   0xA901          ADD   R1,SP,#+4
  \     0xE   0x4770          BX      LR

  \                             In section .text, align 4, keep-with-next
  \               ??DataTable4:
  \     0x0   0x4002'1400             DC32  0x40021400
```

```
\                               In section .text, align 4, keep-with-next
\                ??DataTable4_1:
\        0x0   0x4002'3830              DC32  0x40023830

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_2:
\        0x0   0x4002'0C00              DC32  0x40020c00

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_3:
\        0x0   0x....'....              DC32  num_out

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_4:
\        0x0   0x4000'0C00              DC32  0x40000c00

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_5:
\        0x0   0x4002'3840              DC32  0x40023840

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_6:
\        0x0   0x4000'7000              DC32  0x40007000

\                               In section .text, align 4, keep-with-next
\                ??DataTable4_7:
\        0x0   0x4002'0C14              DC32  0x40020c14
        324
        325             #ifdef  USE_FULL_ASSERT
        326             /**
        327             * @brief  Reports the name of the source file and the source line
number
        328             *         where the assert_param error has occurred.
        329             * @param  file: pointer to the source file name
        330             * @param  line: assert_param error line source number
        331             * @retval None
        332             */
        333             void assert_failed(uint8_t *file, uint32_t line)
        334             {
```

```
        335             /* USER CODE BEGIN 6 */
        336             /* User can add his own implementation to report the file name and
line number,
        337             ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
*/
        338             /* USER CODE END 6 */
        339             }
        340             #endif /* USE_FULL_ASSERT */

  Maximum stack usage in bytes:

  .cstack Function
  ------- --------
        0   Error_Handler
        8   LCD_Init
        8   -> HAL_Delay
        8   -> HAL_GPIO_WritePin
        0   -> LCD_SendCmd
        8   -> LCD_SendCmd
        8   LCD_SendCmd
        8   -> HAL_Delay
        0   -> HAL_GPIO_WritePin
        8   -> HAL_GPIO_WritePin
        8   LCD_SendData
        8   -> HAL_Delay
        0   -> HAL_GPIO_WritePin
        8   -> HAL_GPIO_WritePin
       80   SystemClock_Config
       80   -> Error_Handler
       80   -> HAL_PWREx_EnableOverDrive
       80   -> HAL_RCC_ClockConfig
       80   -> HAL_RCC_OscConfig
       80   -> memset
       12   int_to_str
       48   main
       48   -> Error_Handler
       48   -> HAL_Delay
       48   -> HAL_GPIO_Init
       48   -> HAL_GPIO_WritePin
       48   -> HAL_Init
```

```
    48   -> HAL_TIMEx_MasterConfigSynchronization
    48   -> HAL_TIM_OC_ConfigChannel
    48   -> HAL_TIM_OC_Init
    48   -> LCD_Init
    48   -> LCD_SendCmd
    48   -> LCD_SendData
    48   -> SystemClock_Config
    48   -> int_to_str
    48   -> memset


Section sizes:

Bytes  Function/Label
-----  --------------
       4  ??DataTable4
       4  ??DataTable4_1
       4  ??DataTable4_2
       4  ??DataTable4_3
       4  ??DataTable4_4
       4  ??DataTable4_5
       4  ??DataTable4_6
       4  ??DataTable4_7
      50  ?Subroutine0
      16  ?Subroutine1
      10  ?Subroutine2
       6  ?Subroutine3
      10  ?Subroutine4
       4  Error_Handler
      80  LCD_Init
      10  LCD_SendCmd
       8  LCD_SendData
     166  SystemClock_Config
      48  int_to_str
     308  main
      80  num_out
          out
          htim5
```

```
  80 bytes in section .bss
 748 bytes in section .text

 748 bytes of CODE memory
  80 bytes of DATA memory

Errors: none
Warnings: none
```

Task 2

```
################################################################################
#
# IAR ANSI C/C++ Compiler V9.20.4.327/W64 for ARM         06/Apr/2023  17:13:22
# Copyright 1999-2022 IAR Systems AB.
#
#       Cpu mode             =  thumb
#       Endian          =  little
#       Source file     =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\Co
re\Src\main.c
#       Command line         =
#       -f
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\Obj\Application\User\Core\main.o.rsp
#
(S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\C
ore\Src\main.c
#       -D USE_HAL_DRIVER -D STM32F429xx -lC
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\List\Application\User\Core
#       -o
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\Obj\Application\User\Core
#       --debug --endian=little --cpu=Cortex-M4 -e --fpu=VFPv4_sp
```

