Electrical and Computer Engineering Department

ECE 4510 Microcontroller Applications

Design Project #1

Conveyer Belt with Object Detection

Yonatan Beyene & Dawson Hamill

March 22, 2021

Table of Contents

Introduction

The purpose of this lab is to design and interface with a conveyer belt which can stop an object using a photo receiver sensor. We implemented debounced SPST switches, an H-bridge motor driver, a buzzer, a voltage divider circuit, an amplifier, LEDs, a graphical LCD and the STM32F429ZI microcontroller in our design. First, we divided the project into three tasks. Then we implemented each task. After we were able to accomplish each task, we then incorporated the bonus projects. In the following sections, we will describe our design and the C program we developed to implement it.

Design

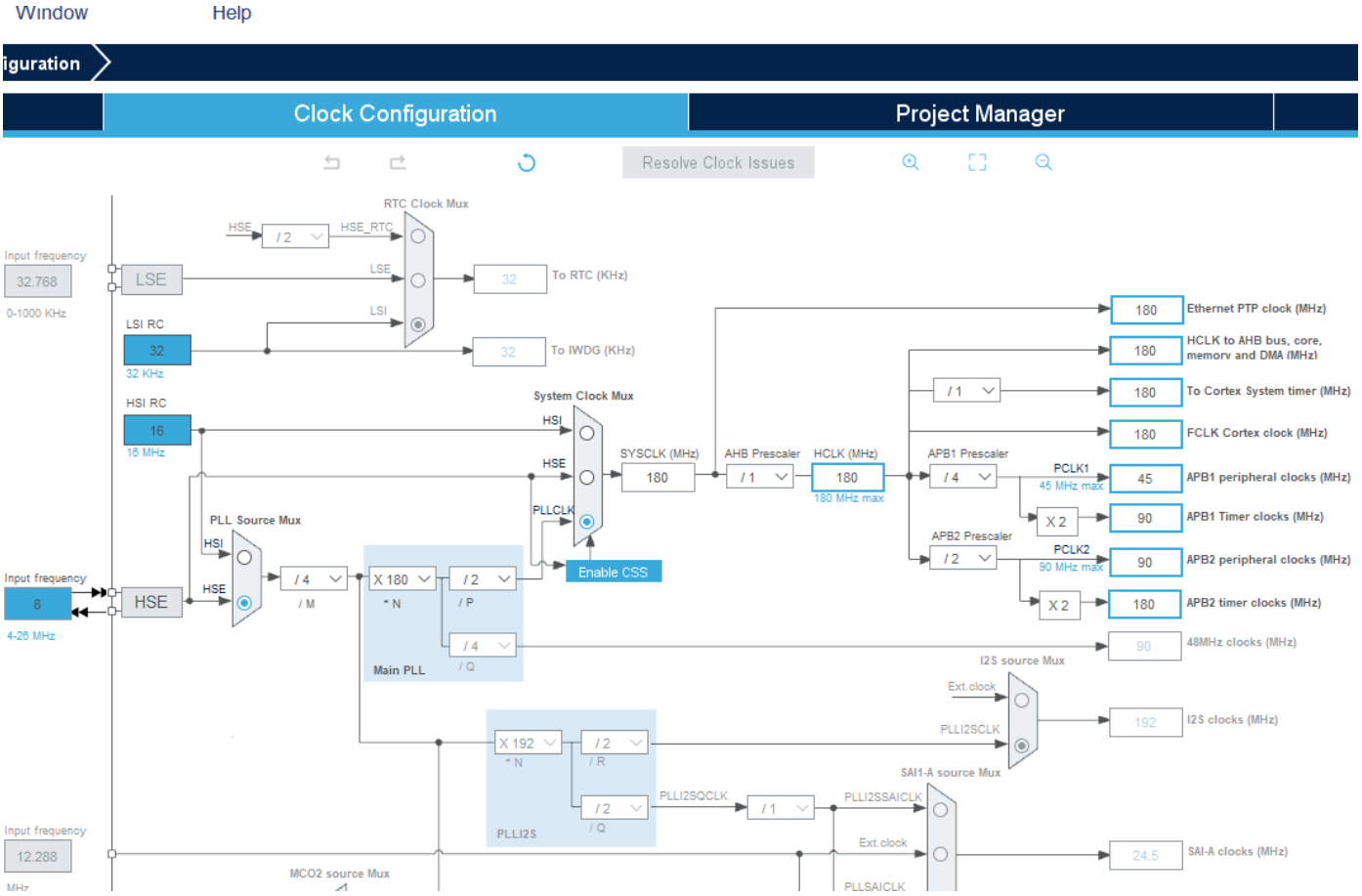
The design section will provide a brief description of the pins and modules used. The configuration settings will also be shown.

Port Configurations

Below are our pin configurations in STM32CubeMX used to implement the project.

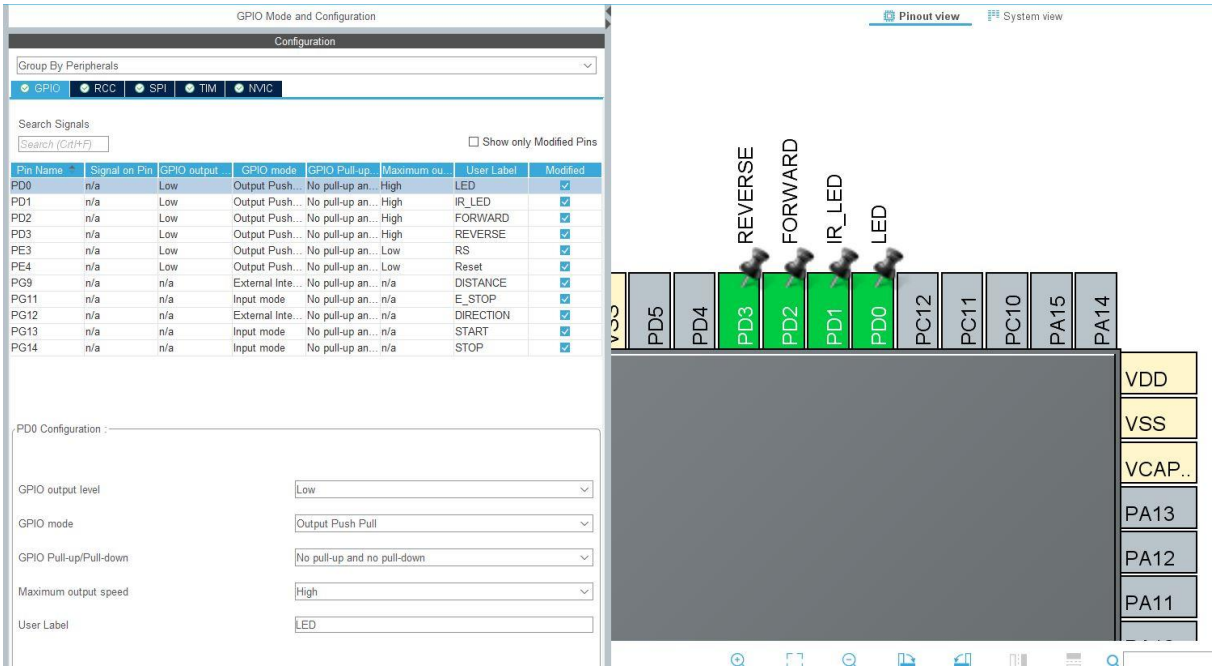
Clock

We first configured the HCLK rate and the AHB clock rate (using the HSE 8MHz clock) to 180MHz. Our configuration is as follows.



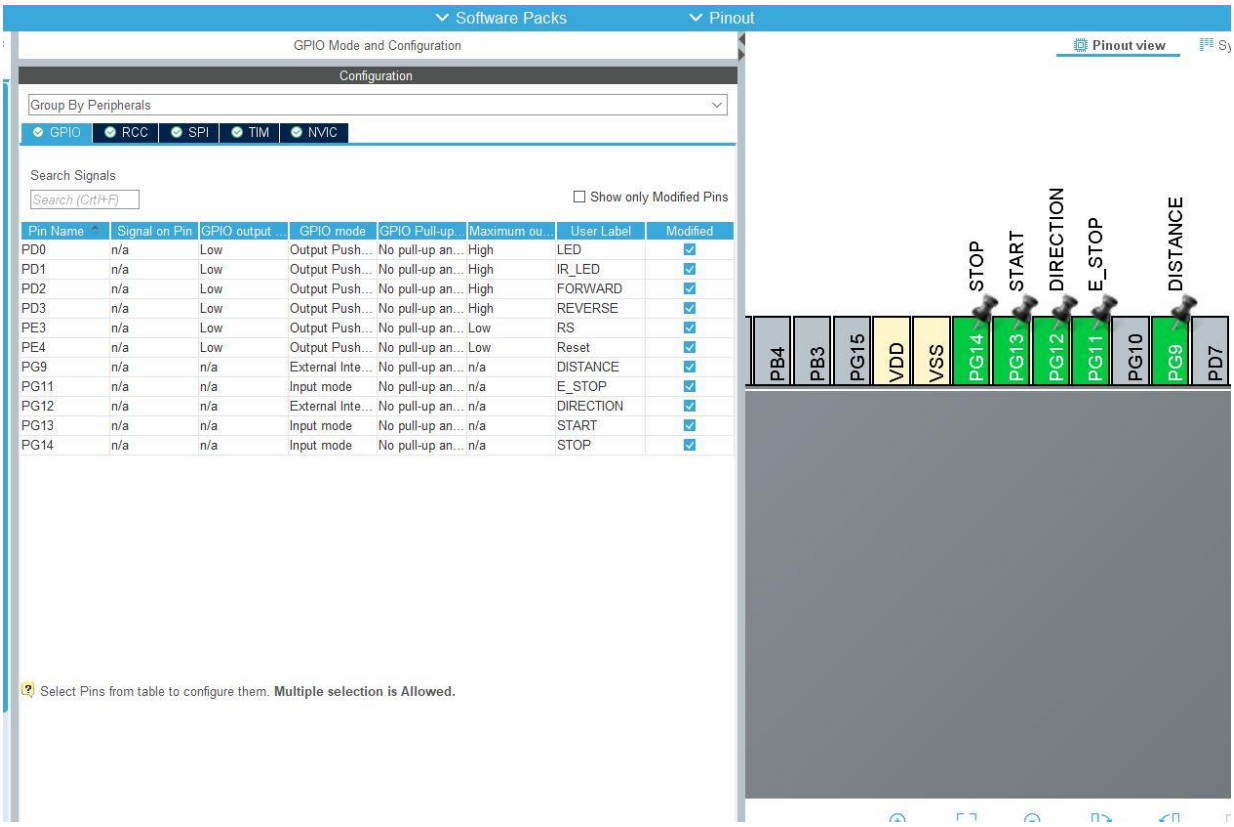
LED indicator and IR LED

The LED indicator is mapped with pin PD0 and IR emitter LED (IR_LED) is mapped with pin PD1. Both pins are GPIO outputs. Configuration is shown below.



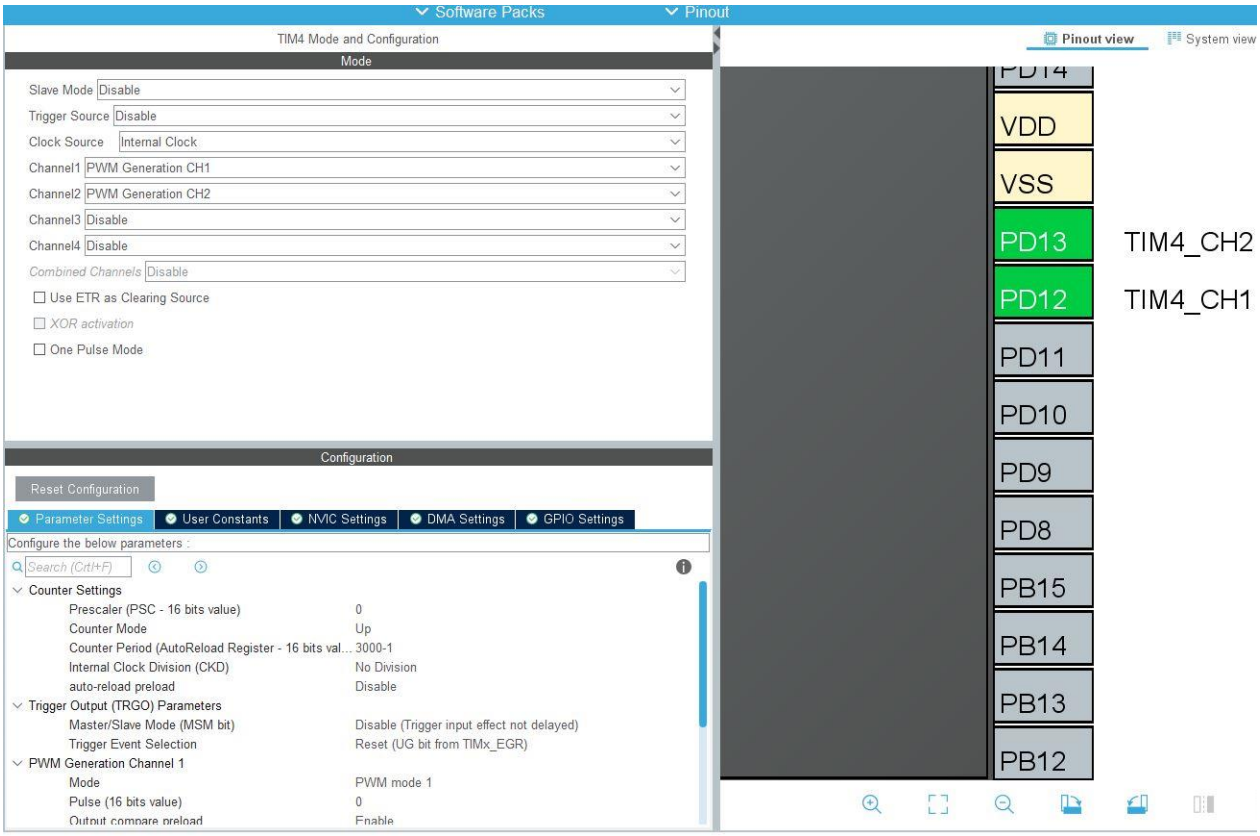
START, STOP and Emergency Stop

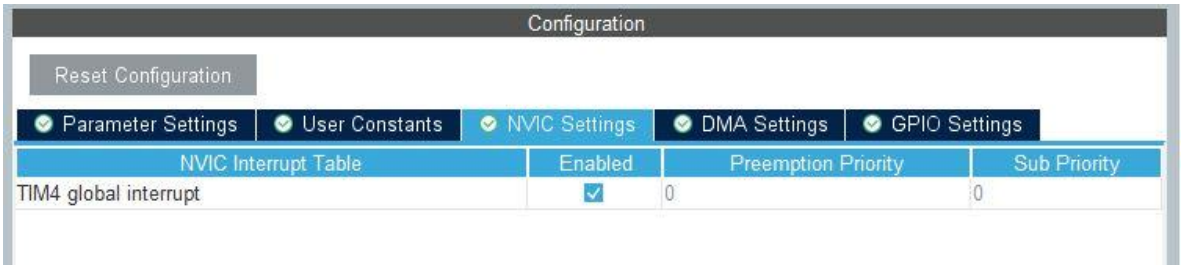
START# signal, STOP# signal (which later is replaced by signals from photo receiver sensors) and Emergency Stop signal are mapped to pin PG13, PG14 and PG11 respectively. All these three signals are active low signals. They are configured as GPIO input. Their configuration is as follows.



