Electrical and Computer Engineering Department
ECE 4510 Microcontroller Applications

Laboratory Design Project 1
Conveyor Belt Controller

Isaac Bagley and Dante Bailey

April 15, 2023

Table of Contents

Introduction **3**
Design **3**
Key Design Points 3
Clock 3
Timers 4
GPIO 5
Schematic Diagram 6
Results **7**
Buzzer Generation Screenshots 7
Motor Signal PWM Screenshots 8
Breadboard Setup 12
Conclusion **13**
Appendix A - C Code **14**
Main.c 14
It.c 27
Appendix B - .lst Files **35**
Main.lst 35
It.lst 72

Introduction

The goal for this project is to create a robust controller for a conveyor belt system. This controller consists of a start and stop control, as well as warning sequences for both start and stop procedures. Furthermore, the controller must be able to take an input frequency and change the duty cycle of the output to the conveyor belt motor according to the table below.

Input Frequency (kHz)	Duty Cycle of Output (%)
4.7	90
4.9	80
5.1	70
5.3	60
5.5	50
5.7	40
6.1	20
6.3	10

Table 1 - Input frequencies and output duty cycles

The start sequence starts when the start control is turned low, turns on a buzzer at a frequency of 5kHz and blinks a warning light 6 times at a rate of 1 blink per second. After the start sequence, the motor is turned on at 50% duty cycle and changed according to the values in table 1, and an LED is turned on to model the IR LED on the conveyor belt. The stop sequence starts when the stop control is turned low. It consists of a buzzer sounding at 3.5kHz while a warning light blinks 10 times at 2 blinks per second. After the stop sequence, the controller goes into a wait state to wait for the start signal to be turned low again, at which point the process starts again.

Design

Key Design Points

Clock

The first aspect of design was to determine the clock rate the system would run at. The clock rate used for the project is 100MHz, with timer clocks of 50MHz. This was chosen for ease of use with timer registers since 50 timer ticks is 1 us. For example, the 500 ms timer needed to turn on and off the light for the start sequence uses a timer with a prescaler of 50, which makes each tick 1us, and an ARR of 500,000 since 500,000 us = 500 ms.

Timers

Various timers were used to accomplish the goals of the project. In all, the design uses 4 timers. Two timers were used for PWM generation by manipulating their ARR and CCR1 registers. These timers are TIM1 and TIM8, which are used for powering the buzzer speaker and the motor control output respectively. The duty cycle of the output is the ratio of the CCR to the ARR, duty cycle (%) = CCR/ARR. The ARR determines the frequency of the output PWM wave.

For this project, TIM8 was used for the motor controls. A 30kHz wave is desired for the output to the conveyor belt, so an ARR value of 3333 with a prescaler of 2 was used, and the CCR value was changed based on the input frequency, but it started at 1667, or 50% of 3333 as specified in the project directions.

TIM 1 was used to power the buzzer speaker through PWM output. Since two different frequencies are needed for the start and stop sequences, different ARR values were used in each case. The prescaler is set up to 50, so that each clock tick is 1MHz. To achieve the 5kHz wave, an ARR value of 200 is used since 1,000,000/5,000 = 200. For the 3.5kHz output, an ARR value of 286 was used since 1,000,000/3,500 = 286. A duty cycle of 50% was used for both output waves, so the CCR value was half of the ARR.

There was one timer, TIM2, used for its output compare function to trigger a periodic interrupt for turning on and off the LED warning light at regular intervals. This function is used both in the start and stop sequences, with a change in ARR between the two sequences to change the frequency of the LED blinking. For the start sequence, an ARR value of 500,000 was used to trigger the interrupt every 500ms to toggle the state of the LED, so it would blink once per second at a 50% duty cycle. The ARR value is changed to 250,000 for the stop sequence so the interrupt would be triggered every 250ms, causing the LED to blink at a rate of 2 blinks per second.

A fourth timer, TIM4, was used for its input capture functionality to process the input waveform during the portion of the program that is outputting to the conveyor belt. This timer was set up to trigger the interrupt on every edge of an input wave, capture the time of the rising edge and falling edge, and find the difference between them to find the period time and duty cycle of the input wave. The period of the input wave is then inverted to find the frequency of the input, which is then used in another portion of the program to determine the PWM output.

GPIO

For functions of the design not controlled by timer output channels directly, GPIO pins were used. These functions include the IR LED model, the LED blinker, the start signal, and the stop signal.

The inputs of start and stop were implemented on GPIO pins with the signal itself coming from a bounce-free switch on a breadboard. The start signal is polled during the idle state to determine when to begin the start sequence, and the stop signal is polled during the operating state to determine when to begin the stop sequence. Start is on PD0, and stop is on PD1.

The output to the IR LED and the LED blinker were implemented using GPIO outputs buffered by a 541 chip going to external LEDs. PD2 is the IR LED and PD3 is the blinker LED. The blinker LED is toggled when the TIM2 interrupt is triggered. The IR LED is turned on only when the program is in the output to motor phase.

Schematic Diagram

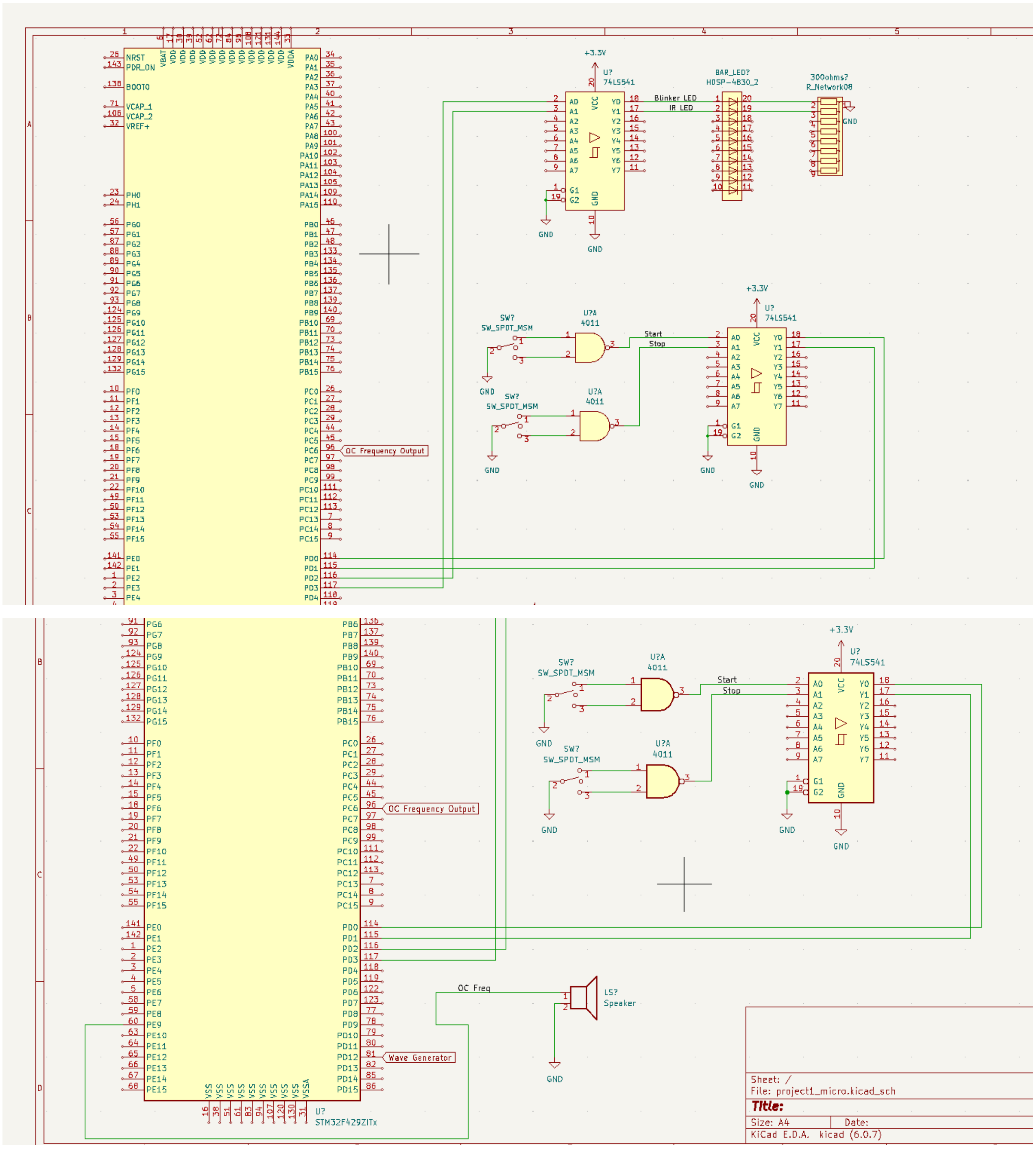
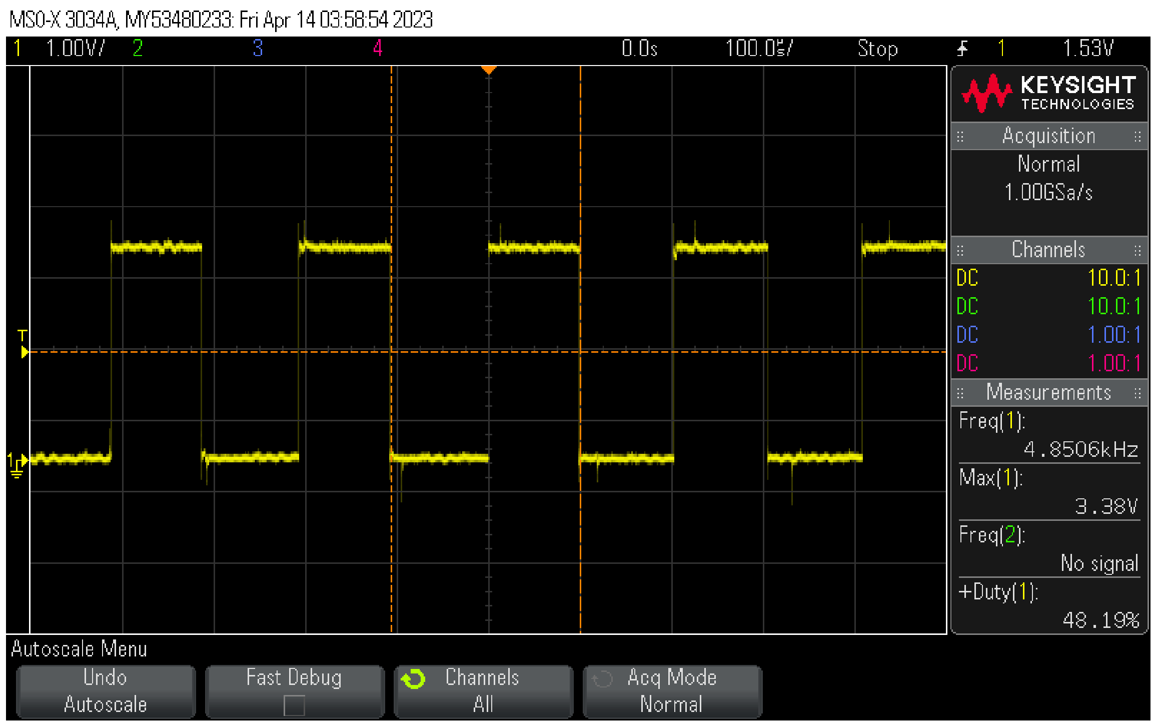


Figure 1 - The schematic diagram for the circuit of the project

Results

Buzzer Generation Screenshots



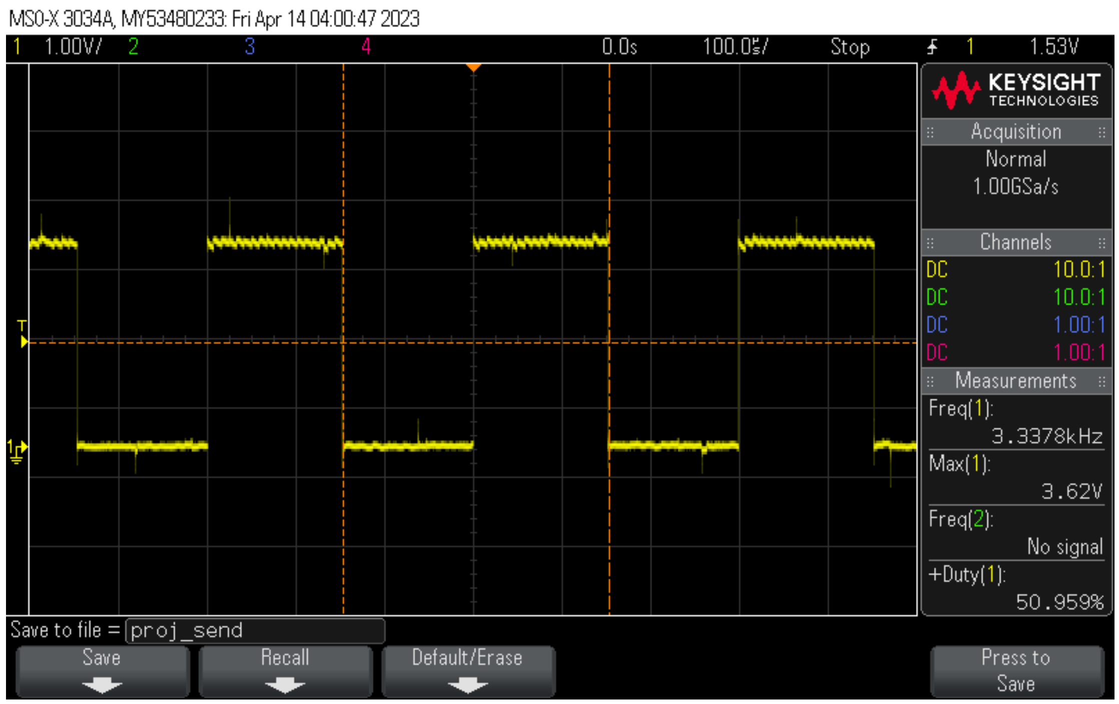
```
MSO-X 3034A, MY53480233: Fri Apr 14 03:58:54 2023
1  1.00V/  2  3  4      0.0s  100.0µs/  Stop  ⨍  1  1.53V

Acquisition
Normal
1.00GSa/s

Channels
DC  10.0:1
DC  10.0:1
DC  1.00:1
DC  1.00:1

Measurements
Freq(1):
      4.8506kHz
Max(1):
      3.38V
Freq(2):
      No signal
+Duty(1):
      48.19%
```

Figure 2 - The output PWM for powering the buzzer speaker during the start sequence



```
MSO-X 3034A, MY53480233: Fri Apr 14 04:00:47 2023
1  1.00V/  2  3  4      0.0s  100.0µs/  Stop  ⨍  1  1.53V

Acquisition
Normal
1.00GSa/s

Channels
DC  10.0:1
DC  10.0:1
DC  1.00:1
DC  1.00:1

Measurements
Freq(1):
      3.3378kHz
Max(1):
      3.62V
Freq(2):
      No signal
+Duty(1):
      50.959%

Save to file = proj_send
```