```
#        --dlib_config S:\School_Work\arm\inc\c\DLib_Config_Full.h -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM/../Core/Inc\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM/../Drivers/STM32F4xx_HAL_Driver/Inc\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM/../Drivers/CMSIS/Include\
#        -Ohz) --dependencies=n
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\Obj\Application\User\Core\main.o.d
#        Locale         =  C
#        List file      =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\List\Application\User\Core\main.lst
#        Object file    =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\E
WARM\Prelab_8_task3\Obj\Application\User\Core\main.o
#        Runtime model:
#        __CPP_Runtime   =  1
#        __SystemLibrary =  DLib
#        __dlib_version  =  6
#        __size_limit    =  32768|ARM.EW.LINKER
#
```

##
#########

S:\School_Work\Spring_2023\MicroconrollerApps\Prelabs\Prelab_8\Prelab_8_task3\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      void LCD_init();
      35      void LCD_SendCmd(uint8_t);
      36      void LCD_SendData(uint8_t);
      37      void int_to_str(uint8_t);
      38      /* USER CODE END PD */
      39
      40      /* Private macro -------------------------------------------------------*/
      41      /* USER CODE BEGIN PM */
      42
      43      /* USER CODE END PM */
      44
      45      /* Private variables ---------------------------------------------------*/

\                             In section .bss, align 4
      46      ADC_HandleTypeDef hadc2;
\             hadc2:
\     0x0                     DS8 72
      47
      48      /* USER CODE BEGIN PV */
      49      uint8_t data = 0;
\             `data`:
\     0x48                    DS8 1
\     0x49                    DS8 1
      50      uint16_t reading = 0;
\             reading:
\     0x4A                    DS8 2

\                             In section .bss, align 4
      51      uint8_t num_out[3];
\             num_out:
\     0x0                     DS8 4
      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_ADC2_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   0xE92D 0x41F0       PUSH          {R4-R8,LR}
  \      0x4   0xB086              SUB   SP,SP,#+24
        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();
  \      0x6   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();
  \      0xA   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();
\       0xE   0x.... 0x....     BL      ?Subroutine2
\             ??CrossCallReturnLabel_2: (+1)
\       0x12  0x2000        MOVS        R0,#+0
\       0x14  0x9000        STR   R0,[SP, #+0]
\       0x16  0x....        LDR.N       R4,??DataTable4
\       0x18  0x....        LDR.N       R0,??DataTable4_1
\       0x1A  0x....        LDR.N       R5,??DataTable4_2
\       0x1C  0x6801        LDR   R1,[R0, #+0]
\       0x1E  0xF041 0x0180     ORR   R1,R1,#0x80
\       0x22  0x6001        STR   R1,[R0, #+0]
\       0x24  0x6802        LDR   R2,[R0, #+0]
\       0x26  0xF002 0x0280     AND   R2,R2,#0x80
\       0x2A  0x9200        STR   R2,[SP, #+0]
\       0x2C  0x2200        MOVS        R2,#+0
\       0x2E  0x9900        LDR   R1,[SP, #+0]
\       0x30  0x9200        STR   R2,[SP, #+0]
\       0x32  0x6803        LDR   R3,[R0, #+0]
\       0x34  0xF043 0x0304     ORR   R3,R3,#0x4
\       0x38  0x6003        STR   R3,[R0, #+0]
\       0x3A  0x6801        LDR   R1,[R0, #+0]
\       0x3C  0xF001 0x0104     AND   R1,R1,#0x4
\       0x40  0x9100        STR   R1,[SP, #+0]
\       0x42  0x9900        LDR   R1,[SP, #+0]
\       0x44  0x9200        STR   R2,[SP, #+0]
\       0x46  0x6803        LDR   R3,[R0, #+0]
\       0x48  0xF043 0x0320     ORR   R3,R3,#0x20
\       0x4C  0x6003        STR   R3,[R0, #+0]
\       0x4E  0x6801        LDR   R1,[R0, #+0]
\       0x50  0xF001 0x0120     AND   R1,R1,#0x20
\       0x54  0x9100        STR   R1,[SP, #+0]
\       0x56  0x9900        LDR   R1,[SP, #+0]
\       0x58  0x9200        STR   R2,[SP, #+0]
\       0x5A  0xF44F 0x51C0     MOV   R1,#+6144
\       0x5E  0x6803        LDR   R3,[R0, #+0]
\       0x60  0xF043 0x0308     ORR   R3,R3,#0x8
\       0x64  0x6003        STR   R3,[R0, #+0]
\       0x66  0x6800        LDR   R0,[R0, #+0]
\       0x68  0xF000 0x0008     AND   R0,R0,#0x8
```

```
\      0x6C   0x9000         STR    R0,[SP, #+0]
\      0x6E   0x9800         LDR    R0,[SP, #+0]
\      0x70   0x.... 0x....  BL     ?Subroutine3
\             ??CrossCallReturnLabel_7: (+1)
\      0x74   0x2200         MOVS         R2,#+0
\      0x76   0xF44F 0x417F  MOV    R1,#+65280
\      0x7A   0x4628         MOV    R0,R5
\      0x7C   0x.... 0x....  BL     HAL_GPIO_WritePin
\      0x80   0xF44F 0x51C0  MOV    R1,#+6144
\      0x84   0x.... 0x....  BL     ?Subroutine1
\             ??CrossCallReturnLabel_0: (+1)
\      0x88   0x4620         MOV    R0,R4
\      0x8A   0x.... 0x....  BL     HAL_GPIO_Init
\      0x8E   0xF44F 0x40C0  MOV    R0,#+24576
\      0x92   0x2100         MOVS         R1,#+0
\      0x94   0x9001         STR    R0,[SP, #+4]
\      0x96   0x9102         STR    R1,[SP, #+8]
\      0x98   0x9103         STR    R1,[SP, #+12]
\      0x9A   0x4620         MOV    R0,R4
\      0x9C   0xA901                ADD    R1,SP,#+4
\      0x9E   0x.... 0x....  BL     HAL_GPIO_Init
\      0xA2   0xF44F 0x417F  MOV    R1,#+65280
\      0xA6   0x.... 0x....  BL     ?Subroutine1
\             ??CrossCallReturnLabel_1: (+1)
\      0xAA   0x4628                MOV    R0,R5
\      0xAC   0x.... 0x....  BL     HAL_GPIO_Init
       95             MX_ADC2_Init();
\      0xB0   0x2210         MOVS         R2,#+16
\      0xB2   0x2100         MOVS         R1,#+0
\      0xB4   0x4668         MOV    R0,SP
\      0xB6   0x.... 0x....  BL     memset