PWM generation for buzzer

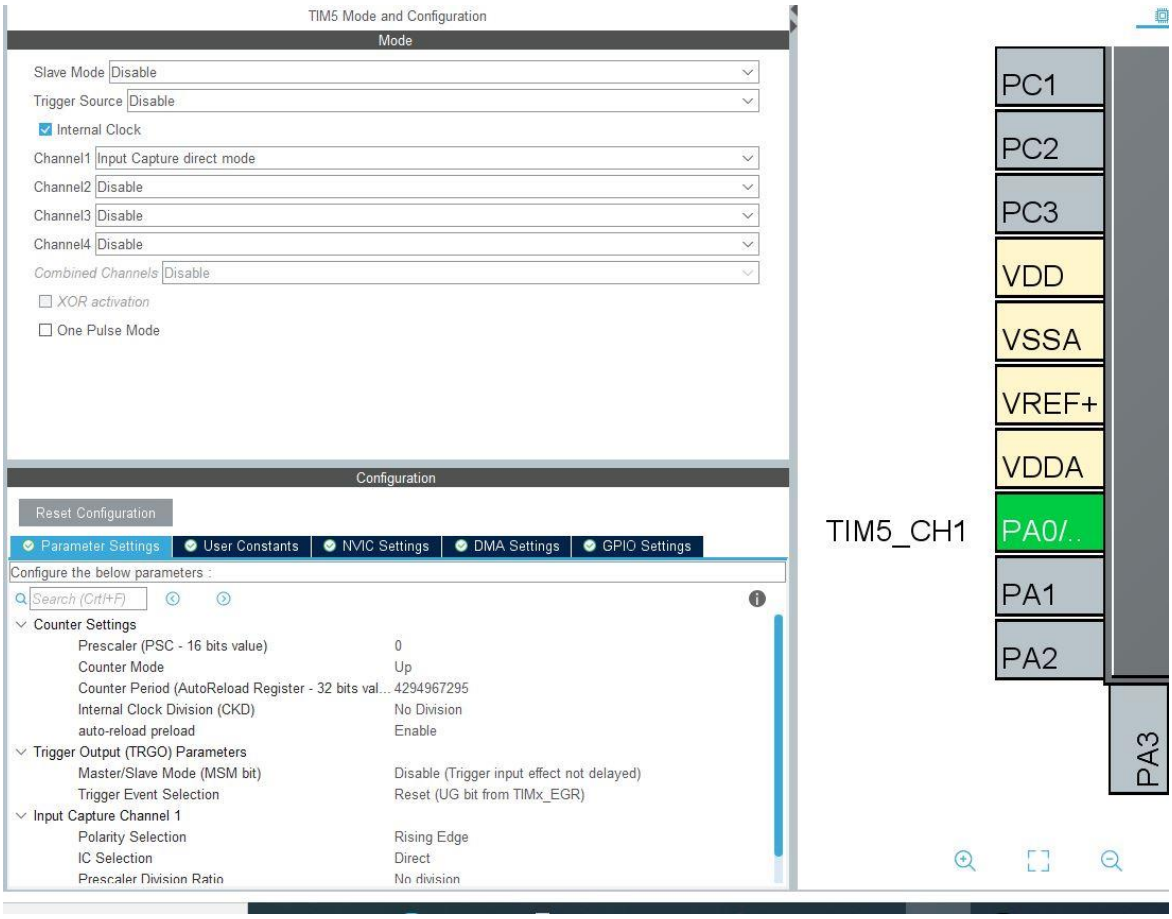
TIM4 channel 1 is used to generate PWM for buzzer. We used PSR-29F08S02-JQ speaker as our buzzer. We are also using TIM4 channel 2 for the motor driver PWM signal generation. That is why we have the auto reload register (ARR) configured to reload after every 3000 counts. We update the ARR in our program to generate 4.5 kHz and 3.6 kHz frequencies for the buzzer. The calculations will be explained in the procedure section. Here is our pin configuration.

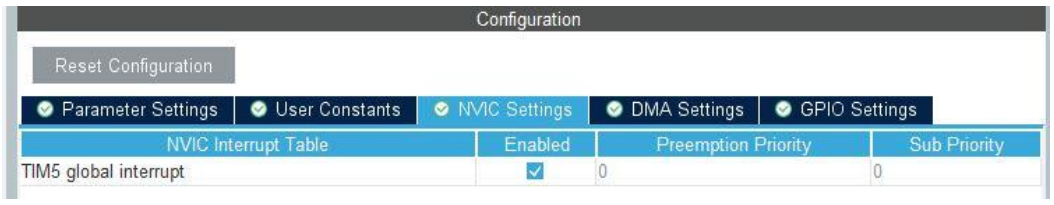




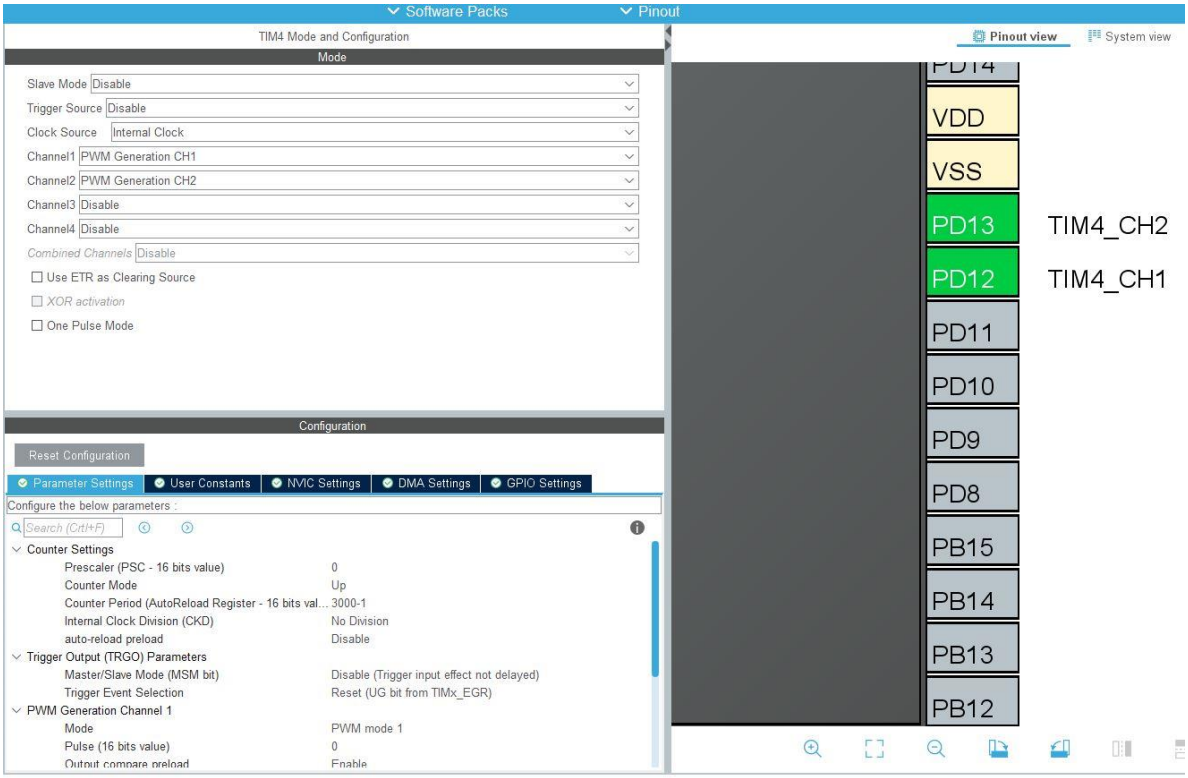
Input capture for Encoder waveform from wavegen and PWM generation for motor driver

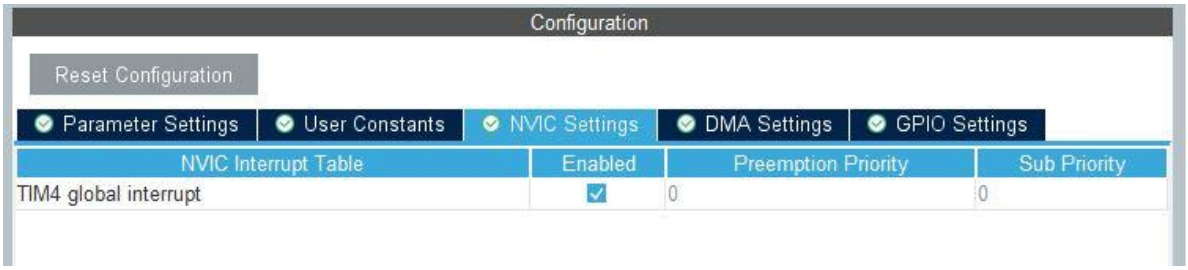
TIM5 channel 1 is used to capture encoder signal from wave generator. Its configuration is shown below.





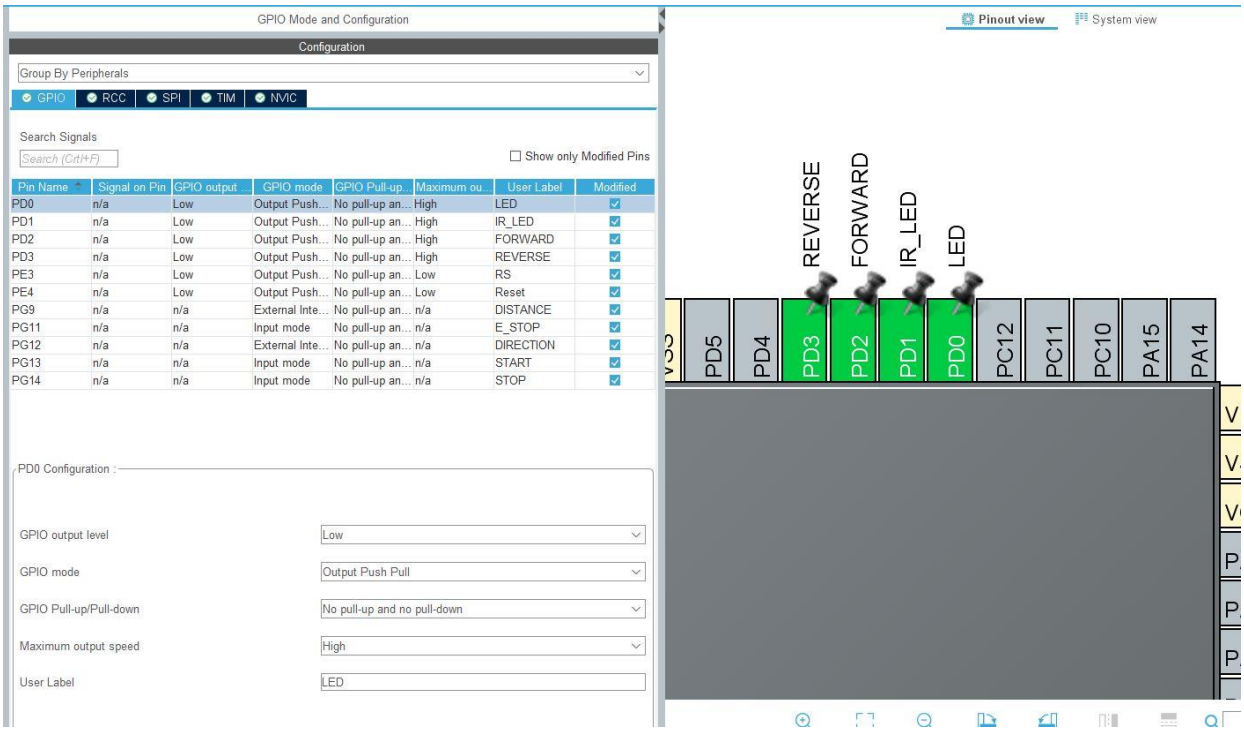
TIM4 channel 2 is used to generate the 30 kHz PWM signal. Its configuration is as follows.



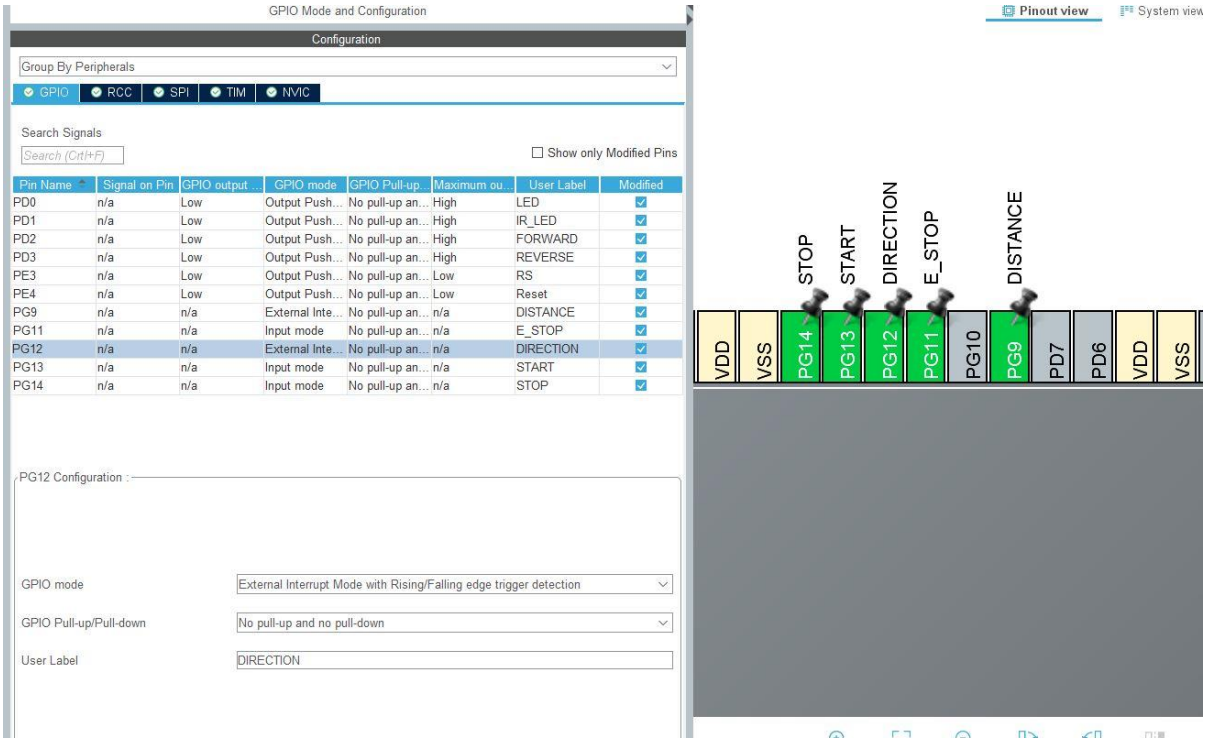


Forward/Reverse control - Forward/Reverse pins, and External interrupt

The forward and reverse pins are pin PD2 and pin PD3 respectively. They both are GPIO output and are never active at the same time (i.e. if PD2 is high, then PD3 is low, and vice versa). They are toggled by an external interrupt. Their configuration is as follows.