Figure 3 - The output PWM signal for the buzzer during the end sequence

Motor Signal PWM Screenshots

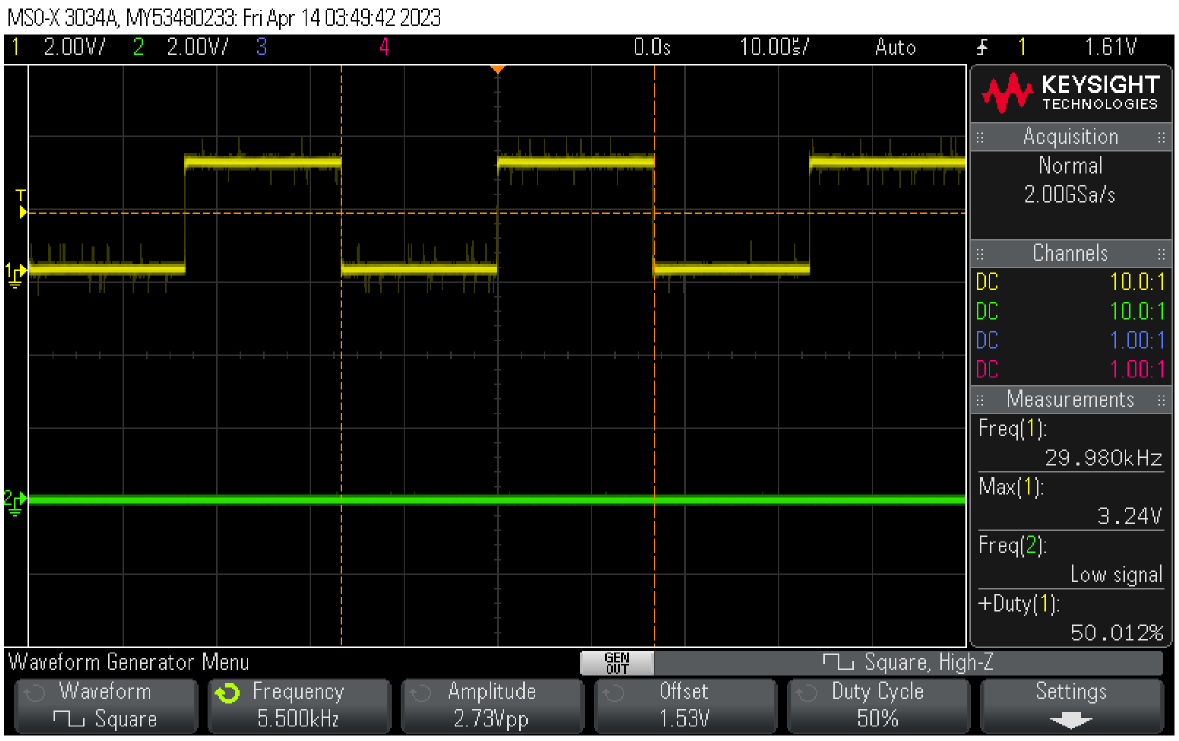


Figure 4 - The base case of outputting a 50% duty cycle 30kHz wave on a 5.5kHz input