\      0xBA   0x....         LDR.N        R4,??DataTable4_3
\      0xBC   0x....         LDR.N        R0,??DataTable4_4
\      0xBE   0x6020                STR    R0,[R4, #+0]
\      0xC0   0xF104 0x0530  ADD    R5,R4,#+48
\      0xC4   0x....         LDR.N        R0,??DataTable4_5
\      0xC6   0x62A0                STR    R0,[R4, #+40]
\      0xC8   0xF44F 0x3180  MOV    R1,#+65536
\      0xCC   0x2001                MOVS         R0,#+1
\      0xCE   0x2200                MOVS         R2,#+0
```

```
\     0xD0   0x61E0              STR    R0,[R4, #+28]
\     0xD2   0x6160        STR   R0,[R4, #+20]
\     0xD4   0x6061        STR   R1,[R4, #+4]
\     0xD6   0x60A2              STR    R2,[R4, #+8]
\     0xD8   0x6122        STR   R2,[R4, #+16]
\     0xDA   0x7622              STRB  R2,[R4, #+24]
\     0xDC   0xF884 0x2020       STRB  R2,[R4, #+32]
\     0xE0   0x62E2              STR    R2,[R4, #+44]
\     0xE2   0x60E2              STR    R2,[R4, #+12]
\     0xE4   0x702A              STRB  R2,[R5, #+0]
\     0xE6   0x4620        MOV   R0,R4
\     0xE8   0x.... 0x....   BL      HAL_ADC_Init
\     0xEC   0xB108              CBZ.N         R0,??main_0
\     0xEE   0x.... 0x....   BL      Error_Handler
\            ??main_0: (+1)
\     0xF2   0x200B        MOVS          R0,#+11
\     0xF4   0x2101        MOVS          R1,#+1
\     0xF6   0x9000        STR   R0,[SP, #+0]
\     0xF8   0x9101        STR   R1,[SP, #+4]
\     0xFA   0x2200        MOVS          R2,#+0
\     0xFC   0x9202              STR    R2,[SP, #+8]
\     0xFE   0x4669        MOV   R1,SP
\     0x100   0x4620             MOV   R0,R4
\     0x102   0x.... 0x....  BL      HAL_ADC_ConfigChannel
\     0x106   0xB108             CBZ.N         R0,??main_1
\     0x108   0x.... 0x....  BL      Error_Handler
      96             /* USER CODE BEGIN 2 */
      97       // PD8...PD15 is data out to display
      98       // PF11 = RS, PF12 = R/W, PF13 = E for display
      99       // ADC2 channel 11 is the ADC channel used (port PC1)
      100            LCD_init();
\            ??main_1: (+1)
\     0x10C   0x.... 0x....  BL      LCD_init
\     0x110   0x.... 0x....  LDR.W         R8,??DataTable4_6
      101            /* USER CODE END 2 */
      102
      103            /* Infinite loop */
      104            /* USER CODE BEGIN WHILE */
      105            while (1)
      106            {
```

```
      107             HAL_ADC_Start(&hadc2);
\             ??main_2: (+1)
\     0x114   0x4620              MOV   R0,R4
\     0x116   0x.... 0x....   BL      HAL_ADC_Start
      108             reading = HAL_ADC_GetValue(&hadc2);
\     0x11A   0x4620              MOV   R0,R4
\     0x11C   0x.... 0x....   BL      HAL_ADC_GetValue
\     0x120   0x4607              MOV   R7,R0
      109             data = (reading / 45); // convert from ADC value to temp in C
\     0x122   0xB280              UXTH          R0,R0
\     0x124   0x212D              MOVS          R1,#+45
\     0x126   0xFB90 0xF6F1    SDIV  R6,R0,R1
      110             int_to_str(data);
\     0x12A   0xB2F0              UXTB R0,R6
\     0x12C   0x.... 0x....   BL      int_to_str
      111             LCD_SendCmd(0x01);
\     0x130   0x762E              STRB R6,[R5, #+24]
\     0x132   0x836F              STRH          R7,[R5, #+26]
\     0x134   0x2001              MOVS          R0,#+1
\     0x136   0x.... 0x....   BL      LCD_SendCmd
      112             for (int i=0; i<3; i++){
\     0x13A   0x2700              MOVS          R7,#+0
      113             LCD_SendData(num_out[i]);
\             ??main_3: (+1)
\     0x13C   0xF818 0x0007    LDRB R0,[R8, R7]
\     0x140   0x.... 0x....   BL      LCD_SendData
      114             }
\     0x144   0x1C7F              ADDS          R7,R7,#+1
\     0x146   0x2F03              CMP   R7,#+3
\     0x148   0xDBF8              BLT.N ??main_3
      115             //LCD_SendData('C');
      116             HAL_Delay(250); // delay for 250ms
\     0x14A   0x20FA              MOVS          R0,#+250
\     0x14C   0x.... 0x....   BL      HAL_Delay
\     0x150   0xE7E0              B.N     ??main_2
      117             /* USER CODE END WHILE */
      118
      119             /* USER CODE BEGIN 3 */
      120             }
      121             /* USER CODE END 3 */
```

```
      122           }
      123
      124           /**
      125           * @brief System Clock Configuration
      126           * @retval None
      127           */

  \                           In section .text, align 2, keep-with-next
      128           void SystemClock_Config(void)
      129           {
  \           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      ?Subroutine2
      130           RCC_OscInitTypeDef RCC_OscInitStruct = {0};
      131           RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
      132
      133           /** Configure the main internal regulator output voltage
      134           */
      135           __HAL_RCC_PWR_CLK_ENABLE();
  \           ??CrossCallReturnLabel_3: (+1)
  \     0x12 0x2000         MOVS         R0,#+0
  \     0x14 0x9000         STR   R0,[SP, #+0]
      136
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1
);
      137
      138           /** Initializes the RCC Oscillators according to the specified
parameters
      139           * in the RCC_OscInitTypeDef structure.
      140           */
      141           RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSE;
      142           RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  \     0x16 0xF44F 0x3380      MOV   R3,#+65536
  \     0x1A 0x....           LDR.N         R0,??DataTable4_7
```

```
\     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,??DataTable4_8
\     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]