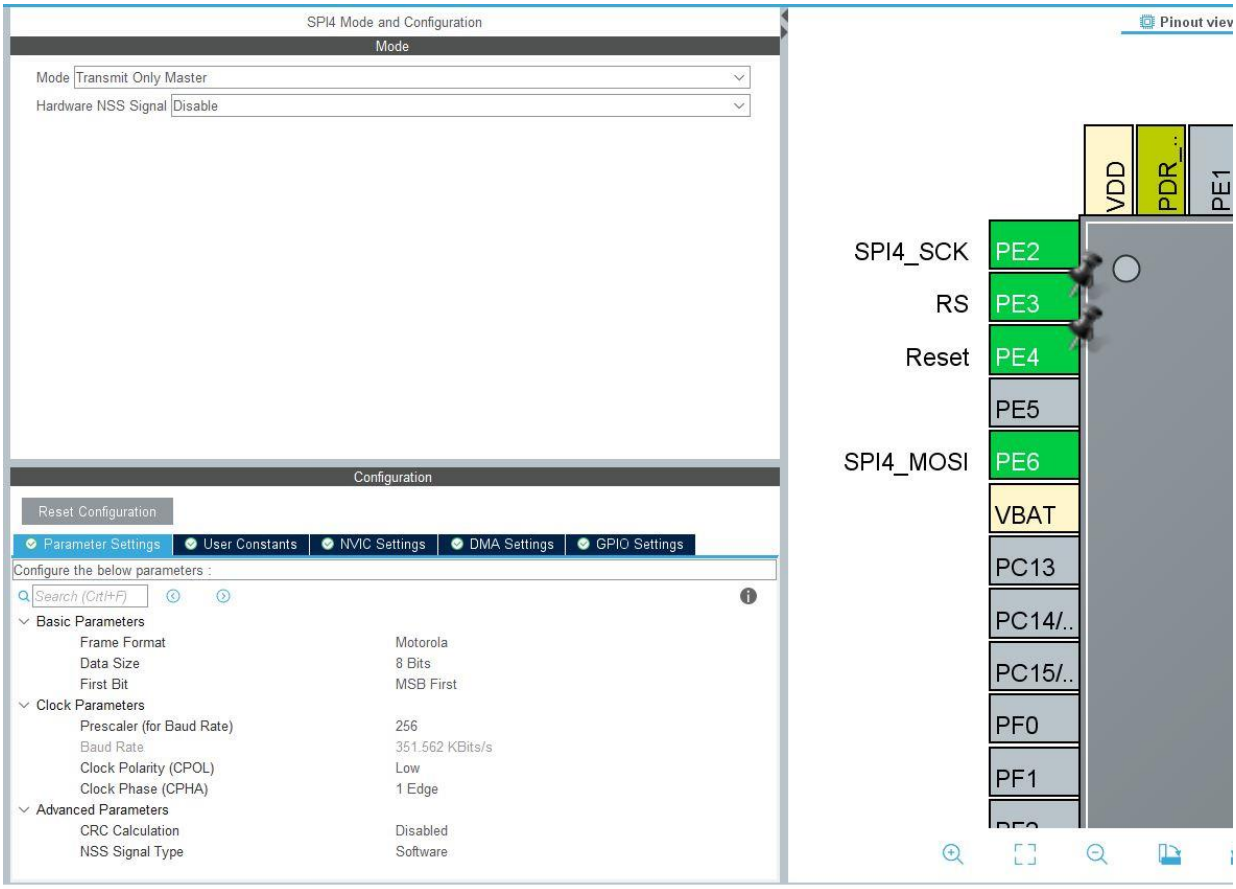


To control the state of forward and backward pins with a switch, we use the external interrupt EXTI15_10. We used pin PG12 to configure this interrupt and called it DIRECTION. The interrupt is triggered with both rising and falling edges. Our configuration is as follows.



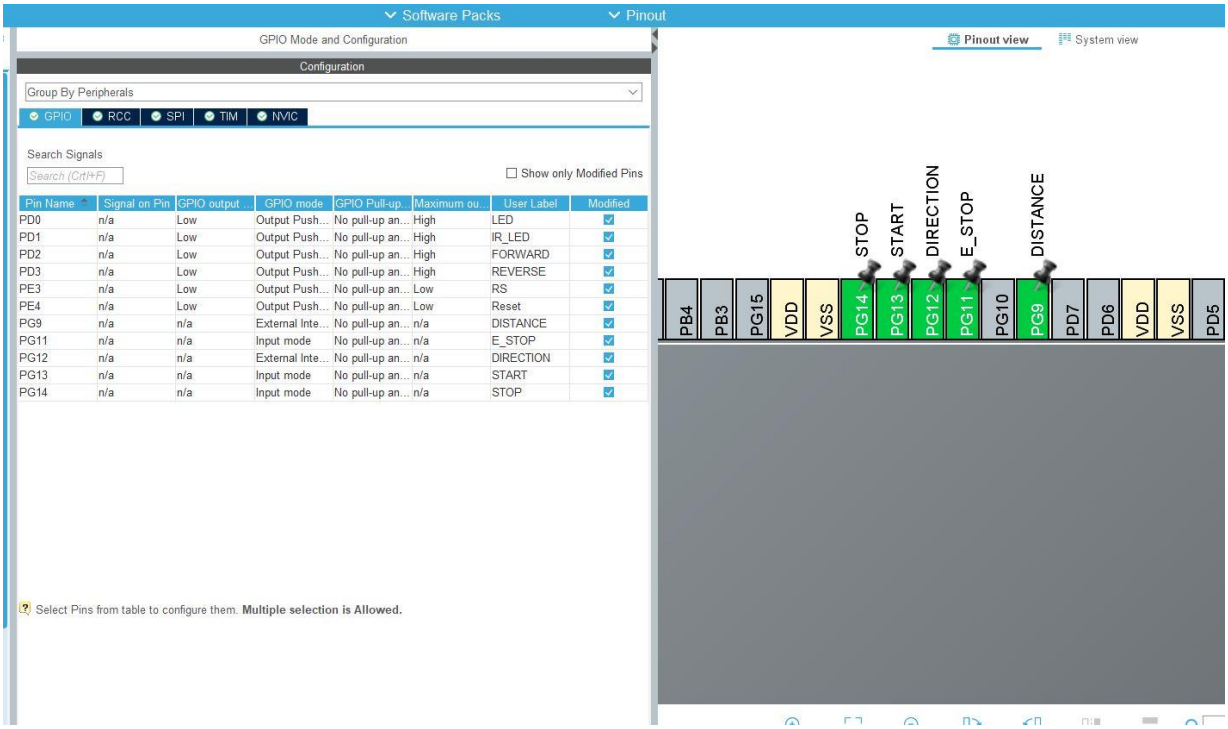
GLCD display – SPI 4 module

To display the status of our motor on GLCD, we configured SPI4 module of our microcontroller. We also set Pin PE3 and Pin PE4 as our RS and Reset pins respectively. Our configuration is as follows.



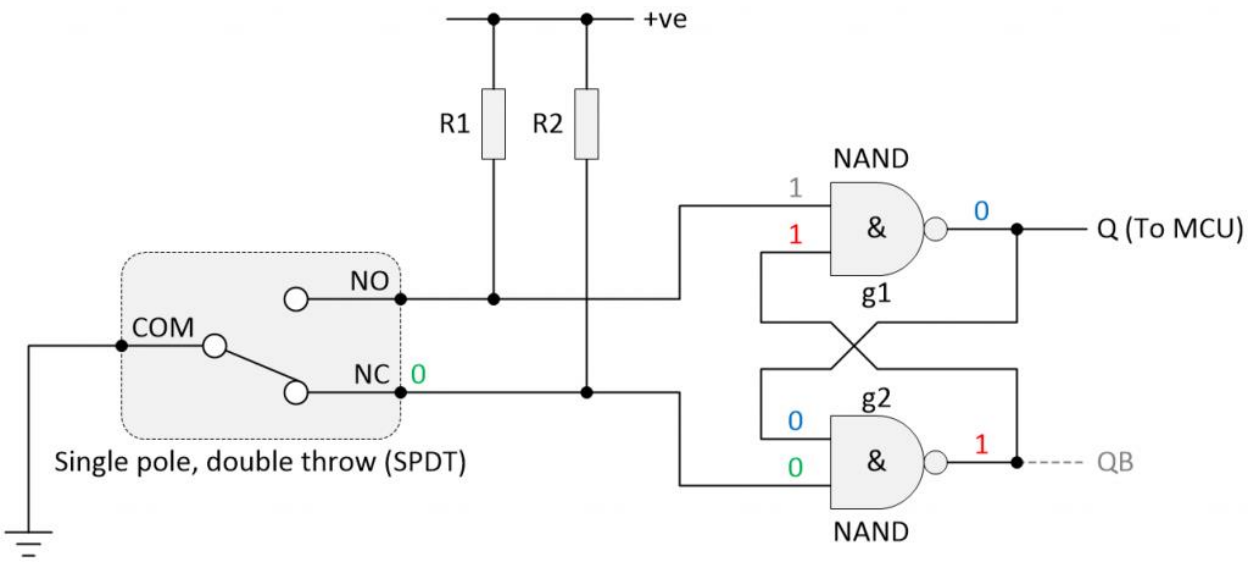
External interrupt to measure distance

We attempted to measure distance by using the encoder pulse signal from the motor. To count a counter for each pulse, we configure another external interrupt EXTI9_5. We used pin PG9 to configure this interrupt and called it DISTANCE. The interrupt is triggered with rising edges. Our configuration is as follows.



Procedure

Here are the procedures we followed to implement the project. The first task we implemented is receiving START# signal, blinking LED indicator 6 times with 1 blinks/sec and sounding the buzzer at 5 kHz (4.5 kHz) until the LED stops blinking. START# signal status comes from a debounced SPST switch. We used this switch for START#, E_STOP# and DIRECTION control signals. The debounced circuit we used for the switches is as follows.

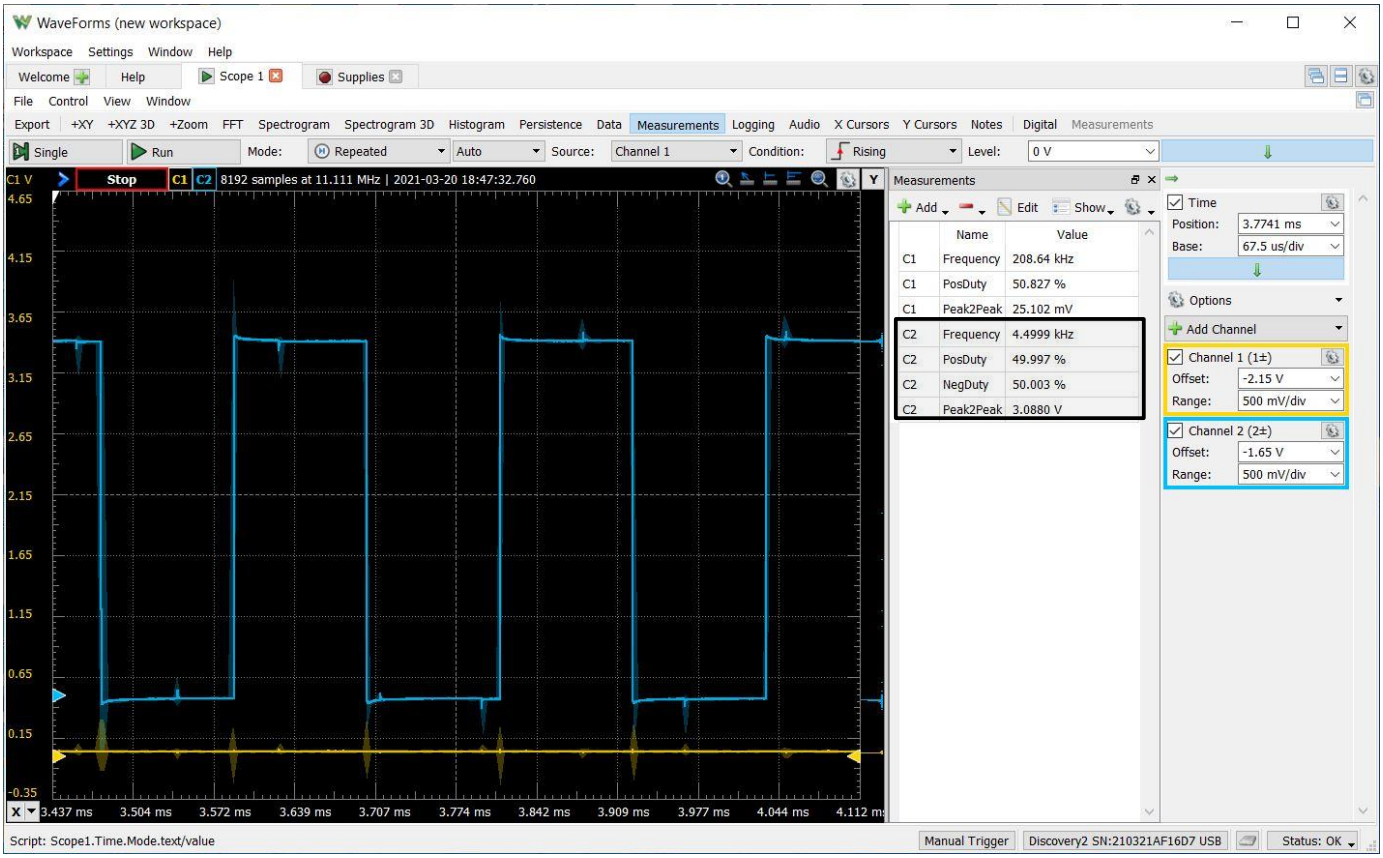


Source: eejournal.com

Once we set up our START# switch, we then configured Timer TIM4 channel 1 to generate PWM signal to sound our buzzer (PSR-29F08S02-JQ speaker). We were required to generate a PWM of 5 kHz frequency for the first step, but after reading the data sheet of the speaker we used, we decided to go with 4.5 kHz because that was the recommended maximum frequency. We configured auto reload register (ARR) to reload after every 20000 counts to generate PWM of 4.5 kHz.

ARR (counter period) = (F_{lIm} * prescaler) / Freq = 90 M / 4.5 kHz = 20,000.

The PWM we generated is as follows.



Then, we blinked the LED (green LED) 6 times using a for-loop. Once the blinking is over, we stop the TIM4 channel 1 PWM interrupt to stop the buzzer.

The second task we implemented is moving the motor and stopping it when an object is sensed. First, we turned on IR_LED (red LED). Then we generated a PWM of 30 kHz with initial duty cycle of 50 % to match 5.5 kHz from Encoder module. Our Encoder module comes from the wave generator of an oscilloscope. We used TIM4 channel 2 to generate the PWM signal. We set the auto reload register (ARR) to reload after every 3000 counts to generate PWM of 30 kHz.

ARR (counter period) = (F_{lIm} * prescaler) / Freq = 90 M / 30 kHz = 3,000.