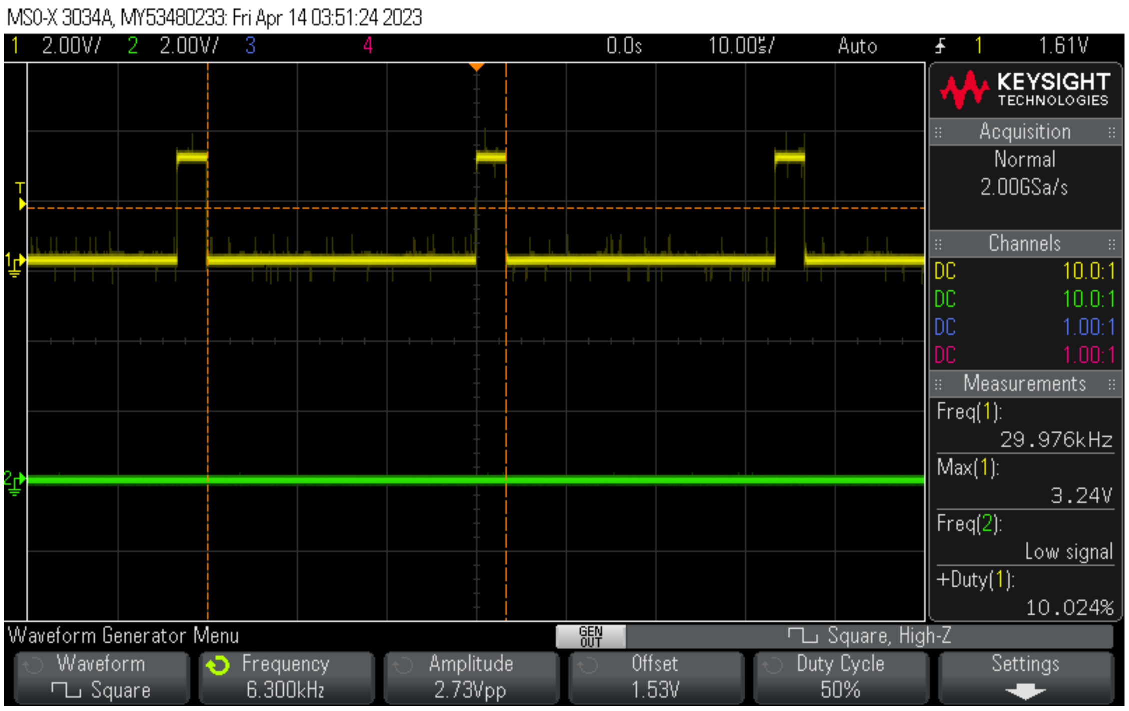


Figure 5 - Outputting a 10% duty cycle wave at 30kHz for a 6.3kHz input signal

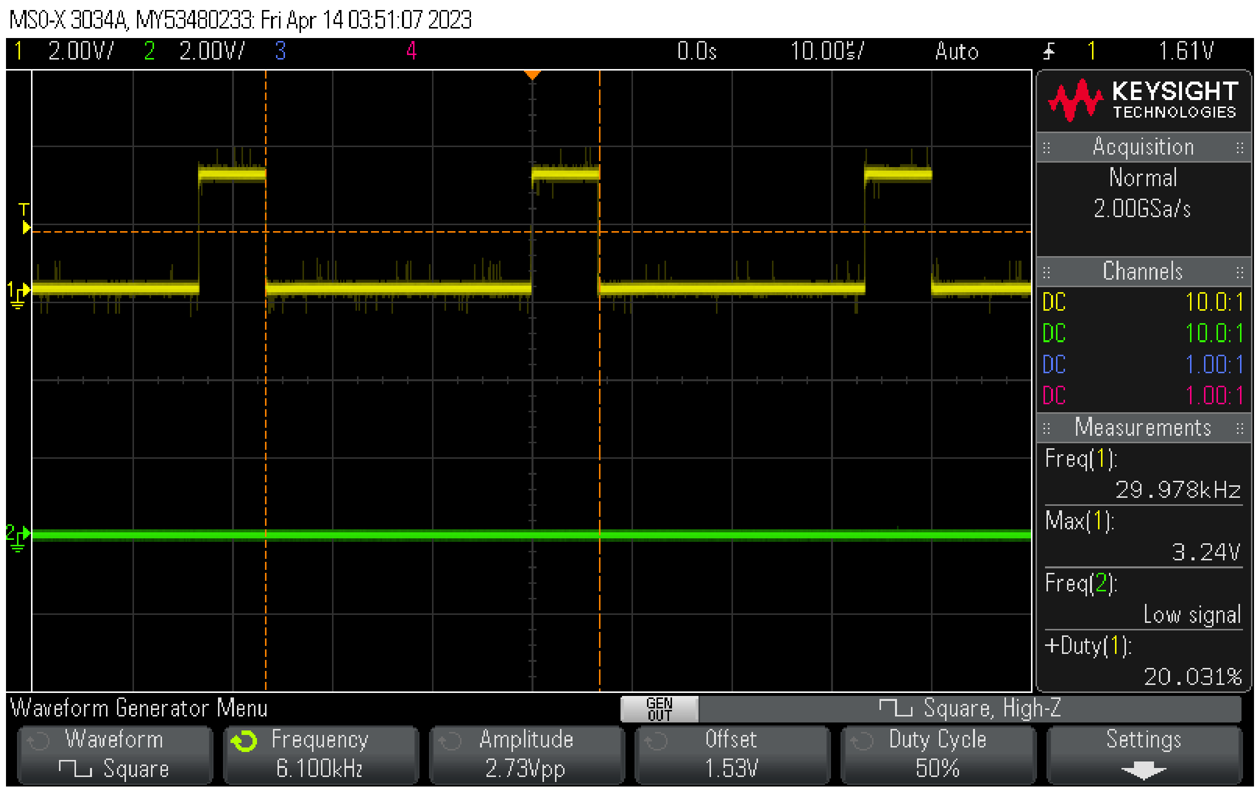


Figure 6 - Outputting a 20% duty cycle wave at 30kHz for a 6.1kHz input signal

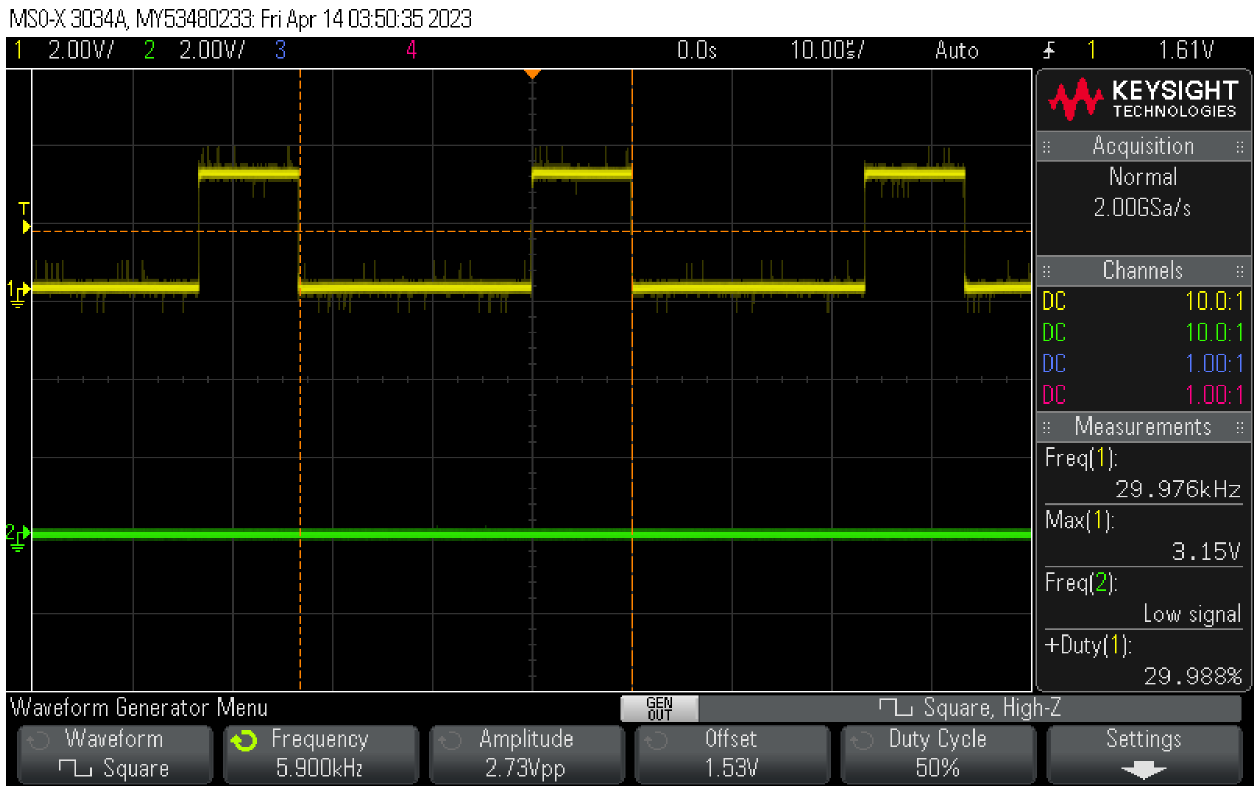


Figure 7 - Outputting a 30% duty cycle wave at 30kHz for a 5.9kHz input signal

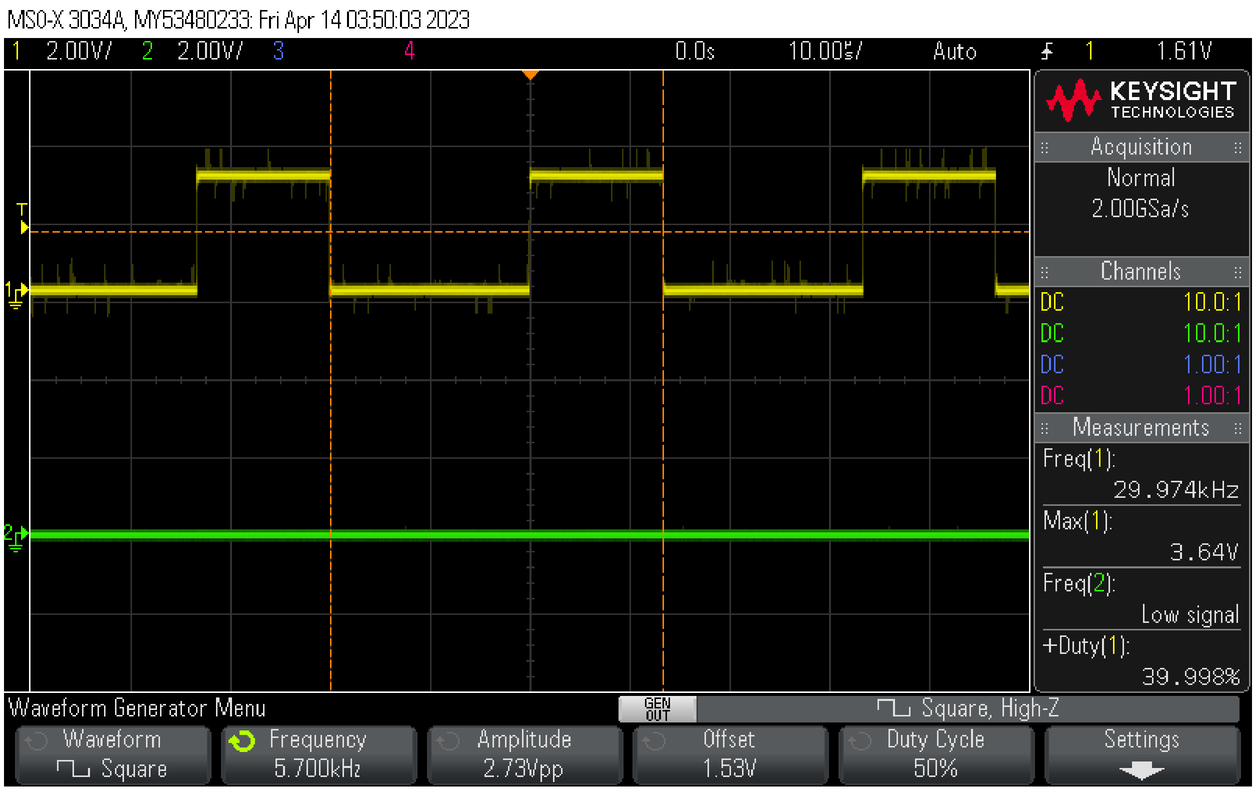


Figure 8 - Outputting a 40% duty cycle wave at 30kHz for a 5.7kHz input signal

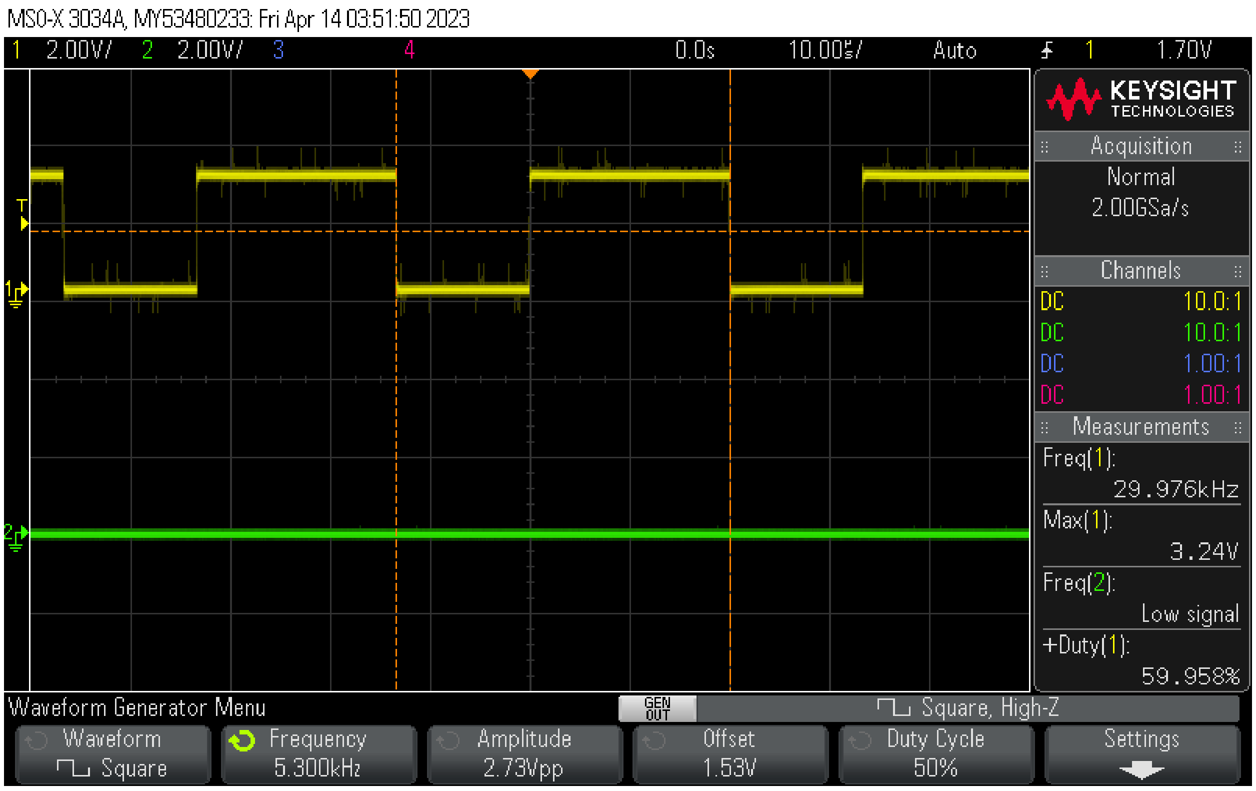
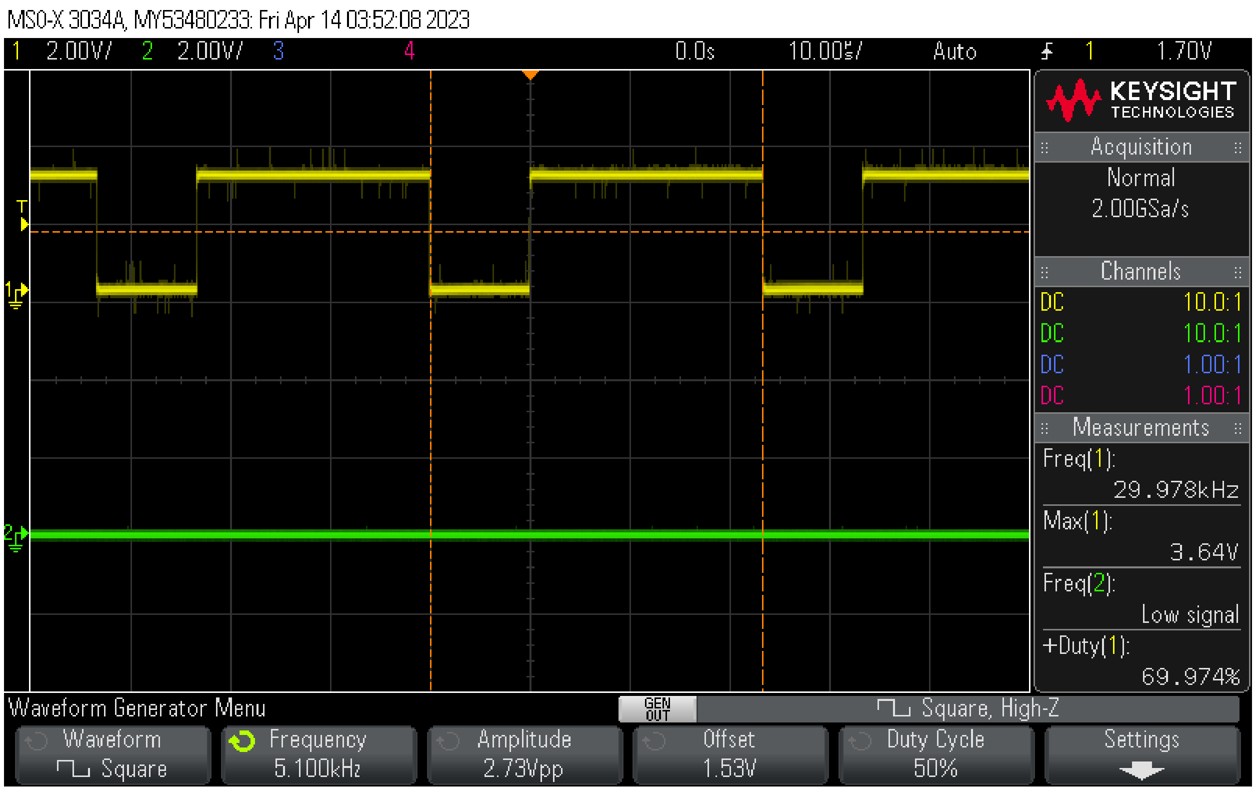


Figure 9 - Outputting a 60% duty cycle wave at 30kHz for a 5.3kHz input signal


MSO-X 3034A, MY53480233: Fri Apr 14 03:52:08 2023

1 2.00V/ 2 2.00V/ 3 4 0.0s 10.00µs/ Auto 1 1.70V

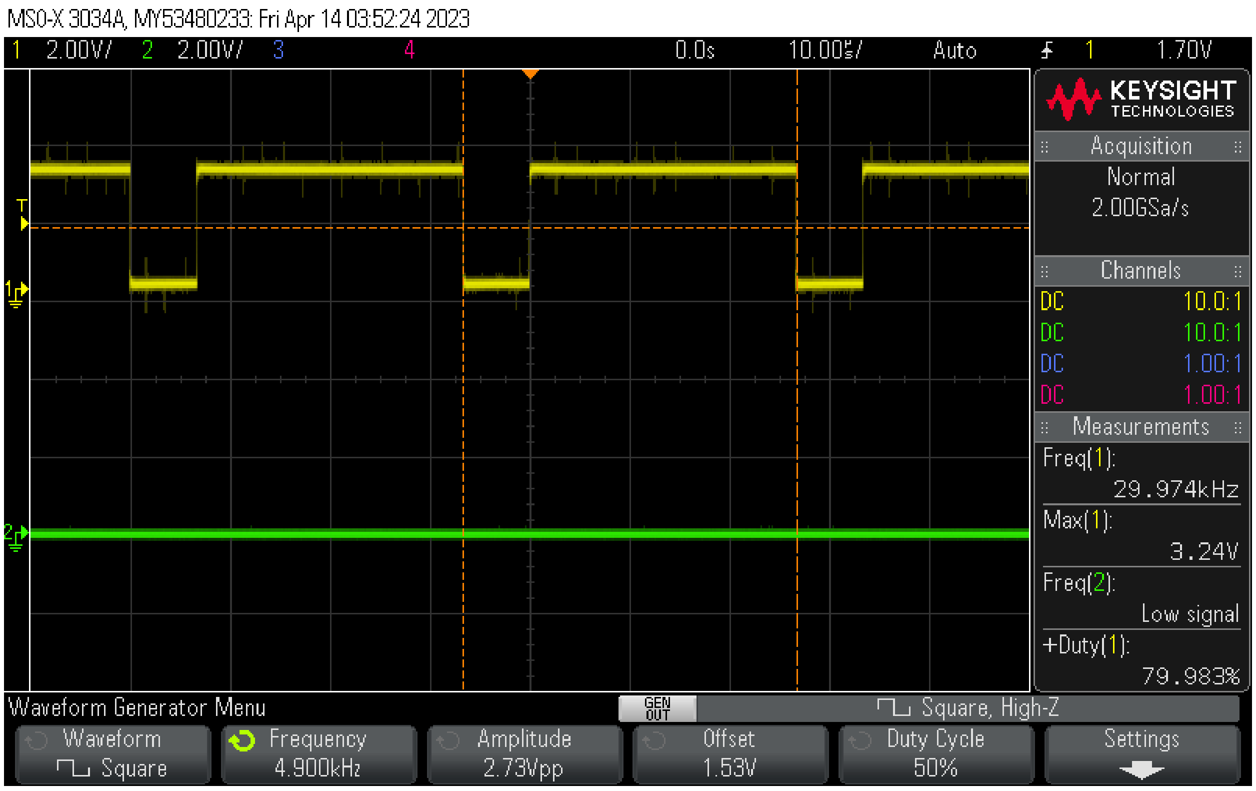
Acquisition
Normal
2.00GSa/s

Channels
DC 10.0:1
DC 10.0:1
DC 1.00:1
DC 1.00:1

Measurements
Freq(1): 29.978kHz
Max(1): 3.64V
Freq(2): Low signal
+Duty(1): 69.974%

Waveform Generator Menu — Square, High-Z
Waveform: Square
Frequency: 5.100kHz
Amplitude: 2.73Vpp
Offset: 1.53V
Duty Cycle: 50%
Settings

Figure 10 - Outputting a 70% duty cycle wave at 30kHz for a 5.1kHz input signal


MSO-X 3034A, MY53480233: Fri Apr 14 03:52:24 2023

1 2.00V/ 2 2.00V/ 3 4 0.0s 10.00µs/ Auto 1 1.70V

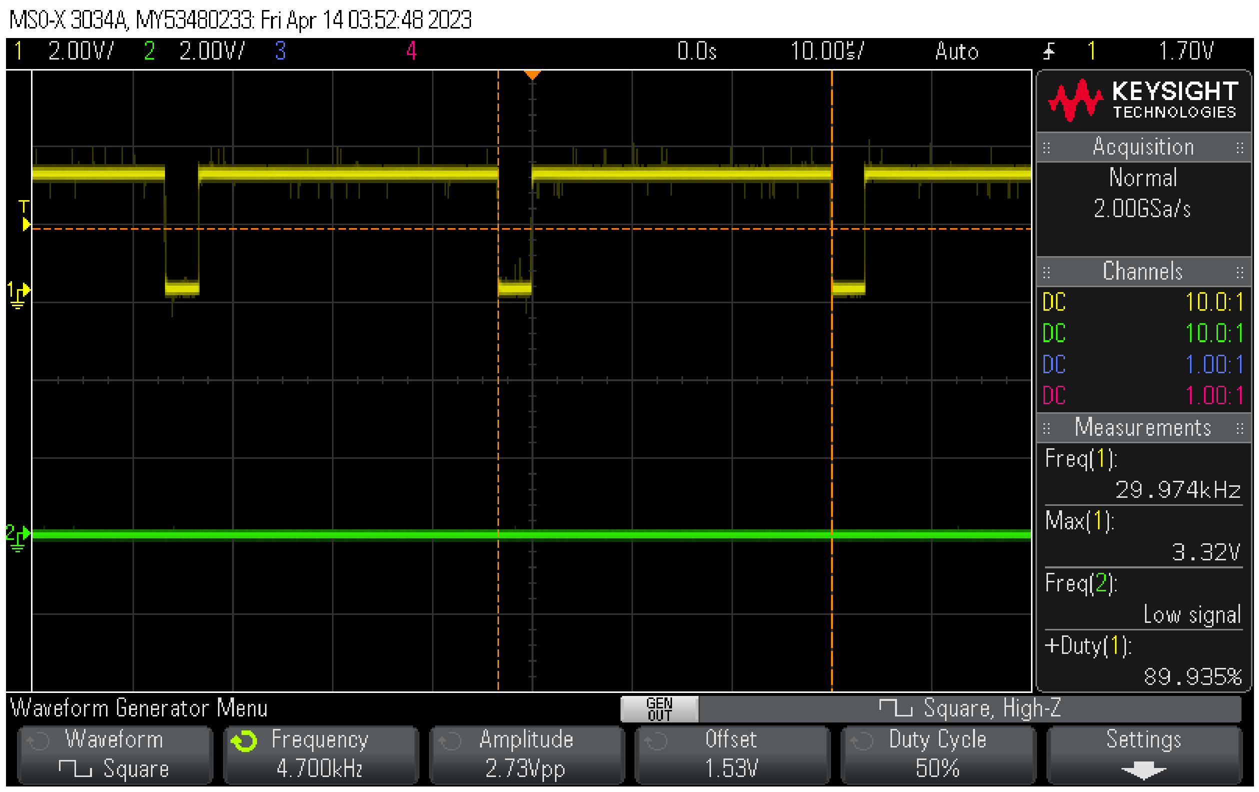
Acquisition
Normal
2.00GSa/s

Channels
DC 10.0:1
DC 10.0:1
DC 1.00:1
DC 1.00:1

Measurements
Freq(1): 29.974kHz
Max(1): 3.24V
Freq(2): Low signal
+Duty(1): 79.983%

Waveform Generator Menu — Square, High-Z
Waveform: Square
Frequency: 4.900kHz
Amplitude: 2.73Vpp
Offset: 1.53V
Duty Cycle: 50%
Settings

Figure 11 - Outputting an 80% duty cycle wave at 30kHz for a 4.9kHz input signal


MSO-X 3034A, MY53480233: Fri Apr 14 03:52:48 2023

1 2.00V/ 2 2.00V/ 3 4 0.0s 10.00µ/ Auto 1 1.70V

Acquisition: Normal, 2.00GSa/s

Channels: DC 10.0:1; DC 10.0:1; DC 1.00:1; DC 1.00:1

Measurements:
Freq(1): 29.974kHz
Max(1): 3.32V
Freq(2): Low signal
+Duty(1): 89.935%

Waveform Generator Menu — GEN OUT — Square, High-Z
Waveform: Square
Frequency: 4.700kHz
Amplitude: 2.73Vpp
Offset: 1.53V
Duty Cycle: 50%
Settings

Figure 12 - Outputting a 90% duty cycle wave at 30kHz for a 4.7kHz input signal

Breadboard Setup

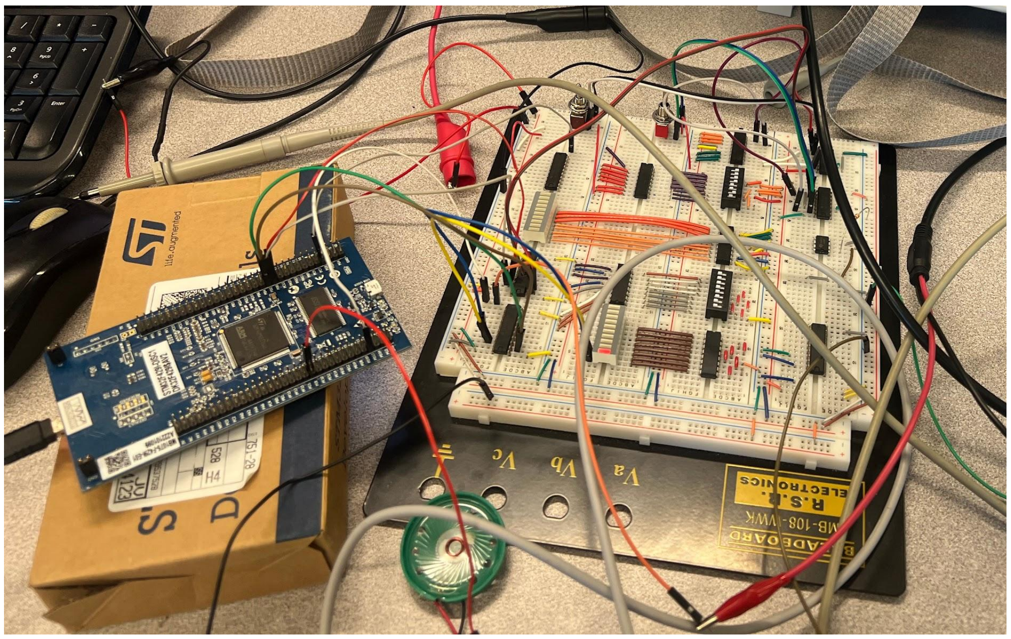


Figure 13 - The breadboard setup for the project

Conclusion

In conclusion, the project was successful. The program was able to process both basic inputs through GPIO, and dynamic inputs using input capture in order to control basic GPIO outputs and dynamic PWM outputs. As shown in the results section, the project was able to meet all of the specifications from the project assignment. The ability to start the program multiple times without a restart of the system makes it a robust controller for the conveyor belt. The output duty cycle being the same as the specified duty cycle requirements, and not changing before the specified input value, means that the controller system provides a stable output within the specifications given for the project.