      143             RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
      144             RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
      145             RCC_OscInitStruct.PLL.PLLM = 4;
      146             RCC_OscInitStruct.PLL.PLLN = 180;
      147             RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
      148             RCC_OscInitStruct.PLL.PLLQ = 4;
      149             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
         150            {
         151            Error_Handler();
  \      0x6E   0x.... 0x....     BL      Error_Handler
         152            }
         153
         154            /** Activate the Over-Drive mode
         155            */
         156            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
         157            {
         158            Error_Handler();
  \      0x78   0xB672            CPSID   I
  \              ??SystemClock_Config_2: (+1)
  \      0x7A   0xE7FE            B.N     ??SystemClock_Config_2
         159            }
         160
         161            /** Initializes the CPU, AHB and APB buses clocks
         162            */
         163            RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
         164
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
         165            RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  \              ??SystemClock_Config_1: (+1)
  \      0x7C   0x2102            MOVS    R1,#+2
  \      0x7E   0x9102            STR     R1,[SP, #+8]
         166            RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  \      0x80   0x2200            MOVS    R2,#+0
  \      0x82   0x9203            STR     R2,[SP, #+12]
  \      0x84   0x200F            MOVS    R0,#+15
         167            RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  \      0x86   0xF44F 0x51A0     MOV     R1,#+5120
  \      0x8A   0x9001            STR     R0,[SP, #+4]
  \      0x8C   0x9104            STR     R1,[SP, #+16]
         168            RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
  \      0x8E   0xF44F 0x5280      MOV   R2,#+4096
  \      0x92   0x9205        STR   R2,[SP, #+20]
         169
         170            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
         171            {
         172            Error_Handler();
  \      0x9E   0xB672                CPSID         I
  \             ??SystemClock_Config_4: (+1)
  \      0xA0   0xE7FE                B.N     ??SystemClock_Config_4
         173            }
         174            }
  \             ??SystemClock_Config_3: (+1)
  \      0xA2   0xB013                ADD   SP,SP,#+76
  \      0xA4   0xBD00                POP   {PC}

  \                            In section .text, align 2, keep-with-next
  \             ?Subroutine2: (+1)
  \      0x0    0x2214        MOVS        R2,#+20
  \      0x2    0x2100        MOVS        R1,#+0
  \      0x4    0xA801        ADD   R0,SP,#+4
  \      0x6    0x.... 0x....     B.W     memset
         175
         176            /**
         177            * @brief ADC2 Initialization Function
         178            * @param None
         179            * @retval None
         180            */
         181            static void MX_ADC2_Init(void)
         182            {
         183
         184            /* USER CODE BEGIN ADC2_Init 0 */
         185
         186            /* USER CODE END ADC2_Init 0 */
         187
         188            ADC_ChannelConfTypeDef sConfig = {0};
```

```
189
190         /* USER CODE BEGIN ADC2_Init 1 */
191
192         /* USER CODE END ADC2_Init 1 */
193
194         /** Configure the global features of the ADC (Clock, Resolution,
Data Alignment and number of conversion)
195         */
196         hadc2.Instance = ADC2;
197         hadc2.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
198         hadc2.Init.Resolution = ADC_RESOLUTION_12B;
199         hadc2.Init.ScanConvMode = DISABLE;
200         hadc2.Init.ContinuousConvMode = DISABLE;
201         hadc2.Init.DiscontinuousConvMode = DISABLE;
202         hadc2.Init.ExternalTrigConvEdge =
ADC_EXTERNALTRIGCONVEDGE_NONE;
203         hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
204         hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
205         hadc2.Init.NbrOfConversion = 1;
206         hadc2.Init.DMAContinuousRequests = DISABLE;
207         hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
208         if (HAL_ADC_Init(&hadc2) != HAL_OK)
209         {
210         Error_Handler();
211         }
212
213         /** Configure for the selected ADC regular channel its
corresponding rank in the sequencer and its sample time.
214         */
215         sConfig.Channel = ADC_CHANNEL_11;
216         sConfig.Rank = 1;
217         sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
218         if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
219         {
220         Error_Handler();
221         }
222         /* USER CODE BEGIN ADC2_Init 2 */
223
224         /* USER CODE END ADC2_Init 2 */
225
```

```
226             }
227
228             /**
229             * @brief GPIO Initialization Function
230             * @param None
231             * @retval None
232             */
233             static void MX_GPIO_Init(void)
234             {
235             GPIO_InitTypeDef GPIO_InitStruct = {0};
236
237             /* GPIO Ports Clock Enable */
238             __HAL_RCC_GPIOH_CLK_ENABLE();
239             __HAL_RCC_GPIOC_CLK_ENABLE();
240             __HAL_RCC_GPIOF_CLK_ENABLE();
241             __HAL_RCC_GPIOD_CLK_ENABLE();
242
243             /*Configure GPIO pin Output Level */
244             HAL_GPIO_WritePin(GPIOF, Register_Select_Pin|R_W_Pin,
GPIO_PIN_RESET);
245