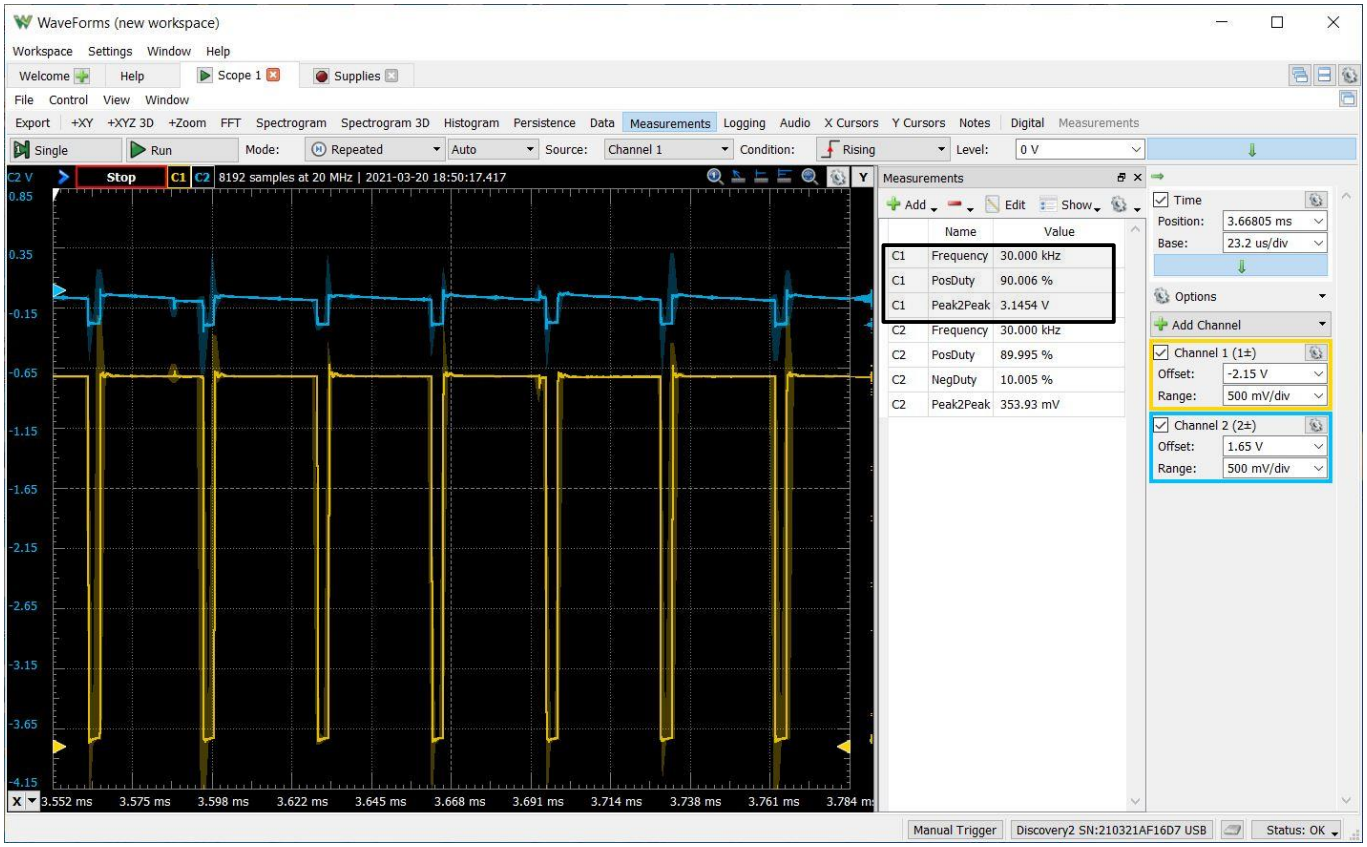
Our encoder signal is captured by Timer TIM5 channel 1 using input capture. We then calculated the frequency of the encoder signal using this timer interrupt. Then based on that we updated the duty cycle of the 30 kHz PWM. We came up with a linear equation that relates frequency of the encoder signal to the required duty cycle of the 30 kHz duty cycle. The equation is as follows.

Duty cycle of PWM = -Captured_freq/20 + 325

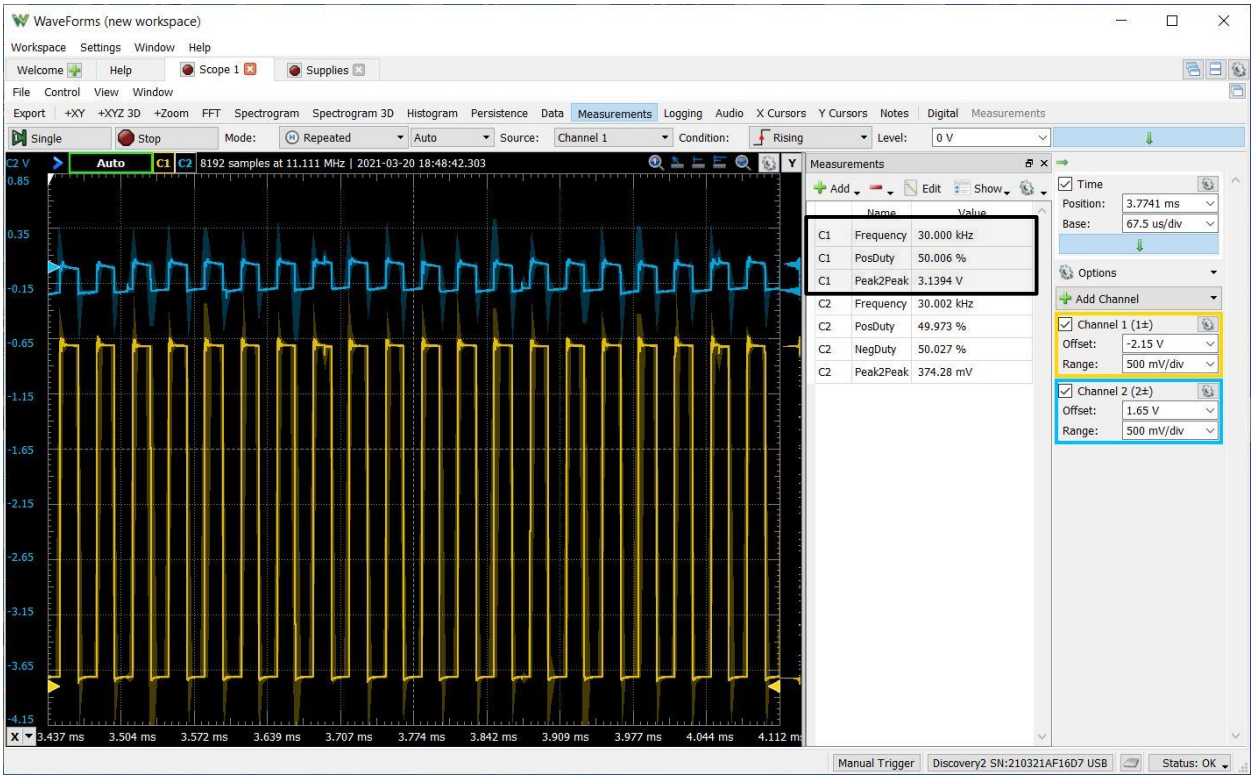
After that we update the capture compare register of TIM4 channel 2 using the following equation.

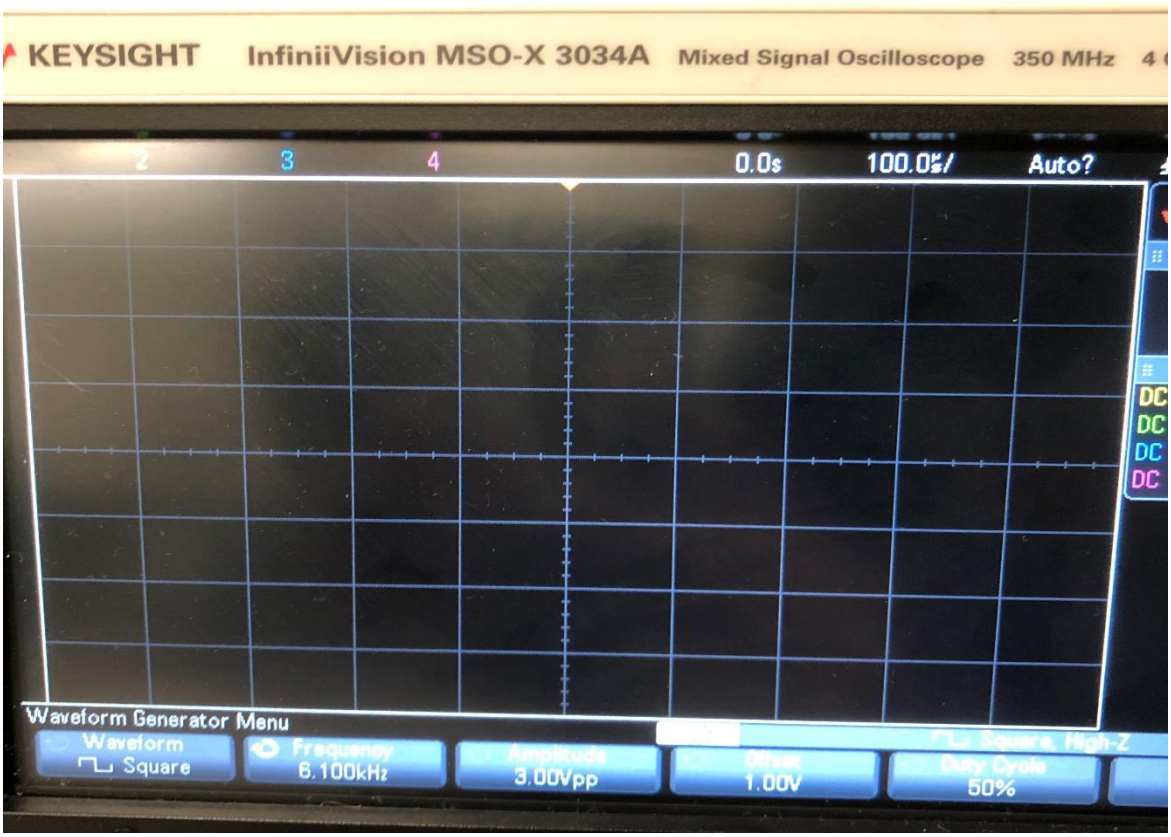
High duty = (uint16_t)Duty cycle of PWM*3000/100 + 24

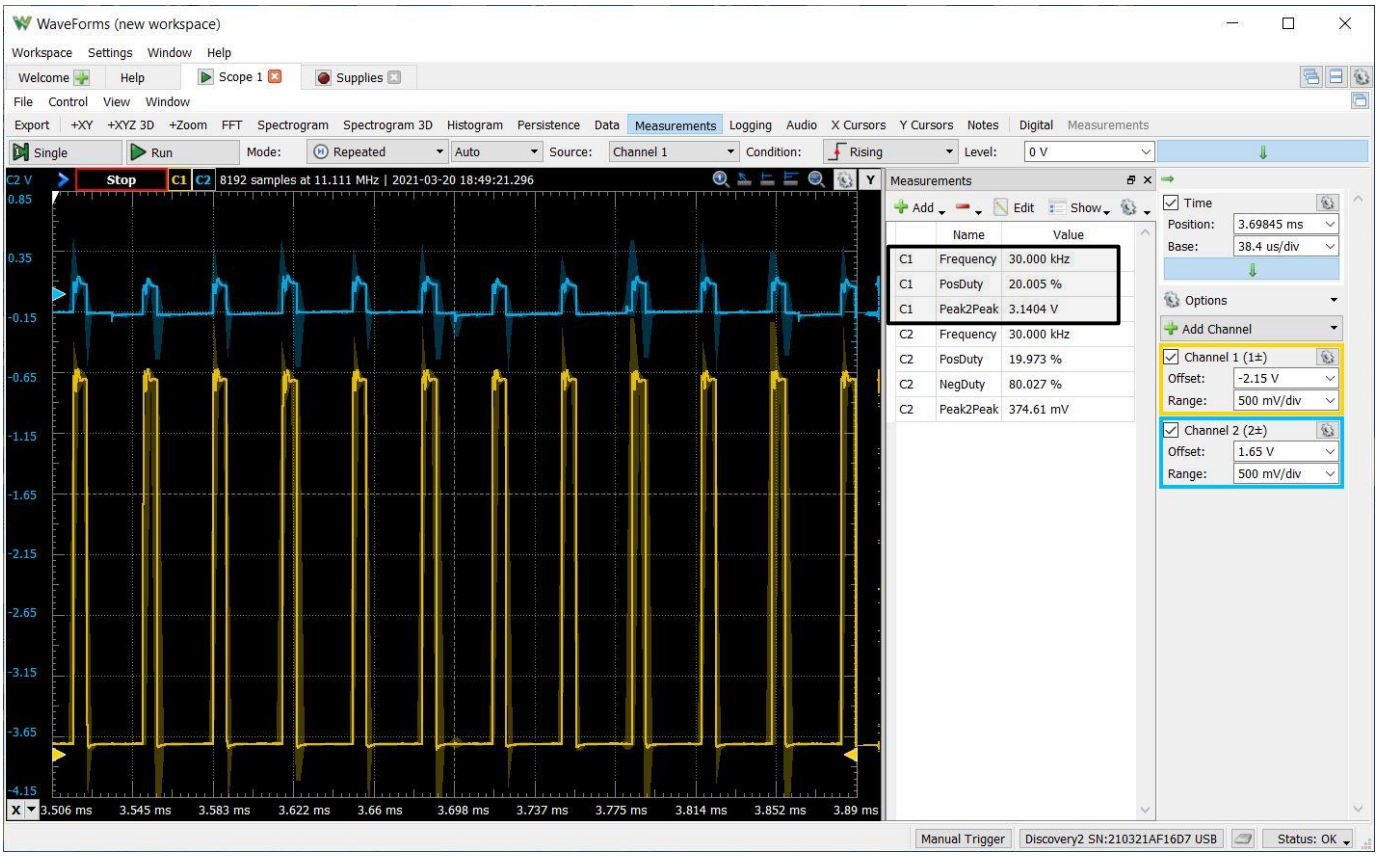
This gave us the correct duty cycles for all encoder module frequencies given. Some of the frequencies and the PWM signals generated are shown below.