Appendix A - C Code

Main.c

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

******************************************************************************
  * @file           : main.c
  * @brief          : Main program body

******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim8;

/* USER CODE BEGIN PV */
int state = 0;
uint8_t START = 1;
uint8_t STOP = 1;
int blinks = 0;
float freq = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM4_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM8_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_TIM4_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  MX_TIM8_Init();
  /* USER CODE BEGIN 2 */
  TIM1 -> PSC = 49 - 1; // set the prescalar for the PWM generation
  TIM1 -> ARR = 100; // 1MHz / 5kHz = 100 ticks/5kHz wave
  TIM1 -> CCR1 = 50; // 200/2 = 100 = 50% duty cycle
  HAL_TIM_PWM_Init(&htim1);

  TIM8 -> ARR = 1667; // 100MHz/30kHz = 1667 ticks/30kHz wave
  TIM8 -> CCR1 = 1667; // 1667 * .5 = 1667 for 50% duty cycle
  HAL_TIM_PWM_Init(&htim8);
  /* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{ START = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_0); // start = PD0
        STOP = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_1); // start = PD1
        HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1); // turn off the buzzer

        if (START == GPIO_PIN_SET && state == 0) {
        blinks = 0; // reset the blink count for reset
        }


        // the start sequence
        else if (((START == GPIO_PIN_RESET) && (state == 0)) && (blinks < 7)){
        TIM2 -> ARR = 500000; // the timer will trigger every half second
        TIM1 -> PSC = 50-1;
        TIM1 -> ARR = 100; // 1MHz / 5kHz = 100 ticks/5kHz wave
        TIM1 -> CCR1 = 50; // 100/2 = 50 = 50% duty cycle
        HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1); // start the blinking
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // start the buzzer
        if (blinks == 6){
        HAL_TIM_OC_Stop_IT(&htim2, TIM_CHANNEL_1); // stop the blink interrupt
        TIM8 -> ARR = 1667;
        TIM8 -> CCR1 = 834;
        HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1); // turn on the output
        HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1); // turn on the input compare
        state++; // goto next state after 6 blinks
        }
        }

        // the PWM generation
        else if (state == 1 && STOP == 1){
        blinks = 0; // reset the blink counter
        HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1); // turn off the buzzer
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET); // turn on the IR LED
        if (freq > 6300){
        TIM8 -> CCR1 = 167; // 1667/10 = 167 for 10% duty cycle
        }
        else if (freq > 6100){
        TIM8 -> CCR1 = 334; // 1667/5 = 334 for 20% duty cycle
```

```
	}
	else if (freq > 5900){
	TIM8 -> CCR1 = 500; // 1667 * .3 = 500 for 30% duty cycle
	}
	else if (freq > 5700){
	TIM8 -> CCR1 = 667; // 1667 * .4 = 667 for 40% duty cycle
	}
	else if (freq > 5500){
	TIM8 -> CCR1 = 834; // 1667 * .5 = 834 for 50% duty cycle
	}
	else if (freq > 5300){
	TIM8 -> CCR1 = 1000; // 1667 * .6 = 1000 for 60% duty cycle
	}
	else if (freq > 5100){
	TIM8 -> CCR1 = 1167; // 1667 * .7 = 1167 for 70% duty cycle
	}
	else if (freq > 4900){
	TIM8 -> CCR1 = 1334; // 1667 * .8 = 1334 for 80% duty cycle
	}
	else if (freq > 4700){
	TIM8 -> CCR1 = 1500; // 1667 * .9 = 1500 for 90% duty cycle
	}
	else {
	TIM8 -> CCR1 = 834; // 1667 * .5 = 834 for 50% duty cycle
	// shouldn't get here, just in case
	}
	}

	// the stop sequence
	else if (STOP == 0 && state == 1) {
	HAL_TIM_OC_Stop_IT(&htim4, TIM_CHANNEL_1); // stop the input capture
	HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1); // stop the output
	HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET); // turn off the IR
LED
	TIM1 -> ARR = 90;
	TIM1 -> CCR1 = 45;
	HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // turn on the buzzer
	TIM2 -> ARR = 250000; // the timer will trigger every quarter second
	HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1); // start blinking the light
```

```
        
        if (blinks == 10){
        HAL_TIM_OC_Stop_IT(&htim2, TIM_CHANNEL_1); // stop the blinking at 10 blinks
        state = 0; // restart the process
        }
        }

        /* 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_SCALE3);

  /** 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 = 100;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
          Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
  {
          Error_Handler();
  }
}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

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

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 50;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 200;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
  {
        Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
        Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
        Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) != HAL_OK)
  {
        Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */
  /* USER CODE END TIM1_Init 2 */
  HAL_TIM_MspPostInit(&htim1);

}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

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

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 50;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 500000;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_OC_Init(&htim2) != HAL_OK)
  {
        Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &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(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
        Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 0;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 65535;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_IC_Init(&htim4) != HAL_OK)
  {
        Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
  {
        Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim4, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
  {
        Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */

}

/**
  * @brief TIM8 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM8_Init(void)
{

  /* USER CODE BEGIN TIM8_Init 0 */

  /* USER CODE END TIM8_Init 0 */

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

  /* USER CODE BEGIN TIM8_Init 1 */

  /* USER CODE END TIM8_Init 1 */
  htim8.Instance = TIM8;
  htim8.Init.Prescaler = 0;
  htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim8.Init.Period = 65535;
  htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim8.Init.RepetitionCounter = 0;
  htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim8) != HAL_OK)
  {
        Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL_OK)
  {
        Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim8, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
        Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim8, &sBreakDeadTimeConfig) != HAL_OK)
  {
        Error_Handler();
  }
  /* USER CODE BEGIN TIM8_Init 2 */

  /* USER CODE END TIM8_Init 2 */
  HAL_TIM_MspPostInit(&htim8);

}

/**
  * @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_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, IR_LED_Pin|LED_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : Start_Pin Stop_Pin */
  GPIO_InitStruct.Pin = Start_Pin|Stop_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pins : IR_LED_Pin LED_Pin */
  GPIO_InitStruct.Pin = IR_LED_Pin|LED_Pin;
  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 */

/* 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 */
```

It.c

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

```
/**

******************************************************************************
  * @file         stm32f4xx_it.c
  * @brief   Interrupt Service Routines.

******************************************************************************
  * @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"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */

/* USER CODE END TD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

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

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
int upper = 0;
float high_pulse = 0;
float low_pulse = 0;
int active_time = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/
extern TIM_HandleTypeDef htim1;
extern TIM_HandleTypeDef htim2;
extern TIM_HandleTypeDef htim4;
/* USER CODE BEGIN EV */
extern int state;
extern int blinks;
extern uint8_t START;
extern uint8_t STOP;
extern float freq;
/* USER CODE END EV */

/******************************************************************************
*/
/*          Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************
*/
/**
```

```
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
	/* USER CODE BEGIN W1_HardFault_IRQn 0 */
	/* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
	/* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
```

```
        /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
        /* USER CODE BEGIN W1_BusFault_IRQn 0 */
        /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
        /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
        /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************
*/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************
*/

/**
  * @brief This function handles TIM1 capture compare interrupt.
  */
void TIM1_CC_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_CC_IRQn 0 */

  /* USER CODE END TIM1_CC_IRQn 0 */
  HAL_TIM_IRQHandler(&htim1);
  /* USER CODE BEGIN TIM1_CC_IRQn 1 */

  /* USER CODE END TIM1_CC_IRQn 1 */
}

/**
  * @brief This function handles TIM2 global interrupt.
  */
void TIM2_IRQHandler(void)
{
  /* USER CODE BEGIN TIM2_IRQn 0 */
  HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_3); // toggle the LED
  if (upper == 0){
   upper = 1;
  }
  else{
   blinks++; // add one to the blink counter
   upper = 0;
  }
  /* USER CODE END TIM2_IRQn 0 */
  HAL_TIM_IRQHandler(&htim2);
  /* USER CODE BEGIN TIM2_IRQn 1 */
  TIM1 -> CNT = 0; // reset the timer count
  /* USER CODE END TIM2_IRQn 1 */
}

/**
  * @brief This function handles TIM4 global interrupt.
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */
  if (HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_12) == GPIO_PIN_SET){
	  high_pulse = TIM4 -> CNT / 50;  // high_pulse holds the time from start to rising edge
  }
  else {
	  low_pulse = TIM4 -> CNT / 50; // low_pulse stores the time from start to falling edge
	  freq = (1/low_pulse)*1000000; // low_pulse is the period in us
	  // freq should be in Hz
	  TIM4->CNT = 0; //reset the timer count
  }
  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