246             /*Configure GPIO pin Output Level */
247             HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
248
|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
249
250             /*Configure GPIO pins : Register_Select_Pin R_W_Pin */
251             GPIO_InitStruct.Pin = Register_Select_Pin|R_W_Pin;
252             GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
253             GPIO_InitStruct.Pull = GPIO_NOPULL;
254             GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
255             HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
256
257             /*Configure GPIO pins : E_Pin PF14 */
258             GPIO_InitStruct.Pin = E_Pin|GPIO_PIN_14;
259             GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
260             GPIO_InitStruct.Pull = GPIO_NOPULL;
261             HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
262
```

```
        263             /*Configure GPIO pins : PD8 PD9 PD10 PD11
        264                                     PD12 PD13 PD14 PD15 */
        265             GPIO_InitStruct.Pin =
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
        266
|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
        267             GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
        268             GPIO_InitStruct.Pull = GPIO_NOPULL;
        269             GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
        270             HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
        271
        272             }
        273
        274             /* USER CODE BEGIN 4 */

  \                             In section .text, align 2, keep-with-next
        275             void LCD_init(){  // from datasheet
  \               LCD_init: (+1)
  \       0x0   0xB580          PUSH          {R7,LR}
        276             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
// set E = 0
  \       0x2   0x2200          MOVS          R2,#+0
  \       0x4   0xF44F 0x5100         MOV   R1,#+8192
  \       0x8   0x....          LDR.N         R0,??DataTable4
  \       0xA   0x.... 0x....       BL      HAL_GPIO_WritePin
        277             HAL_Delay(100);  // setup time
  \       0xE   0x2064          MOVS          R0,#+100
  \       0x10   0x.... 0x....      BL      HAL_Delay
        278             LCD_SendCmd(0x30);
  \       0x14   0x2030         MOVS          R0,#+48
  \       0x16   0x.... 0x....      BL      LCD_SendCmd
        279             HAL_Delay(30);
  \       0x1A   0x201E                 MOVS          R0,#+30
  \       0x1C   0x.... 0x....      BL      HAL_Delay
        280             LCD_SendCmd(0x30);
  \       0x20   0x2030         MOVS          R0,#+48
  \       0x22   0x.... 0x....      BL      LCD_SendCmd
        281             HAL_Delay(10);
  \       0x26   0x200A         MOVS          R0,#+10
  \       0x28   0x.... 0x....      BL      HAL_Delay
```

```
        282             LCD_SendCmd(0x30);
  \     0x2C  0x2030    MOVS    R0,#+48
  \     0x2E  0x.... 0x....  BL  LCD_SendCmd
        283             HAL_Delay(10);
  \     0x32  0x200A    MOVS    R0,#+10
  \     0x34  0x.... 0x....  BL  HAL_Delay
        284             LCD_SendCmd(0x38);
  \     0x38  0x2038    MOVS    R0,#+56
  \     0x3A  0x.... 0x....  BL  LCD_SendCmd
        285             LCD_SendCmd(0x80);
  \     0x3E  0x2080    MOVS    R0,#+128
  \     0x40  0x.... 0x....  BL  LCD_SendCmd
        286             LCD_SendCmd(0x0c);
  \     0x44  0x200C    MOVS    R0,#+12
  \     0x46  0x.... 0x....  BL  LCD_SendCmd
        287             LCD_SendCmd(0x06);
  \     0x4A  0xE8BD 0x4002  POP  {R1,LR}
  \     0x4E  0x2006    MOVS    R0,#+6
  \     0x50            REQUIRE LCD_SendCmd
  \     0x50            ;; // Fall through to label LCD_SendCmd
        288             }
        289

  \                     In section .text, align 2, keep-with-next
        290             void LCD_SendCmd(uint8_t cmd){
  \           LCD_SendCmd: (+1)
  \     0x0   0xB510    PUSH    {R4,LR}
        291             GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the
correct portion of the output to write to the data port of LCD
  \     0x2   0x.... 0x....  BL  ?Subroutine4
        292             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET);
// set to command mode
  \           ??CrossCallReturnLabel_8: (+1)
  \     0x6   0x2200    MOVS    R2,#+0
  \     0x8   0x....    B.N  ?Subroutine0
        293             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET);
// set to write mode
        294             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); //
give an E pulse
        295             HAL_Delay(1);
```

```
296             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
297             }

\                               In section .text, align 2, keep-with-next
\             ?Subroutine0: (+1)
\     0x0   0xF44F 0x6100       MOV   R1,#+2048
\     0x4   0x.... 0x....       BL    ?Subroutine3
\             ??CrossCallReturnLabel_6: (+1)
\     0x8   0x2200              MOVS  R2,#+0
\     0xA   0xF44F 0x5180       MOV   R1,#+4096
\     0xE   0x.... 0x....       BL    ?Subroutine3
\             ??CrossCallReturnLabel_5: (+1)
\     0x12  0x2201              MOVS  R2,#+1
\     0x14  0xF44F 0x5100       MOV   R1,#+8192
\     0x18  0x.... 0x....       BL    ?Subroutine3
\             ??CrossCallReturnLabel_4: (+1)
\     0x1C  0x2001              MOVS  R0,#+1
\     0x1E  0x.... 0x....       BL    HAL_Delay
\     0x22  0x4620              MOV   R0,R4
\     0x24  0xE8BD 0x4010       POP   {R4,LR}
\     0x28  0x2200              MOVS  R2,#+0
\     0x2A  0xF44F 0x5100       MOV   R1,#+8192
\     0x2E  0x.... 0x....       B.W   HAL_GPIO_WritePin

\                               In section .text, align 2, keep-with-next