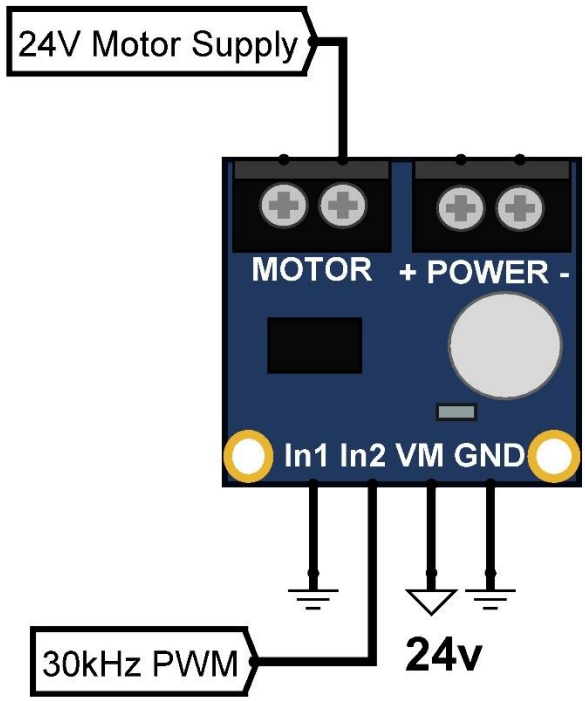

KEYSIGHT
InfiniiVision MSO-X 3034A
Mixed Signal Oscilloscope
350 MHz
4 GSa/s
0.0s
Auto?
0.0V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
2.00GSa/s
Channels
DC 1.00:1
DC 1.00:1
DC 10.0:1
DC 10.0:1
Waveform Generator Menu
Square, High-Z
Waveform
Square
Frequency Fine
5.500kHz
Amplitude
3.00Vpp
Offset
1.00V
Duty Cycle
50%
Settings


WaveForms (new workspace)
Workspace Settings Window Help
Welcome Help Scope 1 Supplies
File Control View Window
Export +XY +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Data Measurements Logging Audio X Cursors Y Cursors Notes Digital Measurements
Single Stop Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V
Auto C1 C2 8192 samples at 11.111 MHz | 2021-03-20 18:48:42.303
Measurements
Add Edit Show
Name Value
C1 Frequency 30.000 kHz
C1 PosDuty 50.006 %
C1 Peak2Peak 3.1394 V
C2 Frequency 30.002 kHz
C2 PosDuty 49.973 %
C2 NegDuty 50.027 %
C2 Peak2Peak 374.28 mV
Time
Position: 3.7741 ms
Base: 67.5 us/div
Options
Add Channel
Channel 1 (1±)
Offset: -2.15 V
Range: 500 mV/div
Channel 2 (2±)
Offset: 1.65 V
Range: 500 mV/div
0.85
0.35
-0.15
-0.65
-1.15
-1.65
-2.15
-2.65
-3.15
-3.65
-4.15
3.437 ms 3.504 ms 3.572 ms 3.639 ms 3.707 ms 3.774 ms 3.842 ms 3.909 ms 3.977 ms 4.044 ms 4.112 ms
Manual Trigger
Discovery2 SN:210321AF16D7 USB
Status: OK

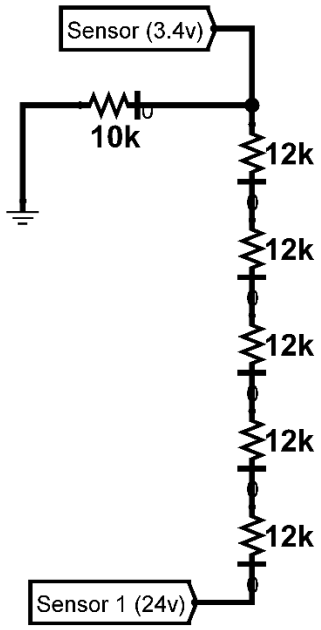

KEYSIGHT
InfiniiVision MSO-X 3034A
Mixed Signal Oscilloscope
350 MHz
0.0s
100.0%/
Auto?
Waveform Generator Menu
Waveform
Square
Frequency
6.100kHz
Amplitude
3.00Vpp
Offset
1.00V
Duty Cycle
50%


WaveForms (new workspace)
Workspace Settings Window Help
Welcome Help Scope 1 Supplies
File Control View Window
Export +XY +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Data Measurements Logging Audio X Cursors Y Cursors Notes Digital Measurements
Single Run Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V
Stop C1 C2 8192 samples at 11.111 MHz | 2021-03-20 18:49:21.296
Measurements
Add Edit Show
Name Value
C1 Frequency 30.000 kHz
C1 PosDuty 20.005 %
C1 Peak2Peak 3.1404 V
C2 Frequency 30.000 kHz
C2 PosDuty 19.973 %
C2 NegDuty 80.027 %
C2 Peak2Peak 374.61 mV
Time
Position: 3.69845 ms
Base: 38.4 us/div
Options
Add Channel
Channel 1 (1±)
Offset: -2.15 V
Range: 500 mV/div
Channel 2 (2±)
Offset: 1.65 V
Range: 500 mV/div
0.85
0.35
-0.15
-0.65
-1.15
-1.65
-2.15
-2.65
-3.15
-3.65
-4.15
3.506 ms
3.545 ms
3.583 ms
3.622 ms
3.66 ms
3.698 ms
3.737 ms
3.775 ms
3.814 ms
3.852 ms
3.89 ms
Manual Trigger
Discovery2 SN:210321AF16D7 USB
Status: OK

Once we generated 30 kHz PWM signal with the correct duty cycle, we connected it to an H-bridge circuit to drive the motor. The H-bridge circuit is connected to 24 Volts and the 3.3 V 30 kHz PWM generated from our microcontroller. The output will be a 24 PWM with 30 kHz frequency. Our H-bridge circuit is shown below.



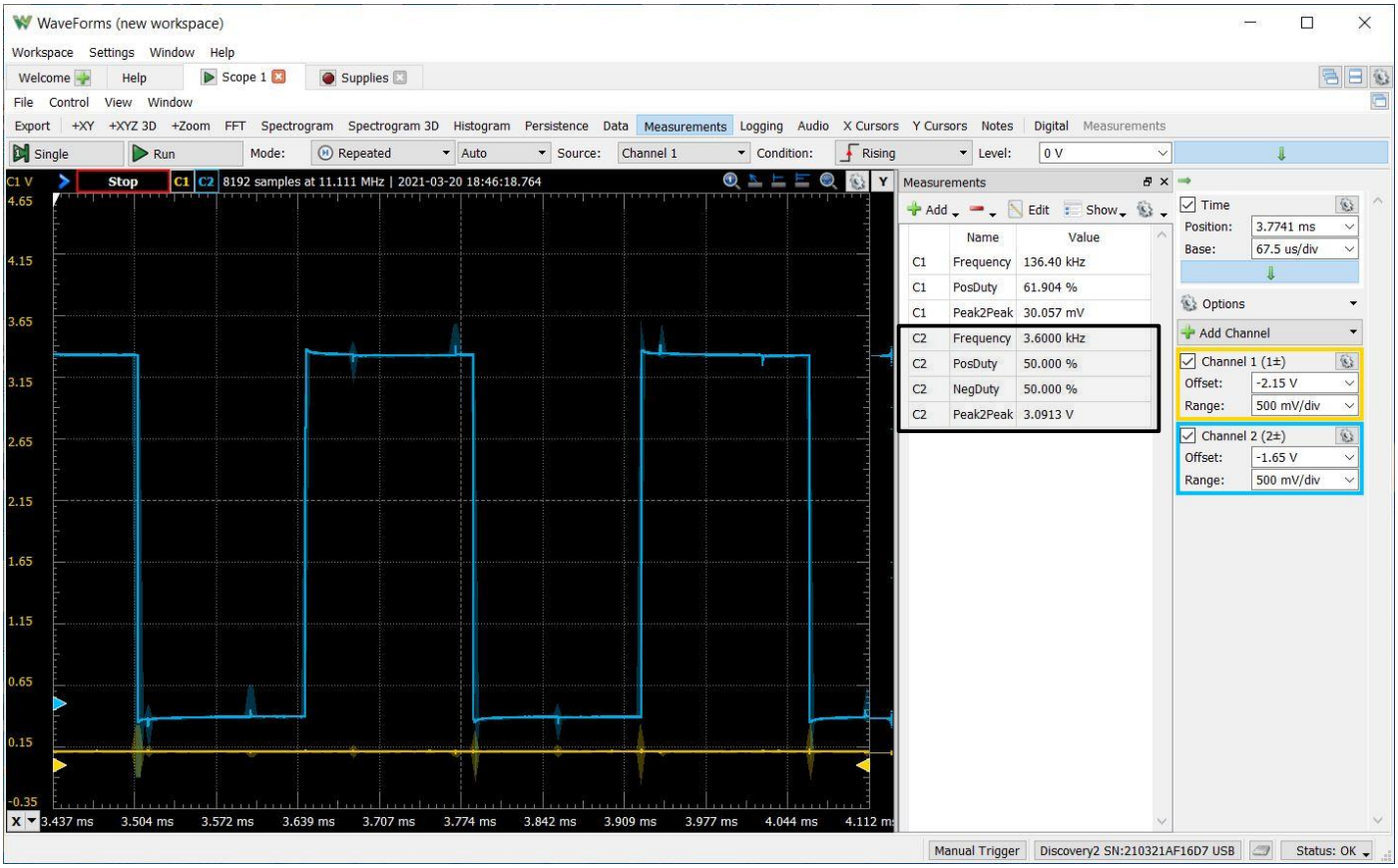
After that, the process must stop the motor when an object is sensed/reached an end position. We first modeled this with an SPST switch so that when a STOP# signal is low (active low), the motor stops. Then after we got that working, we used the signal from photo receiver sensors to control the state of the motor. We used a voltage divider circuit to bring down the 24 V output form the photo receiver sensors to about 3.4 V. Then the 3.4 V output is connected to our STOP# pin in the microcontroller. The photo sensors output 0 V when an object is sensed and 24 V otherwise. Therefore, when STOP# pin is low, we stop our motor (stop PWM signal generation) and then proceed to the next step. Our voltage divider circuit we used is shown on the next page.



The last task we implemented after an object is sensed and the motor is stopped is blinking green LED (LED indicator) 10 time with 2 blinks/sec, sounding the buzzer at 3.5 kHz until LED blinking is over and turning off IR_LED. We used TIM4 channel 1 to generate the PWM for the buzzer. We generated a PWM signal of 3.6 kHz frequency because that gave us a counter of 25000 (We used 25000 instead of 25714 which gives 3.5 kHz).

ARR (counter period) = (F_{lIm} * prescaler) / Freq = 90 M / 3.6kHz = 25,000.

Once the blinking is over, we stop the TIM4 channel 1 PWM interrupt to stop the buzzer. We also turned off IR_LED. The PWM we generated is as follows.

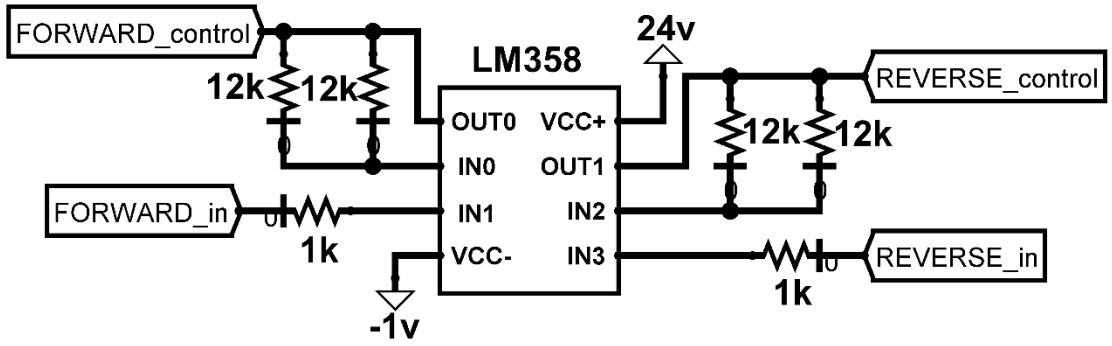


Then the process restarts. We implemented all these 3 tasks inside the while loop so that the process restarts.

Bonus Projects

Forward/Reverse control - Forward/Reverse pins, amplifier circuit and External interrupt

Once we finish implementing the required tasks, we implemented the bonus project tasks into our design. The first one is forward and reverse control. First, we set an external interrupt EXTI15_10. We used pin PG12 to configure this interrupt and called it DIRECTION. The interrupt is triggered with both rising and falling edges. Then we connected this pin into a debounced SPST switch. This switch controls the state of FORWARD and BACKWARD pins on the microcontroller. Inside the interrupt, we developed a code that toggles the FORWARD and BACKWARD pins. We also made sure the two pins are always opposite of one another. After that, we used an amplifier to connect the 3V output form FORWARD and BACKWARD pins to pin 15 and 16 of the motor. The amplifier we used needed to have a gain of approximately 7. We used a non-inverting amplifier and two resistors to adjust the gain. Our amplifier circuit is as follows.



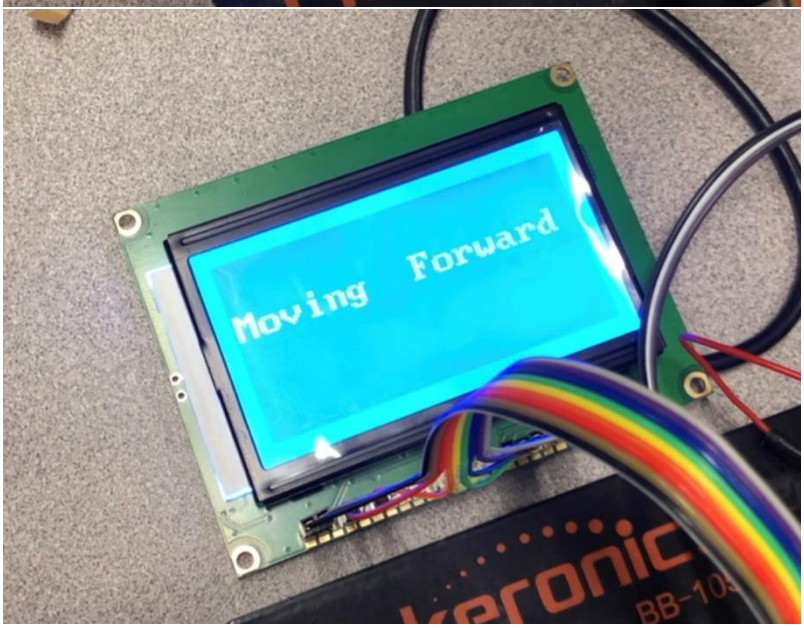
Gain = 1 + Rf/Ri = 1 + (12k || 12k)/1k = 1 + 6k/1k = 7

After adding the amplifier circuit and connecting the amplified FORWARD and BACKWARD voltages to the motor, we successfully operated the belt system moving forward and backward with our switch.

GLCD display – Spi 4 module

After we were able to get direction control, we added a couple of functions to our C program so that we can display the status of the motor on Graphics LCD. We used MOSI (master out slave in), SCL (clock) and RS (register select pin) to send data from our microcontroller (master) to GLCD (slave). We utilized the SPI 4 module of our microcontroller. The functions we implemented are described in the C program section. We used these functions to write the motor status in each step (inside if statements) of the motor motion. Below are some of the messages we displayed on LCD.