Appendix B - .lst Files

Main.lst

```
###############################################################################
#
#
# IAR ANSI C/C++ Compiler V9.20.4.327/W64 for ARM            13/Apr/2023  15:17:02
# Copyright 1999-2022 IAR Systems AB.
#
#        Cpu mode       =  thumb
#        Endian         =  little
#        Source file    =
#        S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\Core\Src\main.c
#        Command line           =
#        -f
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\User\Core\main.o.rsp
#        (S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\Core\Src\main.c
#        -D USE_HAL_DRIVER -D STM32F429xx -lC
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\List\Ap
plication\User\Core
#        -o
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\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\Project1\Project1\EWARM/../Core/Inc\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/STM3
2F4xx_HAL_Driver/Inc\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/STM3
2F4xx_HAL_Driver/Inc/Legacy\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/CMSI
S/Device/ST/STM32F4xx/Include\
#        -I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/CMSI
S/Include\
#        -Ohz) --dependencies=n
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\User\Core\main.o.d
#        Locale           =  C
#        List file        =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\List\Ap
plication\User\Core\main.lst
#        Object file      =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\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\Project1\Project1\Core\Src\main.c
                /* USER CODE BEGIN Header */
                /**

******************************************************************************
                * @file           : main.c
                * @brief          : Main program body

******************************************************************************
                * @attention
```

```
*
* Copyright (c) 2023 STMicroelectronics.
        * All rights reserved.
*
        * This software is licensed under terms that can be found in the LICENSE file
        * in the root directory of this software component.
        * If no LICENSE file comes with this software, it is provided AS-IS.
        *
******************************************************************************
        */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

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

/* USER CODE END PM */

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

\               In section .data, align 4
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
```

```
     44      TIM_HandleTypeDef htim4;
     45      TIM_HandleTypeDef htim8;
     46
     47      /* USER CODE BEGIN PV */
     48      int state = 0;
     49      uint8_t START = 1;
\            START:
\    0x0   0x01            DC8 1
     50      uint8_t STOP = 1;
\            STOP:
\    0x1   0x01            DC8 1
\    0x2   0x00 0x00       DC8 0, 0
\            htim1:
\    0x4   0x0000'0000     DC32 0x0
\    0x8                   DS8 28
\    0x24   0x0000'0000    DC32 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0

\            0x0000'0000

\            0x0000'0000

\            0x0000'0000

\            0x0000'0000

\            0x0000'0000

\            0x0000'0000
\    0x40                  DS8 12
\            htim2:
\    0x4C   0x0000'0000           DC32 0x0
\    0x50                  DS8 28
\    0x6C   0x0000'0000           DC32 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0

\            0x0000'0000

\            0x0000'0000

\            0x0000'0000
\               0x0000'0000

\               0x0000'0000

\               0x0000'0000
\       0x88                    DS8 12
\               htim4:
\       0x94   0x0000'0000     DC32 0x0
\       0x98                    DS8 28
\       0xB4   0x0000'0000              DC32 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000
\       0xD0                    DS8 12
\               htim8:
\       0xDC   0x0000'0000              DC32 0x0
\       0xE0                    DS8 28
\       0xFC   0x0000'0000              DC32 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000

\               0x0000'0000
\       0x118                   DS8 12
\             state:
\      0x124   0x0000'0000          DC32 0
       51     int blinks = 0;
\             blinks:
\      0x128   0x0000'0000          DC32 0
       52     float freq = 0;
\             freq:
\      0x12C   0x0000'0000          DC32 0x0
       53     /* USER CODE END PV */
       54
       55     /* Private function prototypes -----------------------------------------------*/
       56     void SystemClock_Config(void);
       57     static void MX_GPIO_Init(void);
       58     static void MX_TIM4_Init(void);
       59     static void MX_TIM1_Init(void);
       60     static void MX_TIM2_Init(void);
       61     static void MX_TIM8_Init(void);
       62     /* USER CODE BEGIN PFP */
       63
       64     /* USER CODE END PFP */
       65
       66     /* Private user code ---------------------------------------------------------*/
       67     /* USER CODE BEGIN 0 */
       68
       69     /* USER CODE END 0 */
       70
       71     /**
       72            * @brief  The application entry point.
       73            * @retval int
       74            */

\                    In section .text, align 4, keep-with-next
       75     int main(void)
       76     {
\             main: (+1)
\      0x0   0xE92D 0x4FF0          PUSH  {R4-R11,LR}
\      0x4   0xB091          SUB    SP,SP,#+68
       77            /* USER CODE BEGIN 1 */
       78
       79            /* USER CODE END 1 */
```

```

            /* MCU Configuration--------------------------------------------------------*/

            /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
            HAL_Init();
\   0x6   0x.... 0x....       BL      HAL_Init

            /* USER CODE BEGIN Init */

            /* USER CODE END Init */

            /* Configure the system clock */
            SystemClock_Config();
\   0xA   0x.... 0x....       BL      SystemClock_Config

            /* USER CODE BEGIN SysInit */

            /* USER CODE END SysInit */

            /* Initialize all configured peripherals */
            MX_GPIO_Init();
\   0xE   0x.... 0x....       BL      ?Subroutine3
\         ??CrossCallReturnLabel_10: (+1)
\   0x12  0x2000              MOVS            R0,#+0
\   0x14  0x9000              STR     R0,[SP, #+0]
\   0x16  0xF64F 0x79FF       MOVW            R9,#+65535
\   0x1A  0x.... 0x....       LDR.W           R0,??DataTable1_9
\   0x1E  0x.... 0x....       LDR.W           R8,??DataTable1_10
\   0x22  0x6801              LDR     R1,[R0, #+0]
\   0x24  0x.... 0x....       LDR.W           R5,??DataTable1_11
\   0x28  0xF041 0x0180       ORR     R1,R1,#0x80
\   0x2C  0x6001              STR     R1,[R0, #+0]
\   0x2E  0xF105 0x0694       ADD     R6,R5,#+148
\   0x32  0x6802              LDR     R2,[R0, #+0]
\   0x34  0xF002 0x0280       AND     R2,R2,#0x80
\   0x38  0x9200              STR     R2,[SP, #+0]
\   0x3A  0x2200              MOVS            R2,#+0
\   0x3C  0x9900              LDR     R1,[SP, #+0]
\   0x3E  0x9200              STR     R2,[SP, #+0]
\   0x40  0x6803              LDR     R3,[R0, #+0]
\       0x42   0xF043 0x0310          ORR    R3,R3,#0x10
\       0x46   0x6003           STR    R3,[R0, #+0]
\       0x48   0x6801           LDR    R1,[R0, #+0]
\       0x4A   0xF001 0x0110          AND    R1,R1,#0x10
\       0x4E   0x9100           STR    R1,[SP, #+0]
\       0x50   0x9900           LDR    R1,[SP, #+0]
\       0x52   0x9200           STR    R2,[SP, #+0]
\       0x54   0x6803           LDR    R3,[R0, #+0]
\       0x56   0xF043 0x0308          ORR    R3,R3,#0x8
\       0x5A   0x6003           STR    R3,[R0, #+0]
\       0x5C   0x6801           LDR    R1,[R0, #+0]
\       0x5E   0xF001 0x0108          AND    R1,R1,#0x8
\       0x62   0x9100           STR    R1,[SP, #+0]
\       0x64   0x9900           LDR    R1,[SP, #+0]
\       0x66   0x9200           STR    R2,[SP, #+0]
\       0x68   0x210C           MOVS          R1,#+12
\       0x6A   0x6803           LDR    R3,[R0, #+0]
\       0x6C   0xF043 0x0304          ORR    R3,R3,#0x4
\       0x70   0x6003           STR    R3,[R0, #+0]
\       0x72   0x6800           LDR    R0,[R0, #+0]
\       0x74   0xF000 0x0004          AND    R0,R0,#0x4
\       0x78   0x9000           STR    R0,[SP, #+0]
\       0x7A   0x9800           LDR    R0,[SP, #+0]
\       0x7C   0x.... 0x....    BL     ??Subroutine8_0
\              ??CrossCallReturnLabel_22: (+1)
\       0x80   0x2003           MOVS          R0,#+3
\       0x82   0x2100           MOVS          R1,#+0
\       0x84   0x9001           STR    R0,[SP, #+4]
\       0x86   0x9102           STR    R1,[SP, #+8]
\       0x88   0x9103           STR    R1,[SP, #+12]
\       0x8A   0x4640           MOV    R0,R8
\       0x8C   0xA901           ADD    R1,SP,#+4
\       0x8E   0x.... 0x....    BL     HAL_GPIO_Init
\       0x92   0x210C           MOVS          R1,#+12
\       0x94   0x9101           STR    R1,[SP, #+4]
\       0x96   0x2201           MOVS          R2,#+1
\       0x98   0x2100           MOVS          R1,#+0
\       0x9A   0x9103           STR    R1,[SP, #+12]
\       0x9C   0x9104           STR    R1,[SP, #+16]
\       0x9E   0x9202           STR    R2,[SP, #+8]
\       0xA0   0xA901          ADD    R1,SP,#+4
\       0xA2   0x4640          MOV    R0,R8
\       0xA4   0x.... 0x....   BL     HAL_GPIO_Init
        99              MX_TIM4_Init();
\       0xA8   0x.... 0x....   BL     ?Subroutine1
\              ??CrossCallReturnLabel_2: (+1)
\       0xAC   0x2210          MOVS          R2,#+16
\       0xAE   0x.... 0x....   BL     ??Subroutine2_0
\              ??CrossCallReturnLabel_6: (+1)
\       0xB2   0x.... 0x....   LDR.W         R0,??DataTable1_12
\       0xB6   0x6030          STR    R0,[R6, #+0]
\       0xB8   0x2100          MOVS          R1,#+0
\       0xBA   0x6071          STR    R1,[R6, #+4]
\       0xBC   0x60B1          STR    R1,[R6, #+8]
\       0xBE   0x6131          STR    R1,[R6, #+16]
\       0xC0   0x61B1          STR    R1,[R6, #+24]
\       0xC2   0xF8C6 0x900C          STR    R9,[R6, #+12]
\       0xC6   0x4630          MOV    R0,R6
\       0xC8   0x.... 0x....   BL     HAL_TIM_IC_Init
\       0xCC   0xB108          CBZ.N R0,??main_0
\       0xCE   0x.... 0x....   BL     Error_Handler
\              ??main_0: (+1)
\       0xD2   0x.... 0x....   BL     ?Subroutine6
\              ??CrossCallReturnLabel_16: (+1)
\       0xD6   0x4630          MOV    R0,R6
\       0xD8   0x.... 0x....   BL     ?Subroutine9
\              ??CrossCallReturnLabel_28: (+1)
\       0xDC   0xB108          CBZ.N R0,??main_1
\       0xDE   0x.... 0x....   BL     Error_Handler
\              ??main_1: (+1)
\       0xE2   0x210A          MOVS          R1,#+10
\       0xE4   0x9102          STR    R1,[SP, #+8]
\       0xE6   0x2201          MOVS          R2,#+1
\       0xE8   0x9203          STR    R2,[SP, #+12]
\       0xEA   0x2100          MOVS          R1,#+0
\       0xEC   0x9104          STR    R1,[SP, #+16]
\       0xEE   0x2200          MOVS          R2,#+0
\       0xF0   0x9205          STR    R2,[SP, #+20]
\       0xF2   0xA902          ADD    R1,SP,#+8
\       0xF4   0x4630          MOV    R0,R6
```

```
\	0xF6	0x.... 0x....	BL	HAL_TIM_IC_ConfigChannel
\	0xFA	0xB108	CBZ.N	R0,??main_2
\	0xFC	0x.... 0x....	BL	Error_Handler
	100		MX_TIM1_Init();
\		??main_2: (+1)
\	0x100	0x.... 0x....	BL	?Subroutine1
\		??CrossCallReturnLabel_3: (+1)
\	0x104	0x.... 0x....	BL	?Subroutine7
\		??CrossCallReturnLabel_20: (+1)
\	0x108	0x.... 0x....	BL	?Subroutine2
\		??CrossCallReturnLabel_7: (+1)
\	0x10C	0x2032	MOVS	R0,#+50
\	0x10E	0x60A8	STR	R0,[R5, #+8]
\	0x110	0x2100	MOVS	R1,#+0
\	0x112	0x20C8	MOVS	R0,#+200
\	0x114	0x6128	STR	R0,[R5, #+16]
\	0x116	0x60E9	STR	R1,[R5, #+12]
\	0x118	0x6169	STR	R1,[R5, #+20]
\	0x11A	0x61A9	STR	R1,[R5, #+24]
\	0x11C	0x61E9	STR	R1,[R5, #+28]
\	0x11E	0x1D28	ADDS	R0,R5,#+4
\	0x120	0x.... 0x....	LDR.W	R10,??DataTable1_13
\	0x124	0xF8C5 0xA004	STR	R10,[R5, #+4]
\	0x128	0x.... 0x....	BL	HAL_TIM_PWM_Init
\	0x12C	0xB108	CBZ.N	R0,??main_3
\	0x12E	0x.... 0x....	BL	Error_Handler
\		??main_3: (+1)
\	0x132	0x.... 0x....	BL	?Subroutine6
\		??CrossCallReturnLabel_17: (+1)
\	0x136	0x1D28	ADDS	R0,R5,#+4
\	0x138	0x.... 0x....	BL	?Subroutine9
\		??CrossCallReturnLabel_27: (+1)
\	0x13C	0xB108	CBZ.N	R0,??main_4
\	0x13E	0x.... 0x....	BL	Error_Handler
\		??main_4: (+1)
\	0x142	0x.... 0x....	BL	?Subroutine0
\		??CrossCallReturnLabel_0: (+1)
\	0x146	0x1D28	ADDS	R0,R5,#+4
\	0x148	0x.... 0x....	BL	HAL_TIM_PWM_ConfigChannel
\	0x14C	0xB108	CBZ.N	R0,??main_5
```

```
\       0x14E   0x.... 0x....     BL      Error_Handler
\               ??main_5: (+1)
\       0x152   0x2000            MOVS            R0,#+0
\       0x154   0x9002            STR     R0,[SP, #+8]
\       0x156   0x9003            STR     R0,[SP, #+12]
\       0x158   0x9004            STR     R0,[SP, #+16]
\       0x15A   0x9005            STR     R0,[SP, #+20]
\       0x15C   0x9006            STR     R0,[SP, #+24]
\       0x15E   0xF44F 0x5700             MOV     R7,#+8192
\       0x162   0x9707            STR     R7,[SP, #+28]
\       0x164   0x9009            STR     R0,[SP, #+36]
\       0x166   0xA902            ADD     R1,SP,#+8
\       0x168   0x1D28            ADDS R0,R5,#+4
\       0x16A   0x.... 0x....     BL      HAL_TIMEx_ConfigBreakDeadTime
\       0x16E   0xB108            CBZ.N R0,??main_6
\       0x170   0x.... 0x....     BL      Error_Handler
\               ??main_6: (+1)
\       0x174   0x1D28            ADDS R0,R5,#+4
\       0x176   0x.... 0x....     BL      HAL_TIM_MspPostInit
        101             MX_TIM2_Init();
\       0x17A   0x.... 0x....     BL      ?Subroutine1
\               ??CrossCallReturnLabel_4: (+1)
\       0x17E   0x221C            MOVS            R2,#+28
\       0x180   0x.... 0x....     BL      ??Subroutine2_0
\               ??CrossCallReturnLabel_8: (+1)
\       0x184   0xF04F 0x4080             MOV     R0,#+1073741824
\       0x188   0x64E8            STR     R0,[R5, #+76]
\       0x18A   0x2132            MOVS            R1,#+50
\       0x18C   0x6529            STR     R1,[R5, #+80]
\       0x18E   0x2000            MOVS            R0,#+0
\       0x190   0x6568            STR     R0,[R5, #+84]
\       0x192   0x65E8            STR     R0,[R5, #+92]
\       0x194   0x6668            STR     R0,[R5, #+100]
\       0x196   0xF105 0x004C             ADD     R0,R5,#+76
\       0x19A   0x.... 0x....     LDR.W           R1,??DataTable1_14
\       0x19E   0x65A9            STR     R1,[R5, #+88]
\       0x1A0   0x.... 0x....     BL      HAL_TIM_OC_Init
\       0x1A4   0xB108            CBZ.N R0,??main_7
\       0x1A6   0x.... 0x....     BL      Error_Handler
\               ??main_7: (+1)
\   0x1AA   0x.... 0x....   BL      ?Subroutine6
\           ??CrossCallReturnLabel_18: (+1)
\   0x1AE   0xF105 0x004C   ADD     R0,R5,#+76
\   0x1B2   0x.... 0x....   BL      ?Subroutine9
\           ??CrossCallReturnLabel_26: (+1)
\   0x1B6   0xB108          CBZ.N   R0,??main_8
\   0x1B8   0x.... 0x....   BL      Error_Handler
\           ??main_8: (+1)
\   0x1BC   0x2100          MOVS    R1,#+0
\   0x1BE   0x9102          STR     R1,[SP, #+8]
\   0x1C0   0x2200          MOVS    R2,#+0
\   0x1C2   0x9104          STR     R1,[SP, #+16]
\   0x1C4   0x9203          STR     R2,[SP, #+12]
\   0x1C6   0x9206          STR     R2,[SP, #+24]
\   0x1C8   0xA902          ADD     R1,SP,#+8
\   0x1CA   0xF105 0x004C   ADD     R0,R5,#+76
\   0x1CE   0x.... 0x....   BL      HAL_TIM_OC_ConfigChannel
\   0x1D2   0xB108          CBZ.N   R0,??main_9
\   0x1D4   0x.... 0x....   BL      Error_Handler
    102             MX_TIM8_Init();
\           ??main_9: (+1)
\   0x1D8   0x.... 0x....   BL      ?Subroutine1
\           ??CrossCallReturnLabel_5: (+1)
\   0x1DC   0x.... 0x....   BL      ?Subroutine7
\           ??CrossCallReturnLabel_21: (+1)
\   0x1E0   0x.... 0x....   BL      ?Subroutine2
\           ??CrossCallReturnLabel_9: (+1)
\   0x1E4   0x....          LDR.N   R4,??DataTable1_15
\   0x1E6   0x64B4          STR     R4,[R6, #+72]
\   0x1E8   0x2000          MOVS    R0,#+0
\   0x1EA   0xF8C6 0x9054   STR     R9,[R6, #+84]
\   0x1EE   0x64F0          STR     R0,[R6, #+76]
\   0x1F0   0x6530          STR     R0,[R6, #+80]
\   0x1F2   0x65B0          STR     R0,[R6, #+88]
\   0x1F4   0x65F0          STR     R0,[R6, #+92]
\   0x1F6   0x6630          STR     R0,[R6, #+96]
\   0x1F8   0xF105 0x09DC   ADD     R9,R5,#+220
\   0x1FC   0x4648          MOV     R0,R9
\   0x1FE   0x.... 0x....   BL      HAL_TIM_PWM_Init
\   0x202   0xB108          CBZ.N   R0,??main_10
\       0x204   0x.... 0x....      BL      Error_Handler
\               ??main_10: (+1)
\       0x208   0x.... 0x....      BL      ?Subroutine6
\               ??CrossCallReturnLabel_19: (+1)
\       0x20C   0x4648             MOV     R0,R9
\       0x20E   0x.... 0x....      BL      ?Subroutine9
\               ??CrossCallReturnLabel_25: (+1)
\       0x212   0xB108             CBZ.N   R0,??main_11
\       0x214   0x.... 0x....      BL      Error_Handler
\               ??main_11: (+1)
\       0x218   0x.... 0x....      BL      ?Subroutine0
\               ??CrossCallReturnLabel_1: (+1)
\       0x21C   0x4648             MOV     R0,R9
\       0x21E   0x.... 0x....      BL      HAL_TIM_PWM_ConfigChannel
\       0x222   0xB108             CBZ.N   R0,??main_12
\       0x224   0x.... 0x....      BL      Error_Handler
\               ??main_12: (+1)
\       0x228   0x2100             MOVS            R1,#+0
\       0x22A   0x9102             STR     R1,[SP, #+8]
\       0x22C   0x9103             STR     R1,[SP, #+12]
\       0x22E   0x9104             STR     R1,[SP, #+16]
\       0x230   0x9105             STR     R1,[SP, #+20]
\       0x232   0x9106             STR     R1,[SP, #+24]
\       0x234   0x9707             STR     R7,[SP, #+28]
\       0x236   0x9109             STR     R1,[SP, #+36]
\       0x238   0x4648             MOV     R0,R9
\       0x23A   0xA902             ADD     R1,SP,#+8
\       0x23C   0x.... 0x....      BL      HAL_TIMEx_ConfigBreakDeadTime
\       0x240   0xB108             CBZ.N   R0,??main_13
\       0x242   0x.... 0x....      BL      Error_Handler
\               ??main_13: (+1)
\       0x246   0x4648             MOV     R0,R9
\       0x248   0x.... 0x....      BL      HAL_TIM_MspPostInit
        103            /* USER CODE BEGIN 2 */
        104            //TIM1 -> PSC = 49 - 1; // set the prescalar for the PWM generation
        105            TIM1 -> ARR = 100; // 1MHz / 5kHz = 100 ticks/5kHz wave
\       0x24C   0x2064             MOVS            R0,#+100
\       0x24E   0xF8CA 0x002C      STR     R0,[R10, #+44]
        106            TIM1 -> CCR1 = 50; // 200/2 = 100 = 50% duty cycle
\       0x252   0x2132             MOVS            R1,#+50
```

```
  \     0x254   0xF8CA 0x1034       STR     R1,[R10, #+52]
        107                 HAL_TIM_PWM_Init(&htim1);
  \     0x258   0x1D28          ADDS R0,R5,#+4
  \     0x25A   0x.... 0x....    BL      HAL_TIM_PWM_Init
        108
        109                 TIM8 -> ARR = 1667; // 100MHz/30kHz = 1667 ticks/30kHz wave
  \     0x25E   0xF240 0x6083       MOVW          R0,#+1667
  \     0x262   0x62E0          STR     R0,[R4, #+44]
        110                 TIM8 -> CCR1 = 1667; // 1667 * .5 = 1667 for 50% duty cycle
  \     0x264   0x4601          MOV   R1,R0
  \     0x266   0x6361          STR     R1,[R4, #+52]
        111                 HAL_TIM_PWM_Init(&htim8);
  \     0x268   0x4648          MOV   R0,R9
  \     0x26A   0x.... 0x....    BL      HAL_TIM_PWM_Init
  \     0x26E   0xF505 0x7792       ADD   R7,R5,#+292
  \     0x272   0xF04F 0x4B80       MOV   R11,#+1073741824
  \     0x276   0xE002          B.N     ??main_14
        112                 /* USER CODE END 2 */
        113
        114                 /* Infinite loop */
        115                 /* USER CODE BEGIN WHILE */
        116                 while (1)
        117                 { START = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_0); // start = PD0
        118                 STOP = HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_1); // start = PD1
        119                 HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1); // turn off the
buzzer
        120
        121                 if (START == GPIO_PIN_SET && state == 0) {
  \             ??main_15: (+1)
  \     0x278   0x2800          CMP   R0,#+0
  \     0x27A   0xD142          BNE.N??main_16
        122                 blinks = 0; // reset the blink count for reset
  \     0x27C   0x6078          STR     R0,[R7, #+4]
        123                 }
  \             ??main_14: (+1)
  \     0x27E   0x2101          MOVS          R1,#+1
  \     0x280   0x4640          MOV   R0,R8
  \     0x282   0x.... 0x....    BL      HAL_GPIO_ReadPin