\             ?Subroutine4: (+1)
\     0x0   0x....              LDR.N R3,??DataTable4_9
\     0x2   0x....              LDR.N R4,??DataTable4
\     0x4   0x0200              LSLS  R0,R0,#+8
\     0x6   0x6018              STR   R0,[R3, #+0]
\     0x8   0x4770              BX    LR

\                               In section .text, align 2, keep-with-next
\             ?Subroutine3: (+1)
\     0x0   0x4620              MOV   R0,R4
\     0x2   0x.... 0x....       B.W   HAL_GPIO_WritePin
298

\                               In section .text, align 2, keep-with-next
299             void LCD_SendData(uint8_t i){
```

```
  \              LCD_SendData: (+1)
  \      0x0   0xB510          PUSH          {R4,LR}
         300             GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the
correct portion of the output to write to the data port of LCD
  \      0x2   0x.... 0x....   BL      ?Subroutine4
         301             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); //
set to data mode
  \              ??CrossCallReturnLabel_9: (+1)
  \      0x6   0x2201          MOVS          R2,#+1
  \      0x8                   REQUIRE ?Subroutine0
  \      0x8                   ;; // Fall through to label ?Subroutine0
         302             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET);
// set to write mode
         303             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); //
give an E pulse
         304             HAL_Delay(1);
         305             HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
         306             }
         307


  \                             In section .text, align 2, keep-with-next
         308             void int_to_str(uint8_t number) {
  \              int_to_str: (+1)
  \      0x0   0xB530          PUSH          {R4,R5,LR}
         309             int Hex_offset = 0x30;
         310             int first_dig = (number % 1000)/100;  //separate the hundreds digit
         311             first_dig = first_dig + Hex_offset;
         312             num_out[0] = first_dig;
  \      0x2   0x2264          MOVS          R2,#+100
  \      0x4   0xFB90 0xF2F2   SDIV  R2,R0,R2
  \      0x8   0x....          LDR.N         R3,??DataTable4_6
  \      0xA   0xF102 0x0430   ADD   R4,R2,#+48
  \      0xE   0x701C          STRB R4,[R3, #+0]
         313             int second_dig = (number % 100)/10 ; //separate the tens digit
         314             second_dig = second_dig + Hex_offset;
         315             num_out[1] = second_dig;
  \      0x10   0x2164         MOVS          R1,#+100
         316             int third_dig = number % 10;  // separate the ones place
         317             third_dig = third_dig + Hex_offset;
         318             num_out[2] = third_dig;
```

```
\      0x12   0x240A        MOVS        R4,#+10
\      0x14   0xFB01 0x0212      MLS   R2,R1,R2,R0
\      0x18   0x250A        MOVS        R5,#+10
\      0x1A   0xFB90 0xF4F4      SDIV  R4,R0,R4
\      0x1E   0xFB92 0xF2F5      SDIV  R2,R2,R5
\      0x22   0xFB05 0x0014      MLS   R0,R5,R4,R0
\      0x26   0x3230        ADDS        R2,R2,#+48
\      0x28   0x3030        ADDS        R0,R0,#+48
\      0x2A   0x705A              STRB R2,[R3, #+1]
\      0x2C   0x7098        STRB R0,[R3, #+2]
       319            }
\      0x2E   0xBD30              POP   {R4,R5,PC}
       320            /* USER CODE END 4 */
       321
       322            /**
       323            * @brief  This function is executed in case of error occurrence.
       324            * @retval None
       325            */

\                             In section .text, align 2, keep-with-next
       326            void Error_Handler(void)
       327            {
       328            /* USER CODE BEGIN Error_Handler_Debug */
       329            /* User can add his own implementation to report the HAL error
return state */
       330              __disable_irq();
\            Error_Handler: (+1)
\      0x0   0xB672        CPSID       I
       331              while (1)
\            ??Error_Handler_0: (+1)
\      0x2   0xE7FE        B.N    ??Error_Handler_0
       332            {
       333            }
       334            /* USER CODE END Error_Handler_Debug */
       335            }

\                             In section .text, align 2, keep-with-next
\            ?Subroutine1: (+1)
\      0x0   0x9101        STR   R1,[SP, #+4]
\      0x2   0x2201        MOVS        R2,#+1
```

```
\      0x4   0x2100          MOVS         R1,#+0
\      0x6   0x9103          STR    R1,[SP, #+12]
\      0x8   0x9104          STR    R1,[SP, #+16]
\      0xA   0x9202          STR    R2,[SP, #+8]
\      0xC   0xA901          ADD    R1,SP,#+4
\      0xE   0x4770          BX     LR

\                            In section .text, align 4, keep-with-next
\             ??DataTable4:
\      0x0   0x4002'1400            DC32  0x40021400

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_1:
\      0x0   0x4002'3830            DC32  0x40023830

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_2:
\      0x0   0x4002'0C00            DC32  0x40020c00

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_3:
\      0x0   0x....'....            DC32  hadc2

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_4:
\      0x0   0x4001'2100            DC32  0x40012100

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_5:
\      0x0   0x0F00'0001            DC32  0xf000001

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_6:
\      0x0   0x....'....            DC32  num_out

\                            In section .text, align 4, keep-with-next
\             ??DataTable4_7:
\      0x0   0x4002'3840            DC32  0x40023840

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