Moving Backward
Makeronic


Moving Forward


Halted


Emergency stop

Emergency Stop

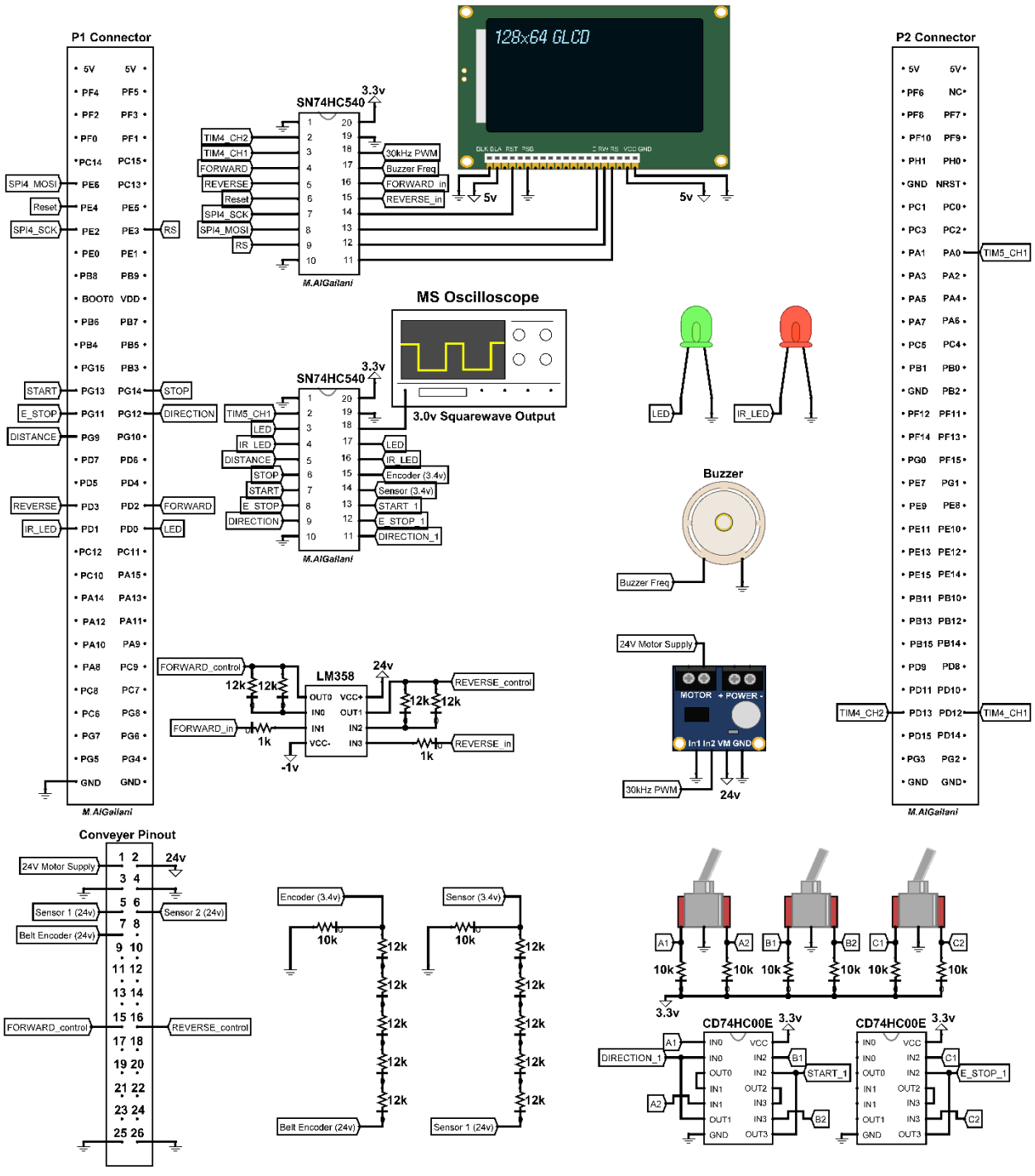
We used a debounced SPST switch to implement an Emergency Stop. The switch is connected to E_STOP# pin on our microcontroller. Whenever that pin is low (active low), we stop all the tasks immediately. An if statement is used at the beginning to check for the status of this pin, and the status is checked again within the various nested process states.

External interrupt to measure distance and Voltage divider circuit for encoder pulse

We attempted to measure distance by using the encoder pulse signal from the motor. First, we stepped down the 24 V pulse to 3 V pulse using our voltage divider circuit. We then connected the 3 V pulse to an external interrupt EXTI9_5 to increment or decrement a counter inside the interrupt. The interrupt is configured to trigger on rising edges. We increase the counter if the object moves forward and decrement if it moves backward. We then multiplied the count by 0.5 inches which is the approximate length of one tile width and tried to display it on GLCD. However, we were not able to get correct ASCII numbers to appear. Our implementation is described in the *further considerations* section after the C program.

Detailed Schematic

Below is a detailed schematic of the whole system, including appropriate buffers for all signals:



C program

The entire process was devised to run in a single infinite while-loop with brief interrupts for certain features. Within this while-loop, the process continually repeats through various stages dependent on the input control signals. – START#, STOP#, E_STOP#, and DIRECTION. Descriptions will be given for each major section of code, along with detailed comments on what the program is doing.

[main.c]

To start off, the `stdio.h` and `string.h` libraries are included:

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */
```

Next, an assortment of statements are defined:

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define START      HAL_GPIO_ReadPin(GPIOG, START_Pin)   // get START value
#define STOP       HAL_GPIO_ReadPin(GPIOG, STOP_Pin)    // get STOP value
#define E_STOP     HAL_GPIO_ReadPin(GPIOG, E_STOP_Pin)  // get EMERGENCY STOP value
#define DIRECTION  HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin) // get DIRECTION value
#define IR_LED_ON  HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_SET)   // IR LED on
#define IR_LED_OFF HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_RESET) // IR LED off
#define LED_ON     HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_SET)   // LED on
#define LED_OFF    HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_RESET) // LED off
/* USER CODE END PD */
```

Then a handful of private variables are created:

```
/* USER CODE BEGIN PV */
 uint8_t  state;                 // state of process
 uint32_t time1, time2;          // input capture times
 uint32_t diff;                  // difference between time1 and time2
 uint32_t f;                     // frequency of input signal
 int16_t  dutycycle;             // duty cycle percentage
 uint16_t highduty;              // high duty cycle portion

 uint8_t array[3];               // array for LCD commands
 uint8_t count;                  // count to track # of belt tiles moved
 uint8_t distance;               // distance of object from end of belt
 uint8_t remainder;              // used for distance output as a string
 uint8_t firstDigit, secondDigit; // distance displayed in two digits
 char    distanceString[2];      // string holding distance value
/* USER CODE END PV */
```

Functions are created to work with the LCD:

```
/* USER CODE BEGIN PFP */
// -----------------------------------------------------------------
// LCD functions

// startup initialization sequence:
void startup()
{
   HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_RESET);
   HAL_Delay(100); // wait >40ms after power is applied
   HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_SET);
   HAL_Delay(10);  // wait >100us
   Sendcmd(0x30);  // wake up
   HAL_Delay(10);  // wait >100us
   Sendcmd(0x30);  // wakeup
   HAL_Delay(10);  // wait >37us
   Sendcmd(0x0C);  // display ON, cursor ON
   HAL_Delay(10);  // wait >100us
   Sendcmd(0x01);  // display clear
   HAL_Delay(15);  // wait >10ms
   Sendcmd(0x06);
   HAL_Delay(10);
}

// send command to LCD:
void Sendcmd(uint8_t cmd)
{
    array[0] = 0xF8;  // send 1111 1000 to sync LCD
    array[1] = (cmd & 0xF0);
    array[2] = ( (cmd << 4) & 0xF0);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
    HAL_SPI_Transmit(&hspi4,array,3,1);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
}

// send data directly to LCD:
void sendData(uint8_t temp)
{
    array[0] = 0xFA;  // send 1111 1010 to write to LCD (Rs=1, RW=0)
    array[1] = (temp & 0xF0);
    array[2] = ( (temp << 4) & 0xF0);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
    HAL_SPI_Transmit(&hspi4,array,3,1);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
}

// write a string onto the LCD:
void writeString(char* string)
{
  int length = strlen(string);
    for(int i = 0; i < length; i++)
    {
      sendData(string[i]);
    }
 }

// -----------------------------------------------------------------
/* USER CODE END PFP */
```

Then the LCD startup runs:

```
/* USER CODE BEGIN 2 */
   startup();
/* USER CODE END 2 */
```

Finally, the while-loop handles the process flow. Note that there are many nested if-else statements. The outermost if-else block is a polling method for detecting the emergency stop, which is active low.

Hence if E_STOP# is high, then the finite state machine will begin carrying out the assigned process. The process runs in three separate states. State 0 sounds a buzzer and blinks an LED to signify the startup sequence. State 1 turns the motor and IR LED on. State 2 contains the bulk of the functionality, such as calculating the captured frequency, updating the PWM motor driving signal, writing to the LCD the conveyer direction, and checking for the STOP# command. Once STOP# goes low, the process halts. The halt sequence runs by sounding the buzzer and blinking an LED. Moreover, an additional LCD message is displays. The process repeats indefinitely so long as START# is low.

Lastly, if E_STOP# is low, then an emergency stop immediately shuts the system down.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
  {
   // first check if an emergency stop (active low) has not been activated (when it is 1):
   if (E_STOP)
   {
      //-------------------------------------------------------------------------
      // state 0:
      if (START == 0 && state == 0 && E_STOP)
      {
          HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);   // start input capture
          TIM4->ARR = 20000-1;                          // set auto reload register
          highduty = 10000;                             // set 50% duty cycle
          HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);  // start buzzer
          for(int i = 0; i < 6; i++)                    // blink LED 6 times
          {
            LED_ON;
            HAL_Delay(500);                             // on for .5 seconds
            LED_OFF;
            HAL_Delay(500);                             // off for .5 seconds
          }
          HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1);   // stop buzzer
          state = 1;                                    // move to next state
      }
      //-------------------------------------------------------------------------
      // state 1:
      if (START == 0 && state == 1 && E_STOP)
      {
          IR_LED_ON;                                    // turn on IR LED while motor moves
          TIM4->ARR = 3000-1;                           // set ARR for 30kHz PWM signal
          highduty = 1500;                              // initialize to 50% duty cycle
          HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_2);  // generate signal to start motor
          state = 2;                                    // move to next state
      }
```

```
//-----------------------------------------------------------------------
// state 2:
if ((state == 2) && E_STOP)
{
   // calculate the frequency, ensuring it is not mistakenly negative
   // (note that the frequency uses the APB1 timer clock of 90MHz):
   if (time1 > time2)
    {
      diff = (time1-time2);
      f = 90000000/diff;
    }
    else if (time2 > time1)
    {
      diff = (time2-time1);
      f = 90000000/diff;
    }

    // update the duty cycle according to the captured frequency
    // (this equation relates duty cycle of PWM to the captured encoder frequency):
    dutycycle = (-f/20 + 325);
    highduty  = (uint16_t)dutycycle*3000/100  + 24;// calculate high portion

    // if reversing, display "Moving Backward" on the LCD:
    if (DIRECTION)
    {
      HAL_Delay(100);                           // 100ms delay
      Sendcmd(0x90);                            // write on line 2
      writeString("Moving Backward");           // display message
    }

    // otherwise display "Moving Forward":
    else
    {
      HAL_Delay(100);                           // 100ms delay
      Sendcmd(0x90);                            // write on line 2
      writeString("Moving Forward");            // display message
    }

    // check if STOP is active. If so, the conveyer belt halts
    // and a buzzer sounds. A message is also displayed:
    if (STOP == 0) // STOP is active low
    {
      HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
      HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture

      TIM4->ARR = 25000-1;                      // update auto reload register
      highduty = 12500;                         // set 50% duty cycle
      HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);// start buzzer

      Sendcmd(0x01);                            // clear screen
      HAL_Delay(100);                           // brief delay
      Sendcmd(0x90);                            // send command to write on line 2
      writeString("Halted");                    // display "Halted" message

      for(int i = 0; i < 10; i++)               // blink LED 10 times
      {
        LED_ON;
        HAL_Delay(250);                         // on for .5 seconds
        LED_OFF;
        HAL_Delay(250);                         // off for .5 seconds
      }
```

```
            IR_LED_OFF;                                  // turn off IR LED
            HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
            state = 0;                                   // reset state to 0
            Sendcmd(0x01);                               // clear screen
          }
      }
      //---------------------------------------------------------------------
   }

   // if emergency stop activated (active low), immediately shut off:
   else
   {
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
        HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture
        LED_OFF;                                    // turn off LED
        IR_LED_OFF;                                 // turn off IR LED
        state = 0;                                  // return to state 0
        HAL_Delay(100);                             // 100ms delay
        Sendcmd(0x90);                              // write on line 2
        writeString("Emergency stop ");             // display message
   }

    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

[stm32f4xx_it.c]

Interrupts are used for three purposes. First, an external interrupt on lines 10 – 15 toggles the direction. Second, the TIM4 module controls the duty cycle of two PWM channels which either sound the buzzer (channel 1) or drive the conveyer belt motor (channel 2). Third, the TIM5 module is set to input capture for recording the motor encoder input frequency.

The interrupt file begins with external variable declarations:

```
/* USER CODE BEGIN EV */
extern uint32_t time1, time2;
extern uint16_t highduty;
extern uint8_t  count;
/* USER CODE END EV */
```

Then moves on to the external interrupt request handler for lines 10 – 15:

```
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */
  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  // toggle the REVERSE pin:
  HAL_GPIO_TogglePin(GPIOD, REVERSE_Pin);
  // update the FORWARD pin accordingly:
  if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
```

```
  {
    HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_RESET);
  }
  else
  {
    HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_SET);
  }
  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

Followed by the PWM duty cycle control for the buzzer and motor driving signals:

```
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */
  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */
    if (TIM4->ARR == 19999)      // if using startup buzzer:
    {
      TIM4->CCR1 = highduty;     // set CCR1 to relevant highduty value
      TIM4->CCR2 = 0;            // set CCR2 to zero
    }
    else if (TIM4->ARR == 24999) // if using stop sequence buzzer:
    {
      TIM4->CCR1 = highduty;     // set CCR1 to relevant highduty value
      TIM4->CCR2 = 0;            // set CCR2 to zero
    }
    else                         // otherwise generate 30kHz motor signal:
    {
      TIM4->CCR1 = 0;            // set CCR1 to zero
      TIM4->CCR2 = highduty;     // update CCR2 high portion of duty cycle
    }
  /* USER CODE END TIM4_IRQn 1 */
}
```

Finally, the input capture collects two timestamps from the wavegen input:

```
void TIM5_IRQHandler(void)
{
  /* USER CODE BEGIN TIM5_IRQn 0 */
  /* USER CODE END TIM5_IRQn 0 */
  HAL_TIM_IRQHandler(&htim5);
  /* USER CODE BEGIN TIM5_IRQn 1 */
    // if the first time hasn't been captured yet:
    if(temp2 == 0) {
      time1 = TIM5->CCR1; // get first timestamp
      temp2 = 1;          // prepare for second timestamp
    }
    // otherwise, if the first value has been captured already:
    else if (temp2 == 1) {
      time2 = TIM5->CCR1; // get second timestamp
      temp2 = 0;          // prepare for new timestamp
    }
  /* USER CODE END TIM5_IRQn 1 */
}
```

That concludes the C program.

Further Considerations

We also attempted to calculate the objects distance using the conveyer belt tiles. Since the system outputs a 24v pulse for every tile moved, and it takes roughly 25 tiles to reach the end of the belt, then a few simple calculations can reasonably estimate the distance traveled by the object. However, our implementation did not operate as expected. For this reason, the code was removed from the previous `main.c` and `stm32f4xx_it.c` descriptions. Yet for the sake of completion, here is a further description of that removed code.

[main.c]

The primary change to `main.c` is to include the distance calculations, conversion to ASCII, and LCD display commands. The distance is calculated as `(count % 25) >> 1`. The modulus restarts the distance after every 25 tiles moved, and since the tiles were measured to be roughly half an inch long, we can right shift which is the equivalent of dividing by two or multiplying by 0.5 inches. Two methods were tested separately:

```
    // method 1:
    distance = ((count % 25) >> 1) + 0x30;     // calculate distance
    Sendcmd(0x88);                             // write on line 3
    sprintf(distanceString, "%x", distance);   // convert distance to a string
    writeString(distanceString);               // transmit string to LCD

    // method 2:
    distance = (count % 25) >> 1;              // calculate distance
    remainder = distance;                      // assign distance to remainder
    firstDigit = remainder%10;                 // get first digit  (lsb)
    secondDigit = (remainder - firstDigit)/10; // get second digit (msb)
    sendData(secondDigit + 0x30);              // transmit second ASCII number
    sendData(firstDigit  + 0x30);              // transmit first ASCII number
```

[stm32f4xx_it.c]

An additional external interrupt would be used to increment or decrement a counter for every tile moved. The decision to increase or decrease the count would be determined by the conveyer belt direction. With this information we can estimate the net distance traveled, as shown above.