  \     0x286   0x7028          STRB  R0,[R5, #+0]
  \     0x288   0x2102          MOVS          R1,#+2
  \     0x28A   0x4640          MOV   R0,R8
  \     0x28C   0x.... 0x....   BL    HAL_GPIO_ReadPin
  \     0x290   0x7068          STRB  R0,[R5, #+1]
  \     0x292   0x2100          MOVS          R1,#+0
  \     0x294   0x1D28          ADDS  R0,R5,#+4
  \     0x296   0x.... 0x....   BL    HAL_TIM_PWM_Stop
  \     0x29A   0x7829          LDRB  R1,[R5, #+0]
  \     0x29C   0x6838          LDR   R0,[R7, #+0]
  \     0x29E   0x2901          CMP   R1,#+1
  \     0x2A0   0xD0EA                BEQ.N??main_15
        124
        125
        126             // the start sequence
        127             else if (((START == GPIO_PIN_RESET) && (state == 0)) && (blinks <
7)){
  \     0x2A2   0x2900          CMP   R1,#+0
  \     0x2A4   0xBF08          IT    EQ
  \     0x2A6   0x2800          CMPEQ         R0,#+0
  \     0x2A8   0xD12B                BNE.N??main_16
  \     0x2AA   0x6878          LDR   R0,[R7, #+4]
  \     0x2AC   0x2806          CMP   R0,#+6
  \     0x2AE   0xDCE6                BGT.N ??main_14
        128             TIM2 -> ARR = 500000; // the timer will trigger every half second
  \     0x2B0   0x....          LDR.NR1,??DataTable1_14
  \     0x2B2   0xF8CB 0x102C       STR     R1,[R11, #+44]
        129             TIM1 -> PSC = 50-1;
  \     0x2B6   0x2031          MOVS          R0,#+49
  \     0x2B8   0xF8CA 0x0028       STR     R0,[R10, #+40]
        130             TIM1 -> ARR = 100; // 1MHz / 5kHz = 100 ticks/5kHz wave
  \     0x2BC   0x2164          MOVS          R1,#+100
  \     0x2BE   0xF8CA 0x102C       STR     R1,[R10, #+44]
        131             TIM1 -> CCR1 = 50; // 100/2 = 50 = 50% duty cycle
  \     0x2C2   0x2032          MOVS          R0,#+50
  \     0x2C4   0xF8CA 0x0034       STR     R0,[R10, #+52]
        132             HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1); // start the
blinking
  \     0x2C8   0x.... 0x....   BL    ?Subroutine4
        133             HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // start the buzzer
  \             ??CrossCallReturnLabel_13: (+1)
  \     0x2CC   0x2100          MOVS          R1,#+0
  \     0x2CE   0x1D28                  ADDS R0,R5,#+4
  \     0x2D0   0x.... 0x....   BL      HAL_TIM_PWM_Start
        134             if (blinks == 6){
  \     0x2D4   0x6878          LDR     R0,[R7, #+4]
  \     0x2D6   0x2806          CMP     R0,#+6
  \     0x2D8   0xD1D1                  BNE.N??main_14
        135             HAL_TIM_OC_Stop_IT(&htim2, TIM_CHANNEL_1); // stop the blink
interrupt
  \     0x2DA   0x.... 0x....   BL      ?Subroutine5
        136             TIM8 -> ARR = 1667;
  \             ??CrossCallReturnLabel_15: (+1)
  \     0x2DE   0xF240 0x6083           MOVW            R0,#+1667
  \     0x2E2   0x62E0          STR     R0,[R4, #+44]
        137             TIM8 -> CCR1 = 834;
  \     0x2E4   0xF240 0x3142           MOVW            R1,#+834
  \     0x2E8   0x6361          STR     R1,[R4, #+52]
        138             HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1); // turn on the
output
  \     0x2EA   0x4648          MOV     R0,R9
  \     0x2EC   0x2100          MOVS            R1,#+0
  \     0x2EE   0x.... 0x....   BL      HAL_TIM_PWM_Start
        139             HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1); // turn on the
input compare
  \     0x2F2   0x2100          MOVS            R1,#+0
  \     0x2F4   0x4630          MOV     R0,R6
  \     0x2F6   0x.... 0x....   BL      HAL_TIM_IC_Start_IT
        140             state++; // goto next state after 6 blinks
  \     0x2FA   0x6838          LDR     R0,[R7, #+0]
  \     0x2FC   0x1C40          ADDS R0,R0,#+1
  \             ??main_17: (+1)
  \     0x2FE   0x6038          STR     R0,[R7, #+0]
  \     0x300   0xE7BD                  B.N     ??main_14
        141             }
        142             }
        143
        144             // the PWM generation
        145             else if (state == 1 && STOP == 1){
  \             ??main_16: (+1)
  \     0x302   0x7869          LDRB R1,[R5, #+1]
  \     0x304   0x2801          CMP     R0,#+1
  \     0x306   0xBF08          IT        EQ
  \     0x308   0x2901          CMPEQ           R1,#+1
  \     0x30A   0xD156          BNE.N??main_18
        146               blinks = 0; // reset the blink counter
  \     0x30C   0x2000          MOVS            R0,#+0
  \     0x30E   0x6078          STR     R0,[R7, #+4]
        147               HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1); // turn off the
buzzer
  \     0x310   0x2100          MOVS            R1,#+0
  \     0x312   0x1D28          ADDS R0,R5,#+4
  \     0x314   0x.... 0x....   BL      HAL_TIM_PWM_Stop
        148               HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET); // turn
on the IR LED
  \     0x318   0x2201          MOVS            R2,#+1
  \     0x31A   0x.... 0x....   BL      ?Subroutine8
        149               if (freq > 6300){
  \             ??CrossCallReturnLabel_24: (+1)
  \     0x31E   0xED97 0x0A02           VLDR S0,[R7, #+8]
  \     0x322   0xEDDF 0x....           VLDR.W   S1,??DataTable1
  \     0x326   0x.... 0x....   BL      ?Subroutine10
  \             ??CrossCallReturnLabel_29: (+1)
  \     0x32A   0xDB02                  BLT.N ??main_19
        150               TIM8 -> CCR1 = 167; // 1667/10 = 167 for 10% duty cycle
  \     0x32C   0x20A7          MOVS            R0,#+167
  \             ??main_20: (+1)
  \     0x32E   0x6360          STR     R0,[R4, #+52]
  \     0x330   0xE7A5          B.N     ??main_14
        151               }
        152               else if (freq > 6100){
  \             ??main_19: (+1)
  \     0x332   0xEDDF 0x....           VLDR.W   S1,??DataTable1_1
  \     0x336   0x.... 0x....   BL      ?Subroutine10
  \             ??CrossCallReturnLabel_30: (+1)
  \     0x33A   0xBFA8                  IT        GE
  \     0x33C   0xF44F 0x70A7           MOVGE         R0,#+334
        153               TIM8 -> CCR1 = 334; // 1667/5 = 334 for 20% duty cycle
  \     0x340   0xDAF5                  BGE.N??main_20
        154               }
        155               else if (freq > 5900){
  \     0x342   0xEDDF 0x....           VLDR.W   S1,??DataTable1_2
  \     0x346   0x.... 0x....     BL       ?Subroutine10
  \               ??CrossCallReturnLabel_31: (+1)
  \     0x34A   0xBFA8            IT       GE
  \     0x34C   0xF44F 0x70FA     MOVGE        R0,#+500
        156            TIM8 -> CCR1 = 500; // 1667 * .3 = 500 for 30% duty cycle
  \     0x350   0xDAED            BGE.N??main_20
        157            }
        158            else if (freq > 5700){
  \     0x352   0xEDDF 0x....     VLDR.W   S1,??DataTable1_3
  \     0x356   0x.... 0x....     BL       ?Subroutine10
  \               ??CrossCallReturnLabel_32: (+1)
  \     0x35A   0xBFA8            IT       GE
  \     0x35C   0xF240 0x209B     MOVWGE   R0,#+667
        159            TIM8 -> CCR1 = 667; // 1667 * .4 = 667 for 40% duty cycle
  \     0x360   0xDAE5            BGE.N??main_20
        160            }
        161            else if (freq > 5500){
  \     0x362   0xEDDF 0x....     VLDR.W   S1,??DataTable1_4
  \     0x366   0x.... 0x....     BL       ?Subroutine10
  \               ??CrossCallReturnLabel_33: (+1)
  \     0x36A   0xBFA8            IT       GE
  \     0x36C   0xF240 0x3042     MOVWGE   R0,#+834
        162            TIM8 -> CCR1 = 834; // 1667 * .5 = 834 for 50% duty cycle
  \     0x370   0xDADD            BGE.N??main_20
        163            }
        164            else if (freq > 5300){
  \     0x372   0xEDDF 0x....     VLDR.W   S1,??DataTable1_5
  \     0x376   0x.... 0x....     BL       ?Subroutine10
  \               ??CrossCallReturnLabel_34: (+1)
  \     0x37A   0xBFA8            IT       GE
  \     0x37C   0xF44F 0x707A     MOVGE        R0,#+1000
        165            TIM8 -> CCR1 = 1000; // 1667 * .6 = 1000 for 60% duty cycle
  \     0x380   0xDAD5            BGE.N??main_20
        166            }
        167            else if (freq > 5100){
  \     0x382   0xEDDF 0x....     VLDR.W   S1,??DataTable1_6
  \     0x386   0x.... 0x....     BL       ?Subroutine10
  \               ??CrossCallReturnLabel_35: (+1)
  \     0x38A   0xBFA8            IT       GE
  \     0x38C   0xF240 0x408F     MOVWGE   R0,#+1167
```

```
        168             TIM8 -> CCR1 = 1167; // 1667 * .7 = 1167 for 70% duty cycle
  \     0x390   0xDACD              BGE.N??main_20
        169             }
        170             else if (freq > 4900){
  \     0x392   0xEDDF 0x....       VLDR.W   S1,??DataTable1_7
  \     0x396   0x.... 0x....   BL      ?Subroutine10
  \             ??CrossCallReturnLabel_36: (+1)
  \     0x39A   0xBFA8              IT       GE
  \     0x39C   0xF240 0x5036       MOVWGE   R0,#+1334
        171             TIM8 -> CCR1 = 1334; // 1667 * .8 = 1334 for 80% duty cycle
  \     0x3A0   0xDAC5              BGE.N??main_20
        172             }
        173             else if (freq > 4700){
  \     0x3A2   0xEDDF 0x....       VLDR.W   S1,??DataTable1_8
  \     0x3A6   0x.... 0x....   BL      ?Subroutine10
  \             ??CrossCallReturnLabel_37: (+1)
  \     0x3AA   0xBFA8              IT       GE
  \     0x3AC   0xF240 0x50DC       MOVWGE   R0,#+1500
        174             TIM8 -> CCR1 = 1500; // 1667 * .9 = 1500 for 90% duty cycle
  \     0x3B0   0xDABD              BGE.N??main_20
        175             }
        176             else {
        177             TIM8 -> CCR1 = 834; // 1667 * .5 = 834 for 50% duty cycle
  \     0x3B2   0xF240 0x3142       MOVW        R1,#+834
  \     0x3B6   0x6361          STR     R1,[R4, #+52]
  \             ??main_21: (+1)
  \     0x3B8   0xE761          B.N     ??main_14
        178             // shouldn't get here, just in case
        179             }
        180             }
        181
        182             // the stop sequence
        183             else if (STOP == 0 && state == 1) {
  \             ??main_18: (+1)
  \     0x3BA   0x2900          CMP     R1,#+0
  \     0x3BC   0xBF08              IT       EQ
  \     0x3BE   0x2801          CMPEQ       R0,#+1
  \     0x3C0   0xD1FA              BNE.N??main_21
        184             HAL_TIM_OC_Stop_IT(&htim4, TIM_CHANNEL_1); // stop the input
capture
```

```
  \     0x3C2   0x4630          MOV   R0,R6
  \     0x3C4   0x.... 0x....   BL    HAL_TIM_OC_Stop_IT
        185             HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_1); // stop the output
  \     0x3C8   0x2100          MOVS          R1,#+0
  \     0x3CA   0x4648          MOV   R0,R9
  \     0x3CC   0x.... 0x....   BL    HAL_TIM_PWM_Stop
        186             HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET); //
turn off the IR LED
  \     0x3D0   0x2200          MOVS          R2,#+0
  \     0x3D2   0x.... 0x....   BL    ?Subroutine8
        187             TIM1 -> ARR = 90;
  \             ??CrossCallReturnLabel_23: (+1)
  \     0x3D6   0x205A          MOVS          R0,#+90
  \     0x3D8   0xF8CA 0x002C   STR   R0,[R10, #+44]
        188             TIM1 -> CCR1 = 45;
  \     0x3DC   0x212D                  MOVS          R1,#+45
  \     0x3DE   0xF8CA 0x1034   STR   R1,[R10, #+52]
        189             HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // turn on the
buzzer
  \     0x3E2   0x1D28          ADDS R0,R5,#+4
  \     0x3E4   0x2100          MOVS          R1,#+0
  \     0x3E6   0x.... 0x....   BL    HAL_TIM_PWM_Start
        190             TIM2 -> ARR = 250000; // the timer will trigger every quarter second
  \     0x3EA   0x....          LDR.NR0,??DataTable1_16
  \     0x3EC   0xF8CB 0x002C   STR   R0,[R11, #+44]
        191             HAL_TIM_OC_Start_IT(&htim2, TIM_CHANNEL_1); // start blinking
the light
  \     0x3F0   0x.... 0x....   BL    ?Subroutine4
        192
        193
        194             if (blinks == 10){
  \             ??CrossCallReturnLabel_12: (+1)
  \     0x3F4   0x6878          LDR   R0,[R7, #+4]
  \     0x3F6   0x280A          CMP   R0,#+10
  \     0x3F8   0xD1DE                  BNE.N??main_21
        195             HAL_TIM_OC_Stop_IT(&htim2, TIM_CHANNEL_1); // stop the
blinking at 10 blinks
  \     0x3FA   0x.... 0x....   BL    ?Subroutine5
        196             state = 0; // restart the process
  \             ??CrossCallReturnLabel_14: (+1)
\       0x3FE   0x2000          MOVS            R0,#+0
\       0x400   0xE77D          B.N     ??main_17
        197             }
        198             }
        199
        200             /* USER CODE END WHILE */
        201
        202             /* USER CODE BEGIN 3 */
        203             }
        204             /* USER CODE END 3 */
        205             }

\                       In section .text, align 2, keep-with-next
\               ?Subroutine10: (+1)
\       0x0   0xEEB4 0x0A60        VCMP.F32 S0,S1
\       0x4   0xEEF1 0xFA10        FMSTAT
\       0x8   0x4770          BX      LR

\                       In section .text, align 2, keep-with-next
\               ?Subroutine8: (+1)
\       0x0   0x2104          MOVS            R1,#+4
\               ??Subroutine8_0: (+1)
\       0x2   0x4640          MOV     R0,R8
\       0x4   0x.... 0x....   B.W     HAL_GPIO_WritePin

\                       In section .text, align 2, keep-with-next
\               ?Subroutine5: (+1)
\       0x0   0x2100          MOVS            R1,#+0
\       0x2   0xF105 0x004C        ADD     R0,R5,#+76
\       0x6   0x.... 0x....   B.W     HAL_TIM_OC_Stop_IT

\                       In section .text, align 2, keep-with-next
\               ?Subroutine4: (+1)
\       0x0   0x2100          MOVS            R1,#+0
\       0x2   0xF105 0x004C        ADD     R0,R5,#+76
\       0x6   0x.... 0x....   B.W     HAL_TIM_OC_Start_IT
        206
        207             /**
        208             * @brief System Clock Configuration
        209             * @retval None
        210          */

   \                 In section .text, align 2, keep-with-next
        211     void SystemClock_Config(void)
        212          {
   \            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    ?Subroutine3
        213          RCC_OscInitTypeDef RCC_OscInitStruct = {0};
        214          RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
        215
        216          /** Configure the main internal regulator output voltage
        217          */
        218          __HAL_RCC_PWR_CLK_ENABLE();
   \            ??CrossCallReturnLabel_11: (+1)
   \    0x12   0x2000      MOVS        R0,#+0
   \    0x14   0x9000      STR   R0,[SP, #+0]
        219
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);
        220
        221          /** Initializes the RCC Oscillators according to the specified parameters
        222          * in the RCC_OscInitTypeDef structure.
        223          */
        224          RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
        225          RCC_OscInitStruct.HSEState = RCC_HSE_ON;
   \    0x16   0xF44F 0x3380     MOV   R3,#+65536
   \    0x1A   0x....      LDR.NR0,??DataTable1_17
   \    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.NR0,??DataTable1_18
\       0x32   0x9100            STR    R1,[SP, #+0]
\       0x34   0x2101            MOVS           R1,#+1
\       0x36   0x6802            LDR    R2,[R0, #+0]
\       0x38   0xF361 0x328F             BFI    R2,R1,#+14,#+2
\       0x3C   0x6002            STR    R2,[R0, #+0]
\       0x3E   0x2201            MOVS           R2,#+1
\       0x40   0x6800            LDR    R0,[R0, #+0]
\       0x42   0xF400 0x4040             AND    R0,R0,#0xC000
\       0x46   0x9000            STR    R0,[SP, #+0]
        226             RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
        227             RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
        228             RCC_OscInitStruct.PLL.PLLM = 4;
        229             RCC_OscInitStruct.PLL.PLLN = 100;
        230             RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
        231             RCC_OscInitStruct.PLL.PLLQ = 4;
        232             if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
\       0x48   0xA806            ADD    R0,SP,#+24
\       0x4A   0x9900            LDR    R1,[SP, #+0]
\       0x4C   0x9206            STR    R2,[SP, #+24]
\       0x4E   0x2102            MOVS           R1,#+2
\       0x50   0xF44F 0x0280             MOV    R2,#+4194304
\       0x54   0x910C            STR    R1,[SP, #+48]
\       0x56   0x920D            STR    R2,[SP, #+52]
\       0x58   0x2104            MOVS           R1,#+4
\       0x5A   0x2264            MOVS           R2,#+100
\       0x5C   0x910E            STR    R1,[SP, #+56]
\       0x5E   0x920F            STR    R2,[SP, #+60]
\       0x60   0x2102            MOVS           R1,#+2
\       0x62   0x2204            MOVS           R2,#+4
\       0x64   0x9307            STR    R3,[SP, #+28]
\       0x66   0x9110            STR    R1,[SP, #+64]
\       0x68   0x9211            STR    R2,[SP, #+68]
\       0x6A   0x.... 0x....     BL     HAL_RCC_OscConfig
\       0x6E   0xB108            CBZ.NR0,??SystemClock_Config_0
        233             {
        234             Error_Handler();
\       0x70   0xB672            CPSIDI
\              ??SystemClock_Config_1: (+1)
\       0x72   0xE7FE            B.N    ??SystemClock_Config_1
```

```
        235             }
        236
        237             /** Initializes the CPU, AHB and APB buses clocks
        238             */
        239             RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
        240
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
        241             RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  \             ??SystemClock_Config_0: (+1)
  \     0x74  0x2102            MOVS        R1,#+2
  \     0x76  0x9102            STR    R1,[SP, #+8]
  \     0x78  0x200F            MOVS        R0,#+15
        242             RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
        243             RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  \     0x7A   0xF44F 0x51A0    MOV    R1,#+5120
  \     0x7E   0x9001           STR    R0,[SP, #+4]
  \     0x80  0x2200            MOVS        R2,#+0
  \     0x82  0x9104            STR    R1,[SP, #+16]
        244             RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;
  \     0x84  0x9105            STR    R1,[SP, #+20]
  \     0x86  0x9203            STR    R2,[SP, #+12]
        245
        246             if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
  \     0x88  0x2103            MOVS        R1,#+3
  \     0x8A   0xA801           ADD    R0,SP,#+4
  \     0x8C   0x.... 0x....    BL     HAL_RCC_ClockConfig
  \     0x90  0xB108            CBZ.N R0,??SystemClock_Config_2
        247             {
        248             Error_Handler();
  \     0x92  0xB672            CPSID I
  \             ??SystemClock_Config_3: (+1)
  \     0x94  0xE7FE            B.N    ??SystemClock_Config_3
        249             }
        250             }
  \             ??SystemClock_Config_2: (+1)
  \     0x96  0xB013            ADD    SP,SP,#+76
  \     0x98  0xBD00            POP    {PC}
  \                       In section .text, align 2, keep-with-next
  \               ?Subroutine3: (+1)
  \       0x0   0x2214            MOVS          R2,#+20
  \       0x2   0x2100            MOVS          R1,#+0
  \       0x4   0xA801            ADD   R0,SP,#+4
  \       0x6   0x.... 0x....     B.W     memset

                          /**
                          * @brief TIM1 Initialization Function
                          * @param None
                          * @retval None
                          */
                          static void MX_TIM1_Init(void)
                          {

                          /* USER CODE BEGIN TIM1_Init 0 */

                          /* USER CODE END TIM1_Init 0 */

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

                          /* USER CODE BEGIN TIM1_Init 1 */

                          /* USER CODE END TIM1_Init 1 */
                          htim1.Instance = TIM1;
                          htim1.Init.Prescaler = 50;
                          htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
                          htim1.Init.Period = 200;
                          htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
                          htim1.Init.RepetitionCounter = 0;
                          htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
                          if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
                          {
                          Error_Handler();
                          }
                          sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
  Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
  Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) != HAL_OK)
  {
  Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */
  HAL_TIM_MspPostInit(&htim1);

  }