```
\              ??DataTable4_8:
\      0x0   0x4000'7000          DC32  0x40007000

\                                 In section .text, align 4, keep-with-next
\              ??DataTable4_9:
\      0x0   0x4002'0C14          DC32  0x40020c14
       336
       337            #ifdef  USE_FULL_ASSERT
       338            /**
       339            * @brief  Reports the name of the source file and the source line
number
       340            *         where the assert_param error has occurred.
       341            * @param  file: pointer to the source file name
       342            * @param  line: assert_param error line source number
       343            * @retval None
       344            */
       345            void assert_failed(uint8_t *file, uint32_t line)
       346            {
       347            /* USER CODE BEGIN 6 */
       348            /* User can add his own implementation to report the file name and
line number,
       349            ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
*/
       350            /* USER CODE END 6 */
       351            }
       352            #endif /* USE_FULL_ASSERT */

  Maximum stack usage in bytes:

  .cstack Function
  ------- --------
       0   Error_Handler
       8   LCD_SendCmd
       8   -> HAL_Delay
       0   -> HAL_GPIO_WritePin
       8   -> HAL_GPIO_WritePin
       8   LCD_SendData
       8   -> HAL_Delay
       0   -> HAL_GPIO_WritePin
       8   -> HAL_GPIO_WritePin
```