```
void EXTI9_5_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI9_5_IRQn 0 */

  /* USER CODE END EXTI9_5_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_9);
  /* USER CODE BEGIN EXTI9_5_IRQn 1 */
  if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
  {
    count--;
  }
  else
  {
    count++;
  }
  /* USER CODE END EXTI9_5_IRQn 1 */
}
```

Conclusion

Throughout the course of this project, we were met with many difficulties. Yet we ultimately succeeded in completing the main design with additional bonus work included. We gained experience with new more advanced circuit elements throughout the project and used what we have learned in previous labs to see it to fruition. Moreover, we learned about how debounced SPST switches, an H-bridge motor driver and a buzzer (specifically PSR-29F08S02-JQ speaker) work. We also learned about the Fischertek Conveyor Belt. We used our knowledge of voltage dividers, amplifier circuitry, timer input capture, timer PWM generation and SPI communication with graphical LCD to complete the project. Overall, it was a great learning experience.

"If you want something you never had, you have to do something you've never done"

-Thomas Jefferson

Appendix

Code

[main.c]

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define START      HAL_GPIO_ReadPin(GPIOG, START_Pin)  // get START value
#define STOP       HAL_GPIO_ReadPin(GPIOG, STOP_Pin)   // get STOP value
#define E_STOP     HAL_GPIO_ReadPin(GPIOG, E_STOP_Pin) // get EMERGENCY STOP value
#define IR_LED_ON  HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_SET)   // IR LED off
#define IR_LED_OFF HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_RESET) // IR LED off
#define LED_ON     HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_SET)   // LED off
#define LED_OFF    HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_RESET) // LED off
/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi4;
TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim5;
```

```
/* USER CODE BEGIN PV */
 uint8_t  state;                 // state of process
 uint32_t time1, time2;          // input capture times
 uint32_t diff;                  // difference between time1 and time2
 uint32_t f;                     // frequency of input signal
 int16_t  dutycycle;             // duty cycle percentage
 uint16_t highduty, lowduty;     // high & low duty cycle values

 uint8_t array[3];               // array for LCD commands
 uint8_t count;                  // count to track # of belt tiles moved
 uint8_t distance;               // distance object is from end of belt
 uint8_t remainder;              // used for distance output as a string
 uint8_t firstDigit, secondDigit; // distance displayed in two digits
 char distanceString[2];         // string holding distance value
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM5_Init(void);
static void MX_TIM4_Init(void);
static void MX_SPI4_Init(void);
/* USER CODE BEGIN PFP */

// LCD functions:
void Sendcmd(uint8_t cmd)
{
    array[0] = 0xF8;     //send 1111 1000 to sync LCD
    array[1] = (cmd & 0xF0);
    array[2] = ( (cmd << 4) & 0xF0);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
    HAL_SPI_Transmit(&hspi4,array,3,1);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
}

void sendData(uint8_t temp)
{
    array[0] = 0xFA;  //send 1111 1010 to write to LCD (Rs=1, RW=0)
    array[1] = (temp & 0xF0);
    array[2] = ( (temp << 4) & 0xF0);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
    HAL_SPI_Transmit(&hspi4,array,3,1);
    HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
}

void startup()
{
   HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_RESET);
   HAL_Delay(100); //Wait >40ms after power is applied
   HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_SET);
   HAL_Delay(10); //wait > 100us
   Sendcmd(0x30); //wake up
   HAL_Delay(10); //wait > 100us
   Sendcmd(0x30); //wakeup
   HAL_Delay(10); //wait >37us
   Sendcmd(0x0C); //Display ON Cursor on
   HAL_Delay(10); //Wait >100us
   Sendcmd(0x01); //Display Clear
   HAL_Delay(15); //wait >10ms
   Sendcmd(0x06);
   HAL_Delay(10);
```

```
}

void writeString(char* string)
{
  int length = strlen(string);
    for(int i = 0; i < length; i++)
    {
      sendData(string[i]);
    }
 }

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

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

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM5_Init();
  MX_TIM4_Init();
  MX_SPI4_Init();
  /* USER CODE BEGIN 2 */
  // Initialize the LCD:
  startup();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
   // first check if an emergency stop has been activated:
   if (E_STOP)
   {
```

```
    //---------------------------------------------------------------------
    // state 0:
    if (START == 0 && state == 0 && E_STOP)
    {
        HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);   // start input capture
        TIM4->ARR = 20000-1;                          // set auto reload register
        highduty = 10000;                             // set 50% duty cycle
//      lowduty = 10000;
        HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);  // start buzzer
        for(int i = 0; i < 6; i++)                    // blink LED 6 times
        {
          LED_ON;
          HAL_Delay(500);                             // on for .5 seconds
          LED_OFF;
          HAL_Delay(500);                             // off for .5 seconds
        }
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1);   // stop buzzer
        state = 1;                                    // move to next state
    }
    //---------------------------------------------------------------------
    // state 1:
    if (START == 0 && state == 1 && E_STOP)
    {
        IR_LED_ON;                                    // turn on IR LED while motor moves
        TIM4->ARR = 3000-1;                           // set ARR for 30kHz PWM signal
        highduty = 1500;                              // initialize to 50% duty cycle
//      lowduty = 1500;
        HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_2);  // generate signal to start motor
        state = 2;                                    // move to next state
    }
    //---------------------------------------------------------------------
    // state 2:
    if ((state == 2) && E_STOP)
    {
       // calculate the frequency:
       if (time1 > time2)
        {
          diff = (time1-time2);         // period calculated by difference in time
          f = 90000000/diff;            // ABP1 timer clock to get frequency
        }
        else if (time2 > time1)
        {
          diff = (time2-time1);         // period calculated to always be positive
          f = 90000000/diff;            // again, use ABP1 clock for frequency
        }

        // update the duty cycle according to the captured frequency:
        dutycycle = (-f/20 + 325);                    // calculate cycle length
        highduty  = (uint16_t)dutycycle*3000/100 + 24; // calculate high portion
 //     lowduty   = (100 - (uint16_t)dutycycle)*3000/100 - 24;

        // if reversing, display "Moving Backward" on the LCD:
        if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
        {
        HAL_Delay(100);                 // 100ms delay
        Sendcmd(0x90);                  // write on line 2
        writeString("Moving Backward"); // display message
        // method 1:
  //      distance = (int)0.5*count + 0x30;        // calculate distance
  //      distance = ((count % 25) >> 1);          // calculate distance
  //      Sendcmd(0x88);                           // write on line 3
```

```
//      sprintf(distanceString, "%x", distance);
//      writeString(distanceString);
      // method 2:
//      remainder = distance;
//      firstDigit = remainder%10;
//      remainder = (remainder - firstDigit)/10;
//      secondDigit = remainder;
//      sendData(secondDigit + 0x30);
//      sendData(firstDigit  + 0x30);
      }

      // otherwise display "Moving Forward":
      else
      {
      HAL_Delay(100);               // 100ms delay
      Sendcmd(0x90);                // write on line 2
      writeString("Moving Forward"); // display message
//      Sendcmd(0x88);
//      distance = (count % 25 );
//      distance = 'y';
//      sprintf(distanceString, "%x", distance);
//      writeString(distanceString);
      }

      // check if STOP is active. If so, the conveyer belt halts
      // and a buzzer sounds. A message is also displayed:
      if (STOP == 0) // STOP is active low
      {
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
        HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture

        TIM4->ARR = 25000-1;                        // update auto reload register
        highduty = 12500;                           // set 50% duty cycle
//      lowduty = 12500;
        HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);// start buzzer

        Sendcmd(0x01);                              // clear screen;
        HAL_Delay(100);                             // brief delay
        Sendcmd(0x90);                              // send command to write on line 2
        writeString("Halted");                      // display "Halted" message

        for(int i = 0; i < 10; i++)                 // blink LED 10 times
        {
          LED_ON;
          HAL_Delay(250);                           // on for .5 seconds
          LED_OFF;
          HAL_Delay(250);                           // off for .5 seconds
        }
        IR_LED_OFF;                                 // turn off IR LED
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
        state = 0;                                  // reset state to 0
        Sendcmd(0x01);                              // clear screen;
      }
    }
    //-----------------------------------------------------------------------
}

// if emergency stop activated (active low), immediately shut off:
else
{
     HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
```

```
        HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
        HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture
        LED_OFF;                                    // turn off LED
        IR_LED_OFF;                                 // turn off IR LED
        state = 0;                                  // return to state 0
        HAL_Delay(100);                             // 100ms delay
        Sendcmd(0x90);                              // write on line 2
        writeString("Emergency stop ");             // display message
    }

     /* USER CODE END WHILE */
     /* USER CODE BEGIN 3 */
   }
   /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 180;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Activate the Over-Drive mode
  */
  if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
```

```
    Error_Handler();
  }
}

/**
  * @brief SPI4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI4_Init(void)
{

  /* USER CODE BEGIN SPI4_Init 0 */

  /* USER CODE END SPI4_Init 0 */

  /* USER CODE BEGIN SPI4_Init 1 */

  /* USER CODE END SPI4_Init 1 */
  /* SPI4 parameter configuration*/
  hspi4.Instance = SPI4;
  hspi4.Init.Mode = SPI_MODE_MASTER;
  hspi4.Init.Direction = SPI_DIRECTION_2LINES;
  hspi4.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi4.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi4.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi4.Init.NSS = SPI_NSS_SOFT;
  hspi4.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;
  hspi4.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi4.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi4.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi4.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi4) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI4_Init 2 */

  /* USER CODE END SPI4_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_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {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 = 3000-1;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_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();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */
  HAL_TIM_MspPostInit(&htim4);

}

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

  /* USER CODE BEGIN TIM5_Init 0 */

  /* USER CODE END TIM5_Init 0 */

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

  /* USER CODE BEGIN TIM5_Init 1 */
```

```
  /* USER CODE END TIM5_Init 1 */
  htim5.Instance = TIM5;
  htim5.Init.Prescaler = 0;
  htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim5.Init.Period = 4294967295;
  htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_IC_Init(&htim5) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim5, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM5_Init 2 */

  /* USER CODE END TIM5_Init 2 */

}

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

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

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, RS_Pin|Reset_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
```

```
  HAL_GPIO_WritePin(GPIOD, LED_Pin|IR_LED_Pin|FORWARD_Pin|REVERSE_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : RS_Pin Reset_Pin */
  GPIO_InitStruct.Pin = RS_Pin|Reset_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pins : LED_Pin IR_LED_Pin FORWARD_Pin REVERSE_Pin */
  GPIO_InitStruct.Pin = LED_Pin|IR_LED_Pin|FORWARD_Pin|REVERSE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : DISTANCE_Pin */
  GPIO_InitStruct.Pin = DISTANCE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DISTANCE_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : E_STOP_Pin START_Pin STOP_Pin */
  GPIO_InitStruct.Pin = E_STOP_Pin|START_Pin|STOP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

  /*Configure GPIO pin : DIRECTION_Pin */
  GPIO_InitStruct.Pin = DIRECTION_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DIRECTION_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);

  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

}

/* 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 */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

[stm32f4xx_it.c]

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* 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 */
  uint8_t temp0, temp1, temp2;
/* 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 htim4;
```

```
extern TIM_HandleTypeDef htim5;
/* USER CODE BEGIN EV */
extern uint32_t time1, time2;
extern uint16_t highduty, lowduty;
extern uint8_t  count;
/* 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 EXTI line[9:5] interrupts.
  */
void EXTI9_5_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI9_5_IRQn 0 */

  /* USER CODE END EXTI9_5_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_9);
  /* USER CODE BEGIN EXTI9_5_IRQn 1 */
      if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
      {
        count--;
      }
      else
      {
        count++;
      }
  /* USER CODE END EXTI9_5_IRQn 1 */
}

/**
  * @brief This function handles TIM4 global interrupt.
  */
void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */

  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

    if (TIM4->ARR == 19999)      // if using startup buzzer:
    {
      TIM4->CCR1 = highduty;     // set CCR1 to relevant highduty value
      TIM4->CCR2 = 0;            // set CCR2 to zero
    }
    else if (TIM4->ARR == 24999) // if using stop sequence buzzer:
```

```
    {
      TIM4->CCR1 = highduty;     // set CCR1 to relevant highduty value
      TIM4->CCR2 = 0;            // set CCR2 to zero
    }
    else                         // otherwise generate 30kHz motor signal:
    {
      TIM4->CCR1 = 0;            // set CCR1 to zero
      TIM4->CCR2 = highduty;     // update CCR2 high portion of duty cycle
    }
  /* USER CODE END TIM4_IRQn 1 */
}

/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  // toggle the REVERSE pin:
  HAL_GPIO_TogglePin(GPIOD, REVERSE_Pin);
  // update the FORWARD pin accordingly:
  if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
  {
    HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_RESET);
  }
  else
  {
    HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_SET);
  }

  /* USER CODE END EXTI15_10_IRQn 1 */
}

/**
  * @brief This function handles TIM5 global interrupt.
  */
void TIM5_IRQHandler(void)
{
  /* USER CODE BEGIN TIM5_IRQn 0 */

  /* USER CODE END TIM5_IRQn 0 */
  HAL_TIM_IRQHandler(&htim5);
  /* USER CODE BEGIN TIM5_IRQn 1 */

    // if the first time hasn't been captured yet:
    if(temp2 == 0) {
      time1 = TIM5->CCR1; // get first timestamp
      temp2 = 1;          // prepare for next timestamp
    }
    // otherwise, if the first value has been captured already:
    else if (temp2 == 1) {
      time2 = TIM5->CCR1; // get second timestamp
      temp2 = 0;          // prepare for next timestamp
    }

  /* USER CODE END TIM5_IRQn 1 */
```

```
}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
/************************  (C) COPYRIGHT STMicroelectronics  *****END OF FILE****/
```