```

```
/**
* @brief TIM2 Initialization Function
* @param None
* @retval None
*/
static void MX_TIM2_Init(void)
{

/* USER CODE BEGIN TIM2_Init 0 */

/* USER CODE END TIM2_Init 0 */

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

/* USER CODE BEGIN TIM2_Init 1 */

/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 50;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 500000;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_OC_Init(&htim2) != HAL_OK)
{
Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &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(&htim2, &sConfigOC,
TIM_CHANNEL_1) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */

/* USER CODE END TIM2_Init 2 */

}

/**
* @brief TIM4 Initialization Function
* @param None
* @retval None
*/
static void MX_TIM4_Init(void)
{

/* USER CODE BEGIN TIM4_Init 0 */

/* USER CODE END TIM4_Init 0 */

TIM_MasterConfigTypeDef sMasterConfig = {0};
TIM_IC_InitTypeDef sConfigIC = {0};

/* USER CODE BEGIN TIM4_Init 1 */

/* USER CODE END TIM4_Init 1 */
htim4.Instance = TIM4;
htim4.Init.Prescaler = 0;
htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
htim4.Init.Period = 65535;
htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim4.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_IC_Init(&htim4) != HAL_OK)
{
Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim4,
&sMasterConfig) != HAL_OK)
{
Error_Handler();
}
sConfigIC.ICPolarity =
TIM_INPUTCHANNELPOLARITY_BOTHEDGE;
sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
sConfigIC.ICFilter = 0;
if (HAL_TIM_IC_ConfigChannel(&htim4, &sConfigIC,
TIM_CHANNEL_1) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM4_Init 2 */

/* USER CODE END TIM4_Init 2 */

}

/**
* @brief TIM8 Initialization Function
* @param None
* @retval None
*/
static void MX_TIM8_Init(void)
{

/* USER CODE BEGIN TIM8_Init 0 */

/* USER CODE END TIM8_Init 0 */

TIM_MasterConfigTypeDef sMasterConfig = {0};
TIM_OC_InitTypeDef sConfigOC = {0};
TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};
```

```

/* USER CODE BEGIN TIM8_Init 1 */

/* USER CODE END TIM8_Init 1 */
htim8.Instance = TIM8;
htim8.Init.Prescaler = 0;
htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
htim8.Init.Period = 65535;
htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim8.Init.RepetitionCounter = 0;
htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_PWM_Init(&htim8) != HAL_OK)
{
Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL_OK)
{
Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
if (HAL_TIM_PWM_ConfigChannel(&htim8, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
Error_Handler();
}
sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
sBreakDeadTimeConfig.DeadTime = 0;
sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
if (HAL_TIMEx_ConfigBreakDeadTime(&htim8, &sBreakDeadTimeConfig) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM8_Init 2 */

/* USER CODE END TIM8_Init 2 */
HAL_TIM_MspPostInit(&htim8);

}

/**
* @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_GPIOE_CLK_ENABLE();
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOC_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOD, IR_LED_Pin|LED_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : Start_Pin Stop_Pin */
GPIO_InitStruct.Pin = Start_Pin|Stop_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
        HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

        /*Configure GPIO pins : IR_LED_Pin LED_Pin */
        GPIO_InitStruct.Pin = IR_LED_Pin|LED_Pin;
        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 */

        /* USER CODE END 4 */

        /**
        * @brief  This function is executed in case of error occurrence.
        * @retval None
        */

\       In section .text, align 2, keep-with-next
        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();
\   Error_Handler: (+1)
\   0x0  0xB672        CPSID I
        while (1)
\   ??Error_Handler_0: (+1)
\   0x2  0xE7FE        B.N    ??Error_Handler_0
        {
        }
        /* USER CODE END Error_Handler_Debug */
        }

\       In section .text, align 2, keep-with-next
\   ?Subroutine9: (+1)
\   0x0  0x4669        MOV   R1,SP
```

```
\	0x2	0x.... 0x....	B.W	HAL_TIMEx_MasterConfigSynchronization

\			In section .text, align 2, keep-with-next
\		?Subroutine7: (+1)
\	0x0	0x221C		MOVS		R2,#+28
\	0x2	0x2100		MOVS		R1,#+0
\	0x4	0xA80A		ADD	R0,SP,#+40
\	0x6	0x.... 0x....	B.W	memset

\			In section .text, align 2, keep-with-next
\		?Subroutine6: (+1)
\	0x0	0x2100		MOVS		R1,#+0
\	0x2	0x9100		STR	R1,[SP, #+0]
\	0x4	0x9101		STR	R1,[SP, #+4]
\	0x6	0x4770		BX	LR

\			In section .text, align 2, keep-with-next
\		?Subroutine2: (+1)
\	0x0	0x2220		MOVS		R2,#+32
\		??Subroutine2_0: (+1)
\	0x2	0x2100		MOVS		R1,#+0
\	0x4	0xA802		ADD	R0,SP,#+8
\	0x6	0x.... 0x....	B.W	memset

\			In section .text, align 2, keep-with-next
\		?Subroutine1: (+1)
\	0x0	0x2208		MOVS		R2,#+8
\	0x2	0x2100		MOVS		R1,#+0
\	0x4	0x4668		MOV	R0,SP
\	0x6	0x.... 0x....	B.W	memset

\			In section .text, align 2, keep-with-next
\		?Subroutine0: (+1)
\	0x0	0x2060		MOVS		R0,#+96
\	0x2	0x2100		MOVS		R1,#+0
\	0x4	0x900A		STR	R0,[SP, #+40]
\	0x6	0x910B		STR	R1,[SP, #+44]
\	0x8	0x2200		MOVS		R2,#+0
\	0xA	0x910D		STR	R1,[SP, #+52]
\	0xC	0x910F		STR	R1,[SP, #+60]
```

```
\       0xE   0x920C            STR     R2,[SP, #+48]
\       0x10   0x920E           STR     R2,[SP, #+56]
\       0x12   0x9210           STR     R2,[SP, #+64]
\       0x14   0xA90A           ADD     R1,SP,#+40
\       0x16   0x4770           BX      LR

\                          In section .text, align 4, keep-with-next
\               ??DataTable1:
\       0x0   0x45C4'E001     DC32   0x45c4e001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_1:
\       0x0   0x45BE'A001     DC32   0x45bea001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_2:
\       0x0   0x45B8'6001     DC32   0x45b86001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_3:
\       0x0   0x45B2'2001     DC32   0x45b22001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_4:
\       0x0   0x45AB'E001     DC32   0x45abe001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_5:
\       0x0   0x45A5'A001     DC32   0x45a5a001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_6:
\       0x0   0x459F'6001     DC32   0x459f6001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_7:
\       0x0   0x4599'2001     DC32   0x45992001

\                          In section .text, align 4, keep-with-next
\               ??DataTable1_8:
```

```
\       0x0   0x4592'E001      DC32   0x4592e001

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_9:
\       0x0   0x4002'3830      DC32   0x40023830

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_10:
\       0x0   0x4002'0C00      DC32   0x40020c00

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_11:
\       0x0   0x....'....      DC32   START

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_12:
\       0x0   0x4000'0800      DC32   0x40000800

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_13:
\       0x0   0x4001'0000      DC32   0x40010000

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_14:
\       0x0   0x0007'A120      DC32   0x7a120

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_15:
\       0x0   0x4001'0400      DC32   0x40010400

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_16:
\       0x0   0x0003'D090      DC32   0x3d090

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_17:
\       0x0   0x4002'3840      DC32   0x40023840

\                        In section .text, align 4, keep-with-next
\               ??DataTable1_18:
```

```
\       0x0   0x4000'7000      DC32   0x40007000
        529
        530              #ifdef  USE_FULL_ASSERT
        531              /**
        532              * @brief  Reports the name of the source file and the source line number
        533              *         where the assert_param error has occurred.
        534              * @param  file: pointer to the source file name
        535              * @param  line: assert_param error line source number
        536              * @retval None
        537              */
        538              void assert_failed(uint8_t *file, uint32_t line)
        539              {
        540              /* USER CODE BEGIN 6 */
        541              /* User can add his own implementation to report the file name and line
number,
        542              ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
        543              /* USER CODE END 6 */
        544              }
        545              #endif /* USE_FULL_ASSERT */

  Maximum stack usage in bytes:

  .cstack Function
  ------- --------
        0   Error_Handler
        80   SystemClock_Config
        80   -> HAL_RCC_ClockConfig
        80   -> HAL_RCC_OscConfig
        80   -> memset
        104   main
        104   -> Error_Handler
        104   -> HAL_GPIO_Init
        104   -> HAL_GPIO_ReadPin
        104   -> HAL_GPIO_WritePin
        104   -> HAL_Init
        104   -> HAL_TIMEx_ConfigBreakDeadTime
        104   -> HAL_TIMEx_MasterConfigSynchronization
        104   -> HAL_TIM_IC_ConfigChannel
        104   -> HAL_TIM_IC_Init
        104   -> HAL_TIM_IC_Start_IT
```

```
     104   -> HAL_TIM_MspPostInit
     104   -> HAL_TIM_OC_ConfigChannel
     104   -> HAL_TIM_OC_Init
     104   -> HAL_TIM_OC_Start_IT
     104   -> HAL_TIM_OC_Stop_IT
     104   -> HAL_TIM_PWM_ConfigChannel
     104   -> HAL_TIM_PWM_Init
     104   -> HAL_TIM_PWM_Start
     104   -> HAL_TIM_PWM_Stop
     104   -> SystemClock_Config
     104   -> memset


Section sizes:

Bytes  Function/Label
-----  --------------
       4  ??DataTable1
       4  ??DataTable1_1
       4  ??DataTable1_10
       4  ??DataTable1_11
       4  ??DataTable1_12
       4  ??DataTable1_13
       4  ??DataTable1_14
       4  ??DataTable1_15
       4  ??DataTable1_16
       4  ??DataTable1_17
       4  ??DataTable1_18
       4  ??DataTable1_2
       4  ??DataTable1_3
       4  ??DataTable1_4
       4  ??DataTable1_5
       4  ??DataTable1_6
       4  ??DataTable1_7
       4  ??DataTable1_8
       4  ??DataTable1_9
      24  ?Subroutine0
      10  ?Subroutine1
      10  ?Subroutine10
      10  ?Subroutine2
```

```
         10  ?Subroutine3
         10  ?Subroutine4
         10  ?Subroutine5
         8  ?Subroutine6
         10  ?Subroutine7
         8  ?Subroutine8
         6  ?Subroutine9
         4  Error_Handler
         304  START
         STOP
         htim1
         htim2
         htim4
         htim8
         state
         blinks
         freq
         154  SystemClock_Config
   1'026  main


   304 bytes in section .data
 1'376 bytes in section .text

 1'376 bytes of CODE memory
   304 bytes of DATA memory

Errors: none
Warnings: none
```