```
   8   LCD_init
   8   -> HAL_Delay
   8   -> HAL_GPIO_WritePin
   0   -> LCD_SendCmd
   8   -> LCD_SendCmd
  80   SystemClock_Config
  80   -> Error_Handler
  80   -> HAL_PWREx_EnableOverDrive
  80   -> HAL_RCC_ClockConfig
  80   -> HAL_RCC_OscConfig
  80   -> memset
  12   int_to_str
  48   main
  48   -> Error_Handler
  48   -> HAL_ADC_ConfigChannel
  48   -> HAL_ADC_GetValue
  48   -> HAL_ADC_Init
  48   -> HAL_ADC_Start
  48   -> HAL_Delay
  48   -> HAL_GPIO_Init
  48   -> HAL_GPIO_WritePin
  48   -> HAL_Init
  48   -> LCD_SendCmd
  48   -> LCD_SendData
  48   -> LCD_init
  48   -> SystemClock_Config
  48   -> int_to_str
  48   -> memset


Section sizes:

Bytes  Function/Label
-----  --------------
   4  ??DataTable4
   4  ??DataTable4_1
   4  ??DataTable4_2
   4  ??DataTable4_3
   4  ??DataTable4_4
   4  ??DataTable4_5
```

```
        4  ??DataTable4_6
        4  ??DataTable4_7
        4  ??DataTable4_8
        4  ??DataTable4_9
        50  ?Subroutine0
        16  ?Subroutine1
        10  ?Subroutine2
        6  ?Subroutine3
        10  ?Subroutine4
        4  Error_Handler
        10  LCD_SendCmd
        8  LCD_SendData
        80  LCD_init
        166  SystemClock_Config
        76  hadc2
        data
        reading
        48  int_to_str
        338  main
        4  num_out


  80 bytes in section .bss
 786 bytes in section .text

 786 bytes of CODE memory
  80 bytes of DATA memory

Errors: none
Warnings: none
```

Task 3

```
/* 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 */
void LCD_init();
void LCD_SendCmd(uint8_t);
void LCD_SendData(uint8_t);
void int_to_str(uint8_t);
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc2;
```

```
/* USER CODE BEGIN PV */
uint8_t data = 0;
uint16_t reading = 0;
uint8_t num_out[3];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC2_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_ADC2_Init();
  /* USER CODE BEGIN 2 */
// PD8...PD15 is data out to display
// PF11 = RS, PF12 = R/W, PF13 = E for display
// ADC2 channel 11 is the ADC channel used (port PC1)
  LCD_init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        HAL_ADC_Start(&hadc2);
        reading = HAL_ADC_GetValue(&hadc2);
        data = (reading / 45); // convert from ADC value to temp in C
        int_to_str(data);
        LCD_SendCmd(0x01);
        for (int i=0; i<3; i++){
        LCD_SendData(num_out[i]);
        }
        //LCD_SendData('C');
        HAL_Delay(250); // delay for 250ms
        /* 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();
  }
}

/**
  * @brief ADC2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC2_Init(void)
{

  /* USER CODE BEGIN ADC2_Init 0 */

  /* USER CODE END ADC2_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC2_Init 1 */

  /* USER CODE END ADC2_Init 1 */

  /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and
number of conversion)
  */
  hadc2.Instance = ADC2;
  hadc2.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc2.Init.Resolution = ADC_RESOLUTION_12B;
  hadc2.Init.ScanConvMode = DISABLE;
  hadc2.Init.ContinuousConvMode = DISABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
```

```
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  hadc2.Init.DMAContinuousRequests = DISABLE;
  hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
          Error_Handler();
  }

  /** Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_11;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
          Error_Handler();
  }
  /* USER CODE BEGIN ADC2_Init 2 */

  /* USER CODE END ADC2_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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOF, Register_Select_Pin|R_W_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD,
GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15,
GPIO_PIN_RESET);

  /*Configure GPIO pins : Register_Select_Pin R_W_Pin */
  GPIO_InitStruct.Pin = Register_Select_Pin|R_W_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : E_Pin PF14 */
  GPIO_InitStruct.Pin = E_Pin|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  /*Configure GPIO pins : PD8 PD9 PD10 PD11
                          PD12 PD13 PD14 PD15 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void LCD_init(){  // from datasheet
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET); // set E = 0
  HAL_Delay(100);  // setup time
  LCD_SendCmd(0x30);
  HAL_Delay(30);
  LCD_SendCmd(0x30);
```

```
  HAL_Delay(10);
  LCD_SendCmd(0x30);
  HAL_Delay(10);
  LCD_SendCmd(0x38);
  LCD_SendCmd(0x80);
  LCD_SendCmd(0x0c);
  LCD_SendCmd(0x06);
}

void LCD_SendCmd(uint8_t cmd){
  GPIOD -> ODR = (cmd<<8)| (0x0000); // place the command in the correct portion of
the output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_RESET); // set to command
mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void LCD_SendData(uint8_t i){
  GPIOD -> ODR = (i<<8)| (0x0000); // place the command in the correct portion of the
output to write to the data port of LCD
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_11, GPIO_PIN_SET); // set to data mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_12, GPIO_PIN_RESET); // set to write mode
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_SET); // give an E pulse
  HAL_Delay(1);
  HAL_GPIO_WritePin(GPIOF, GPIO_PIN_13, GPIO_PIN_RESET);
}

void int_to_str(uint8_t number) {
  int Hex_offset = 0x30;
  int first_dig = (number % 1000)/100;  //separate the hundreds digit
  first_dig = first_dig + Hex_offset;
  num_out[0] = first_dig;
  int second_dig = (number % 100)/10 ; //separate the tens digit
  second_dig = second_dig + Hex_offset;
  num_out[1] = second_dig;
  int third_dig = number % 10;  // separate the ones place
  third_dig = third_dig + Hex_offset;
```

```
  num_out[2] = third_dig;
}
/* 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 */
```