It.lst

```
###############################################################################
#
#
# IAR ANSI C/C++ Compiler V9.20.4.327/W64 for ARM            12/Apr/2023  18:28:46
# Copyright 1999-2022 IAR Systems AB.
#
```

```
#	Cpu mode	=  thumb
#	Endian		=  little
#	Source file	=
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\Core\Src\stm32f4xx_it.c
#	Command line		=
#	-f
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\User\Core\stm32f4xx_it.o.rsp
#
(S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\Core\Src\stm32f4xx_it.c
#	-D USE_HAL_DRIVER -D STM32F429xx -lC
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\List\Ap
plication\User\Core
#	-o
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\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\Project1\Project1\EWARM/../Core/Inc\
#	-I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/STM3
2F4xx_HAL_Driver/Inc\
#	-I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/STM3
2F4xx_HAL_Driver/Inc/Legacy\
#	-I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/CMSI
S/Device/ST/STM32F4xx/Include\
#	-I
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM/../Drivers/CMSI
S/Include\
#        -Ohz) --dependencies=n
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\User\Core\stm32f4xx_it.o.d
#        Locale          =  C
#        List file       =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\List\Ap
plication\User\Core\stm32f4xx_it.lst
#        Object file     =
#
S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\EWARM\Project1\Obj\Ap
plication\User\Core\stm32f4xx_it.o
#        Runtime model:
#        __CPP_Runtime   =  1
#        __SystemLibrary =  DLib
#        __dlib_version  =  6
#        __size_limit    =  32768|ARM.EW.LINKER
#
###############################################################################
#

S:\School_Work\Spring_2023\MicroconrollerApps\Project1\Project1\Core\Src\stm32f4xx_it.c
          /* USER CODE BEGIN Header */
          /**
          
******************************************************************************
          * @file         stm32f4xx_it.c
          * @brief   Interrupt Service Routines.
          
******************************************************************************
          * @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"
     #include "stm32f4xx_it.h"
     /* Private includes ----------------------------------------------------------*/
     /* USER CODE BEGIN Includes */
     /* USER CODE END Includes */

     /* Private typedef -----------------------------------------------------------*/
     /* USER CODE BEGIN TD */

     /* USER CODE END TD */

     /* Private define ------------------------------------------------------------*/
     /* USER CODE BEGIN PD */

     /* USER CODE END PD */

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

     /* USER CODE END PM */

     /* Private variables ---------------------------------------------------------*/
     /* USER CODE BEGIN PV */

\             In section .bss, align 4
     int upper = 0;
\    upper:
\    0x0               DS8 4

\             In section .bss, align 4
     float high_pulse = 0;
\    high_pulse:
\    0x0               DS8 4
```

```
        float low_pulse = 0;
  \     low_pulse:
  \     0x4                 DS8 4

  \                 In section .bss, align 4
        int active_time = 0;
  \     active_time:
  \     0x0                 DS8 4

        /* USER CODE END PV */

        /* Private function prototypes -----------------------------------------------*/
        /* USER CODE BEGIN PFP */

        /* USER CODE END PFP */

        /* Private user code ---------------------------------------------------------*/
        /* USER CODE BEGIN 0 */

        /* USER CODE END 0 */

        /* External variables --------------------------------------------------------*/
        extern TIM_HandleTypeDef htim1;
        extern TIM_HandleTypeDef htim2;
        extern TIM_HandleTypeDef htim4;
        /* USER CODE BEGIN EV */
        extern int state;
        extern int blinks;
        extern uint8_t START;
        extern uint8_t STOP;
        extern float freq;
        /* USER CODE END EV */


/******************************************************************************
*/
        /*           Cortex-M4 Processor Interruption and Exception Handlers          */

/******************************************************************************
*/
```

```
     76     /**
     77             * @brief This function handles Non maskable interrupt.
     78             */

\                   In section .text, align 2, keep-with-next
     79     void NMI_Handler(void)
     80     {
     81             /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
     82
     83             /* USER CODE END NonMaskableInt_IRQn 0 */
     84             /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
     85             while (1)
\           NMI_Handler: (+1)
\           ??NMI_Handler_0: (+1)
\    0x0  0xE7FE        B.N     ??NMI_Handler_0
     86             {
     87             }
     88             /* USER CODE END NonMaskableInt_IRQn 1 */
     89     }
     90
     91     /**
     92             * @brief This function handles Hard fault interrupt.
     93             */

\                   In section .text, align 2, keep-with-next
     94     void HardFault_Handler(void)
     95     {
     96             /* USER CODE BEGIN HardFault_IRQn 0 */
     97
     98             /* USER CODE END HardFault_IRQn 0 */
     99             while (1)
\           HardFault_Handler: (+1)
\           ??HardFault_Handler_0: (+1)
\    0x0  0xE7FE        B.N     ??HardFault_Handler_0
    100             {
    101             /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    102             /* USER CODE END W1_HardFault_IRQn 0 */
    103             }
    104             }
    105
```

```
            /**
            * @brief This function handles Memory management fault.
            */

\           In section .text, align 2, keep-with-next
            void MemManage_Handler(void)
      {
            /* USER CODE BEGIN MemoryManagement_IRQn 0 */

            /* USER CODE END MemoryManagement_IRQn 0 */
            while (1)
\     MemManage_Handler: (+1)
\     ??MemManage_Handler_0: (+1)
\  0x0  0xE7FE      B.N    ??MemManage_Handler_0
            {
            /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
            /* USER CODE END W1_MemoryManagement_IRQn 0 */
            }
      }

            /**
            * @brief This function handles Pre-fetch fault, memory access fault.
            */

\           In section .text, align 2, keep-with-next
            void BusFault_Handler(void)
            {
            /* USER CODE BEGIN BusFault_IRQn 0 */

            /* USER CODE END BusFault_IRQn 0 */
            while (1)
\     BusFault_Handler: (+1)
\     ??BusFault_Handler_0: (+1)
\  0x0  0xE7FE      B.N    ??BusFault_Handler_0
            {
            /* USER CODE BEGIN W1_BusFault_IRQn 0 */
            /* USER CODE END W1_BusFault_IRQn 0 */
            }
            }
```

```
     136             /**
     137             * @brief This function handles Undefined instruction or illegal state.
     138             */

\                    In section .text, align 2, keep-with-next
     139             void UsageFault_Handler(void)
     140             {
     141             /* USER CODE BEGIN UsageFault_IRQn 0 */
     142
     143             /* USER CODE END UsageFault_IRQn 0 */
     144             while (1)
\          UsageFault_Handler: (+1)
\          ??UsageFault_Handler_0: (+1)
\    0x0  0xE7FE         B.N    ??UsageFault_Handler_0
     145             {
     146             /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
     147             /* USER CODE END W1_UsageFault_IRQn 0 */
     148             }
     149             }
     150
     151             /**
     152             * @brief This function handles System service call via SWI instruction.
     153             */

\                    In section .text, align 2, keep-with-next
     154             void SVC_Handler(void)
     155             {
     156             /* USER CODE BEGIN SVCall_IRQn 0 */
     157
     158             /* USER CODE END SVCall_IRQn 0 */
     159             /* USER CODE BEGIN SVCall_IRQn 1 */
     160
     161             /* USER CODE END SVCall_IRQn 1 */
     162             }
\          SVC_Handler: (+1)
\    0x0  0x4770         BX     LR
     163
     164             /**
     165             * @brief This function handles Debug monitor.
     166             */
```

```
\                      In section .text, align 2, keep-with-next
    167                void DebugMon_Handler(void)
    168                {
    169                /* USER CODE BEGIN DebugMonitor_IRQn 0 */
    170
    171                /* USER CODE END DebugMonitor_IRQn 0 */
    172                /* USER CODE BEGIN DebugMonitor_IRQn 1 */
    173
    174                /* USER CODE END DebugMonitor_IRQn 1 */
    175                }
\              DebugMon_Handler: (+1)
\   0x0  0x4770            BX      LR
    176
    177                /**
    178                * @brief This function handles Pendable request for system service.
    179                */

\                      In section .text, align 2, keep-with-next
    180                void PendSV_Handler(void)
    181                {
    182                /* USER CODE BEGIN PendSV_IRQn 0 */
    183
    184                /* USER CODE END PendSV_IRQn 0 */
    185                /* USER CODE BEGIN PendSV_IRQn 1 */
    186
    187                /* USER CODE END PendSV_IRQn 1 */
    188                }
\              PendSV_Handler: (+1)
\   0x0  0x4770            BX      LR
    189
    190                /**
    191                * @brief This function handles System tick timer.
    192                */

\                      In section .text, align 2, keep-with-next
    193                void SysTick_Handler(void)
    194                {
    195                /* USER CODE BEGIN SysTick_IRQn 0 */
    196
```

```
        197             /* USER CODE END SysTick_IRQn 0 */
        198             HAL_IncTick();
  \             SysTick_Handler: (+1)
  \     0x0  0x.... 0x....       B.W    HAL_IncTick
        199             /* USER CODE BEGIN SysTick_IRQn 1 */
        200
        201             /* USER CODE END SysTick_IRQn 1 */
        202             }
        203
        204
/******************************************************************************
*/
        205             /* STM32F4xx Peripheral Interrupt Handlers                        */
        206             /* Add here the Interrupt Handlers for the used peripherals.
*/
        207             /* For the available peripheral interrupt handler names,          */
        208             /* please refer to the startup file (startup_stm32f4xx.s).        */
        209
/******************************************************************************
*/
        210
        211     /**
        212             * @brief This function handles TIM1 capture compare interrupt.
        213             */

  \                     In section .text, align 2, keep-with-next
        214             void TIM1_CC_IRQHandler(void)
        215             {
        216             /* USER CODE BEGIN TIM1_CC_IRQn 0 */
        217
        218             /* USER CODE END TIM1_CC_IRQn 0 */
        219             HAL_TIM_IRQHandler(&htim1);
  \             TIM1_CC_IRQHandler: (+1)
  \     0x0  0x....              LDR.NR0,??DataTable3_1
  \     0x2  0x.... 0x....       B.W    HAL_TIM_IRQHandler
        220             /* USER CODE BEGIN TIM1_CC_IRQn 1 */
        221
        222             /* USER CODE END TIM1_CC_IRQn 1 */
        223             }
        224
```

```
     225              /**
     226              * @brief This function handles TIM2 global interrupt.
     227              */

\                     In section .text, align 2, keep-with-next
     228              void TIM2_IRQHandler(void)
     229              {
\             TIM2_IRQHandler: (+1)
\    0x0  0xB580             PUSH  {R7,LR}
     230              /* USER CODE BEGIN TIM2_IRQn 0 */
     231              HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_3); // toggle the LED
\    0x2  0x2108             MOVS          R1,#+8
\    0x4  0x....             LDR.NR0,??DataTable3_2
\    0x6  0x.... 0x....      BL     HAL_GPIO_TogglePin
     232              if (upper == 0){
\    0xA  0x....             LDR.NR1,??DataTable3_3
\    0xC  0x6808             LDR    R0,[R1, #+0]
\    0xE  0xB908             CBNZ.N   R0,??TIM2_IRQHandler_0
     233              upper = 1;
\    0x10  0x2001            MOVS          R0,#+1
\    0x12  0xE004            B.N    ??TIM2_IRQHandler_1
     234              }
     235              else{
     236              blinks++; // add one to the blink counter
\             ??TIM2_IRQHandler_0: (+1)
\    0x14  0x....            LDR.NR0,??DataTable3_4
\    0x16  0x6802            LDR    R2,[R0, #+0]
\    0x18  0x1C52            ADDS R2,R2,#+1
\    0x1A  0x6002            STR    R2,[R0, #+0]
     237              upper = 0;
\    0x1C  0x2000            MOVS          R0,#+0
     238              }
     239              /* USER CODE END TIM2_IRQn 0 */
     240              HAL_TIM_IRQHandler(&htim2);
\             ??TIM2_IRQHandler_1: (+1)
\    0x1E  0x6008            STR    R0,[R1, #+0]
\    0x20  0x....            LDR.NR0,??DataTable3_5
\    0x22  0x.... 0x....     BL     HAL_TIM_IRQHandler
     241              /* USER CODE BEGIN TIM2_IRQn 1 */
     242              TIM1 -> CNT = 0; // reset the timer count
```

```
  \      0x26   0x....            LDR.NR1,??DataTable3_6
  \      0x28   0x2000            MOVS          R0,#+0
  \      0x2A   0x6008            STR    R0,[R1, #+0]
         243              /* USER CODE END TIM2_IRQn 1 */
         244              }
  \      0x2C   0xBD01            POP    {R0,PC}
         245
         246              /**
         247              * @brief This function handles TIM4 global interrupt.
         248              */

  \                       In section .text, align 4, keep-with-next
         249              void TIM4_IRQHandler(void)
         250              {
  \              TIM4_IRQHandler: (+1)
  \      0x0   0xB538             PUSH  {R3-R5,LR}
         251              /* USER CODE BEGIN TIM4_IRQn 0 */
         252              if (HAL_GPIO_ReadPin(GPIOD, GPIO_PIN_12) == GPIO_PIN_SET){
  \      0x2   0xF44F 0x5180             MOV   R1,#+4096
  \      0x6   0x....             LDR.NR0,??DataTable3_2
  \      0x8   0x....             LDR.NR4,??DataTable3_7
  \      0xA   0x....             LDR.NR5,??DataTable3_8
  \      0xC   0x.... 0x....      BL     HAL_GPIO_ReadPin
  \      0x10   0x2801            CMP    R0,#+1
  \      0x12   0xD10A            BNE.N??TIM4_IRQHandler_0
         253              high_pulse = TIM4 -> CNT / 50;  // high_pulse holds the time from start
to rising edge
  \      0x14   0x6828            LDR    R0,[R5, #+0]
  \      0x16   0x2132            MOVS          R1,#+50
  \      0x18   0xFBB0 0xF1F1     UDIV  R1,R0,R1
  \      0x1C   0xEE00 0x1A10     VMOV          S0,R1
  \      0x20   0xEEB8 0x0A40     VCVT.F32.U32 S0,S0
  \      0x24   0xED84 0x0A00     VSTR  S0,[R4, #0]
  \      0x28   0xE016            B.N    ??TIM4_IRQHandler_1
         254              }
         255              else {
         256              low_pulse = TIM4 -> CNT / 50; // low_pulse stores the time from start to
falling edge
  \              ??TIM4_IRQHandler_0: (+1)
  \      0x2A   0x682A            LDR    R2,[R5, #+0]
\    0x2C   0x2032          MOVS          R0,#+50
\    0x2E   0xFBB2 0xF0F0        UDIV  R0,R2,R0
\    0x32   0xEE00 0x0A90        VMOV          S1,R0
\    0x36   0xEEB8 0x0A60        VCVT.F32.U32 S0,S1
\    0x3A   0xED84 0x0A01        VSTR  S0,[R4, #+4]
     257              freq = (1/low_pulse)*1000000; // low_pulse is the period in us
\    0x3E   0xEEB7 0x1A00        VMOV.F32 S2,#1.0
\    0x42   0xEE81 0x0A00        VDIV.F32 S0,S2,S0
\    0x46   0xEDDF 0x....        VLDR.W   S3,??DataTable3
\    0x4A   0x....          LDR.NR0,??DataTable3_9
\    0x4C   0xEE60 0x1A21        VMUL.F32 S3,S0,S3
\    0x50   0xEDC0 0x1A00        VSTR  S3,[R0, #0]
     258              // freq should be in Hz
     259              TIM4->CNT = 0; //reset the timer count
\    0x54   0x2100          MOVS          R1,#+0
\    0x56   0x6029          STR    R1,[R5, #+0]
     260              }
     261              /* USER CODE END TIM4_IRQn 0 */
     262              HAL_TIM_IRQHandler(&htim4);
\            ??TIM4_IRQHandler_1: (+1)
\    0x58   0xE8BD 0x4032        POP    {R1,R4,R5,LR}
\    0x5C   0x....          LDR.NR0,??DataTable3_10
\    0x5E   0x.... 0x....   B.W    HAL_TIM_IRQHandler
     263              /* USER CODE BEGIN TIM4_IRQn 1 */
     264
     265              /* USER CODE END TIM4_IRQn 1 */
     266              }

\                     In section .text, align 4, keep-with-next
\            ??DataTable3:
\    0x0   0x4974'2400     DC32  0x49742400

\                     In section .text, align 4, keep-with-next
\            ??DataTable3_1:
\    0x0   0x....'....     DC32  htim1

\                     In section .text, align 4, keep-with-next
\            ??DataTable3_2:
\    0x0   0x4002'0C00     DC32  0x40020c00
\                      In section .text, align 4, keep-with-next
\              ??DataTable3_3:
\       0x0   0x....'....          DC32   upper

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_4:
\       0x0   0x....'....          DC32   blinks

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_5:
\       0x0   0x....'....          DC32   htim2

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_6:
\       0x0   0x4001'0024     DC32   0x40010024

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_7:
\       0x0   0x....'....          DC32   high_pulse

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_8:
\       0x0   0x4000'0824     DC32   0x40000824

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_9:
\       0x0   0x....'....          DC32   freq

\                      In section .text, align 4, keep-with-next
\              ??DataTable3_10:
\       0x0   0x....'....          DC32   htim4
        267
        268            /* USER CODE BEGIN 1 */
        269
        270            /* USER CODE END 1 */

Maximum stack usage in bytes:

.cstack Function
------- --------
```

```
        0   BusFault_Handler
        0   DebugMon_Handler
        0   HardFault_Handler
        0   MemManage_Handler
        0   NMI_Handler
        0   PendSV_Handler
        0   SVC_Handler
        0   SysTick_Handler
        0   -> HAL_IncTick
        0   TIM1_CC_IRQHandler
        0   -> HAL_TIM_IRQHandler
        8   TIM2_IRQHandler
        8   -> HAL_GPIO_TogglePin
        8   -> HAL_TIM_IRQHandler
       16   TIM4_IRQHandler
       16   -> HAL_GPIO_ReadPin
        0   -> HAL_TIM_IRQHandler
        0   UsageFault_Handler


   Section sizes:

   Bytes  Function/Label
   -----  --------------
        4  ??DataTable3
        4  ??DataTable3_1
        4  ??DataTable3_10
        4  ??DataTable3_2
        4  ??DataTable3_3
        4  ??DataTable3_4
        4  ??DataTable3_5
        4  ??DataTable3_6
        4  ??DataTable3_7
        4  ??DataTable3_8
        4  ??DataTable3_9
        2  BusFault_Handler
        2  DebugMon_Handler
        2  HardFault_Handler
        2  MemManage_Handler
        2  NMI_Handler
```

```
          2  PendSV_Handler
          2  SVC_Handler
          4  SysTick_Handler
          6  TIM1_CC_IRQHandler
          46  TIM2_IRQHandler
          98  TIM4_IRQHandler
          2  UsageFault_Handler
          4  active_time
          8  high_pulse
          low_pulse
          4  upper


  16 bytes in section .bss
 214 bytes in section .text

 214 bytes of CODE memory
  16 bytes of DATA memory

Errors: none
Warnings: none
```