List File

[main.lst]

```
###############################################################################
#
# IAR ANSI C/C++ Compiler V8.50.9.278/W32 for ARM         21/Mar/2021  21:35:17
# Copyright 1999-2020 IAR Systems AB.
#
#    Cpu mode
#    Endian                  =  little
#    Source file             =  C:\Users\Mitch\Desktop\HW4\Core\Src\main.c
#    Command line            =
#        -f C:\Users\Mitch\AppData\Local\Temp\EW179C.tmp
#        (C:\Users\Mitch\Desktop\HW4\Core\Src\main.c -D USE_HAL_DRIVER -D
#        STM32F429xx -lC C:\Users\Mitch\Desktop\HW4\EWARM\HW4\List -o
#        C:\Users\Mitch\Desktop\HW4\EWARM\HW4\Obj --debug --endian=little
#        --cpu=Cortex-M4 -e --fpu=VFPv4_sp --dlib_config "C:\Program Files
#        (x86)\IAR Systems\Embedded Workbench 8.4\arm\inc\c\DLib_Config_Full.h"
#        -I C:\Users\Mitch\Desktop\HW4\EWARM/../Core/Inc\\ -I
#        C:\Users\Mitch\Desktop\HW4\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc\\
#        -I
#        C:\Users\Mitch\Desktop\HW4\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\\
#        -I
#        C:\Users\Mitch\Desktop\HW4\EWARM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\\
#        -I C:\Users\Mitch\Desktop\HW4\EWARM/../Drivers/CMSIS/Include\\ -Ohz)
#    Locale                  =  C
#    List file               =
#        C:\Users\Mitch\Desktop\HW4\EWARM\HW4\List\main.lst
#    Object file             =
#        C:\Users\Mitch\Desktop\HW4\EWARM\HW4\Obj\main.o
#    Runtime model:
#      __SystemLibrary        =  DLib
#      __dlib_file_descriptor =  1
#      __dlib_version         =  6
#      __size_limit           =  32768|ARM.EW.LINKER
#
###############################################################################

C:\Users\Mitch\Desktop\HW4\Core\Src\main.c
      1          /* USER CODE BEGIN Header */
      2          /**
      3
******************************************************************************
      4            * @file           : main.c
      5            * @brief          : Main program body
      6
******************************************************************************
      7            * @attention
      8            *
      9            * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
     10            * All rights reserved.</center></h2>
     11            *
     12            * This software component is licensed by ST under BSD 3-Clause license,
     13            * the "License"; You may not use this file except in compliance with the
     14            * License. You may obtain a copy of the License at:
     15            *                        opensource.org/licenses/BSD-3-Clause
     16            *
```

```
     17
**********************************************************************
     18            */
     19          /* USER CODE END Header */
     20          /* Includes ------------------------------------------------------------
-*/
     21          #include "main.h"
     22
     23          /* Private includes ----------------------------------------------------
-*/
     24          /* USER CODE BEGIN Includes */
     25          #include <stdio.h>
     26          #include <string.h>
     27          /* USER CODE END Includes */
     28
     29          /* Private typedef -----------------------------------------------------
-*/
     30          /* USER CODE BEGIN PTD */
     31
     32          /* USER CODE END PTD */
     33
     34          /* Private define ------------------------------------------------------
-*/
     35          /* USER CODE BEGIN PD */
     36          #define START      HAL_GPIO_ReadPin(GPIOG, START_Pin)  // get START value
     37          #define STOP       HAL_GPIO_ReadPin(GPIOG, STOP_Pin)   // get STOP value
     38          #define E_STOP     HAL_GPIO_ReadPin(GPIOG, E_STOP_Pin) // get EMERGENCY STOP
value
     39          #define IR_LED_ON  HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_SET)   // IR
LED off
     40          #define IR_LED_OFF HAL_GPIO_WritePin(GPIOD, IR_LED_Pin, GPIO_PIN_RESET) // IR
LED off
     41          #define LED_ON     HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_SET)   //
LED off
     42          #define LED_OFF    HAL_GPIO_WritePin(GPIOD, LED_Pin,    GPIO_PIN_RESET) //
LED off
     43          /* USER CODE END PD */
     44
     45          /* Private macro -------------------------------------------------------
-*/
     46          /* USER CODE BEGIN PM */
     47
     48          /* USER CODE END PM */
     49
     50          /* Private variables ---------------------------------------------------
-*/

   \                                In section .bss, align 4
     51          SPI_HandleTypeDef hspi4;
   \                        hspi4:
   \        0x0                     DS8 88
     52

   \                                In section .bss, align 4
     53          TIM_HandleTypeDef htim4;
     54          TIM_HandleTypeDef htim5;
     55
     56          /* USER CODE BEGIN PV */
     57           uint8_t  state;               // state of process
   \                        state:
   \        0x0                     DS8 1
```

```
\        0x1                        DS8 1
  58           uint32_t time1, time2;         // input capture times
  59           uint32_t diff;                 // difference between time1 and time2
  60           uint32_t f;                    // frequency of input signal
  61           int16_t  dutycycle;            // duty cycle percentage
\                      dutycycle:
\        0x2                        DS8 2
  62           uint16_t highduty, lowduty;    // high & low duty cycle values
\                      highduty:
\        0x4                        DS8 2
\                      lowduty:
\        0x6                        DS8 2
\                      htim4:
\        0x8                        DS8 72
\                      htim5:
\       0x50                        DS8 72
\                      time1:
\       0x98                        DS8 4
\                      time2:
\       0x9C                        DS8 4
\                      diff:
\       0xA0                        DS8 4
\                      f:
\       0xA4                        DS8 4
  63

\                                  In section .bss, align 4
  64           uint8_t array[3];              // array for LCD commands
\                      array:
\        0x0                        DS8 4

\                                  In section .bss, align 1
  65           uint8_t count;                 // count to track # of belt tiles moved
\                      count:
\        0x0                        DS8 1

\                                  In section .bss, align 1
  66           uint8_t distance;              // distance object is from end of belt
\                      distance:
\        0x0                        DS8 1

\                                  In section .bss, align 1
  67           uint8_t remainder;             // used for distance output as a string
\                      remainder:
\        0x0                        DS8 1

\                                  In section .bss, align 1
  68           uint8_t firstDigit, secondDigit; // distance displayed in two digits
\                      firstDigit:
\        0x0                        DS8 1

\                                  In section .bss, align 1
\                      secondDigit:
\        0x0                        DS8 1

\                                  In section .bss, align 2
  69           char distanceString[2];        // string holding distance value
\                      distanceString:
\        0x0                        DS8 2
  70         /* USER CODE END PV */
  71
```

```
     72          /* Private function prototypes -----------------------------------------------
-*/
     73          void SystemClock_Config(void);
     74          static void MX_GPIO_Init(void);
     75          static void MX_TIM5_Init(void);
     76          static void MX_TIM4_Init(void);
     77          static void MX_SPI4_Init(void);
     78          /* USER CODE BEGIN PFP */
     79
     80          // LCD functions:

   \                                In section .text, align 2, keep-with-next
     81          void Sendcmd(uint8_t cmd)
     82          {
   \                         Sendcmd: (+1)
   \        0x0   0xB538              PUSH     {R3-R5,LR}
     83              array[0] = 0xF8;     //send 1111 1000 to sync LCD
   \        0x2   0x....'....         LDR.W    R4,??DataTable8
   \        0x6   0x21F8              MOVS     R1,#+248
     84              array[1] = (cmd & 0xF0);
   \        0x8   0x....              B.N      ?Subroutine0
     85              array[2] = ( (cmd << 4) & 0xF0);
     86              HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
     87              HAL_SPI_Transmit(&hspi4,array,3,1);
     88              HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
     89          }

   \                                In section .text, align 2, keep-with-next
   \                         ?Subroutine0: (+1)
   \        0x0   0xF000 0x02F0       AND      R2,R0,#0xF0
   \        0x4   0x0100              LSLS     R0,R0,#+4
   \        0x6   0x7021              STRB     R1,[R4, #+0]
   \        0x8   0x7062              STRB     R2,[R4, #+1]
   \        0xA   0x70A0              STRB     R0,[R4, #+2]
   \        0xC   0x2201              MOVS     R2,#+1
   \        0xE   0x....'....         LDR.W    R5,??DataTable8_1  ;; 0x40021000
   \       0x12   0x2108              MOVS     R1,#+8
   \       0x14   0x....'....         BL       ??Subroutine1_1
   \                         ??CrossCallReturnLabel_4: (+1)
   \       0x18   0x2301              MOVS     R3,#+1
   \       0x1A   0x2203              MOVS     R2,#+3
   \       0x1C   0x4621              MOV      R1,R4
   \       0x1E   0x....'....         LDR.W    R0,??DataTable8_2
   \       0x22   0x....'....         BL       HAL_SPI_Transmit
   \       0x26   0x4628              MOV      R0,R5
   \       0x28   0xE8BD 0x4038       POP      {R3-R5,LR}
   \       0x2C   0x2200              MOVS     R2,#+0
   \       0x2E   0x2108              MOVS     R1,#+8
   \       0x30   0x....'....         B.W      HAL_GPIO_WritePin

   \                                In section .text, align 2, keep-with-next
   \                         ?Subroutine1: (+1)
   \        0x0   0x2200              MOVS     R2,#+0
   \                         ??Subroutine1_0: (+1)
   \        0x2   0x2101              MOVS     R1,#+1
   \                         ??Subroutine1_1: (+1)
   \        0x4   0x4628              MOV      R0,R5
   \        0x6   0x....'....         B.W      HAL_GPIO_WritePin
     90

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

```
  91          void sendData(uint8_t temp)
  92          {
\                     sendData: (+1)
\        0x0   0xB538             PUSH     {R3-R5,LR}
  93              array[0] = 0xFA;  //send 1111 1010 to write to LCD (Rs=1, RW=0)
\        0x2   0x....'....        LDR.W    R4,??DataTable8
\        0x6   0x21FA             MOVS     R1,#+250
  94              array[1] = (temp & 0xF0);
\        0x8                      REQUIRE ?Subroutine0
\        0x8                      ;; // Fall through to label ?Subroutine0
  95              array[2] = ( (temp << 4) & 0xF0);
  96              HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_SET);
  97              HAL_SPI_Transmit(&hspi4,array,3,1);
  98              HAL_GPIO_WritePin(GPIOE,RS_Pin,GPIO_PIN_RESET);
  99          }
 100

\                                 In section .text, align 2, keep-with-next
 101          void startup()
 102          {
\                     startup: (+1)
\        0x0   0xB510             PUSH     {R4,LR}
 103            HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_RESET);
\        0x2   0x....'....        LDR.W    R4,??DataTable8_1  ;; 0x40021000
\        0x6   0x2200             MOVS     R2,#+0
\        0x8   0x....'....        BL       ?Subroutine4
 104            HAL_Delay(100); //Wait >40ms after power is applied
\                     ??CrossCallReturnLabel_21: (+1)
\        0xC   0x....'....        BL       ?Subroutine2
 105            HAL_GPIO_WritePin(GPIOE,Reset_Pin,GPIO_PIN_SET);
\                     ??CrossCallReturnLabel_14: (+1)
\       0x10   0x2201             MOVS     R2,#+1
\       0x12   0x....'....        BL       ?Subroutine4
 106            HAL_Delay(10); //wait > 100us
\                     ??CrossCallReturnLabel_20: (+1)
\       0x16   0x....'....        BL       ?Subroutine3
 107            Sendcmd(0x30); //wake up
\                     ??CrossCallReturnLabel_18: (+1)
\       0x1A   0x2030             MOVS     R0,#+48
\       0x1C   0x....'....        BL       Sendcmd
 108            HAL_Delay(10); //wait > 100us
\       0x20   0x....'....        BL       ?Subroutine3
 109            Sendcmd(0x30); //wakeup
\                     ??CrossCallReturnLabel_17: (+1)
\       0x24   0x2030             MOVS     R0,#+48
\       0x26   0x....'....        BL       Sendcmd
 110            HAL_Delay(10); //wait >37us
\       0x2A   0x....'....        BL       ?Subroutine3
 111            Sendcmd(0x0C); //Display ON Cursor on
\                     ??CrossCallReturnLabel_16: (+1)
\       0x2E   0x200C             MOVS     R0,#+12
\       0x30   0x....'....        BL       Sendcmd
 112            HAL_Delay(10); //Wait >100us
\       0x34   0x....'....        BL       ?Subroutine3
 113            Sendcmd(0x01); //Display Clear
\                     ??CrossCallReturnLabel_15: (+1)
\       0x38   0x2001             MOVS     R0,#+1
\       0x3A   0x....'....        BL       Sendcmd
 114            HAL_Delay(15); //wait >10ms
\       0x3E   0x200F             MOVS     R0,#+15
\       0x40   0x....'....        BL       HAL_Delay
```

```
  115             Sendcmd(0x06);
  \       0x44   0x2006             MOVS     R0,#+6
  \       0x46   0x....'....        BL       Sendcmd
  116             HAL_Delay(10);
  \       0x4A   0xE8BD 0x4010      POP      {R4,LR}
  \       0x4E   0x200A             MOVS     R0,#+10
  \       0x50   0x....'....        B.W      HAL_Delay
  117           }

  \                                 In section .text, align 2, keep-with-next
  \                       ?Subroutine4: (+1)
  \        0x0   0x2110             MOVS     R1,#+16
  \                       ??Subroutine4_0: (+1)
  \        0x2   0x4620             MOV      R0,R4
  \        0x4   0x....'....        B.W      HAL_GPIO_WritePin

  \                                 In section .text, align 2, keep-with-next
  \                       ?Subroutine3: (+1)
  \        0x0   0x200A             MOVS     R0,#+10
  \        0x2   0x....'....        B.W      HAL_Delay

  \                                 In section .text, align 2, keep-with-next
  \                       ?Subroutine2: (+1)
  \        0x0   0x2064             MOVS     R0,#+100
  \        0x2   0x....'....        B.W      HAL_Delay
  118

  \                                 In section .text, align 2, keep-with-next
  119           void writeString(char* string)
  120           {
  \                       writeString: (+1)
  \        0x0   0xB570             PUSH     {R4-R6,LR}
  \        0x2   0x4604             MOV      R4,R0
  121            int length = strlen(string);
  \        0x4   0x....'....        BL       strlen
  \        0x8   0x4605             MOV      R5,R0
  122              for(int i = 0; i < length; i++)
  \        0xA   0x2600             MOVS     R6,#+0
  \        0xC   0xE003             B.N      ??writeString_0
  123              {
  124                sendData(string[i]);
  \                       ??writeString_1: (+1)
  \        0xE   0x5DA0             LDRB     R0,[R4, R6]
  \       0x10   0x....'....        BL       sendData
  125              }
  \       0x14   0x1C76             ADDS     R6,R6,#+1
  \                       ??writeString_0: (+1)
  \       0x16   0x42AE             CMP      R6,R5
  \       0x18   0xDBF9             BLT.N    ??writeString_1
  126           }
  \       0x1A   0xBD70             POP      {R4-R6,PC}       ;; return
  127
  128           /* USER CODE END PFP */
  129
  130           /* Private user code ---------------------------------------------------------
-*/
  131           /* USER CODE BEGIN 0 */
  132
  133           /* USER CODE END 0 */
  134
  135           /**
```

```
    136            * @brief  The application entry point.
    137            * @retval int
    138            */

   \                                In section .text, align 4, keep-with-next
    139          int main(void)
    140          {
   \                        main: (+1)
   \        0x0   0xE92D 0x4FF0      PUSH     {R4-R11,LR}
   \        0x4   0xB08D             SUB      SP,SP,#+52
    141            /* USER CODE BEGIN 1 */
    142
    143            /* USER CODE END 1 */
    144
    145            /* MCU Configuration--------------------------------------------------------
-*/
    146
    147            /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
    148            HAL_Init();
   \        0x6   0x....'....        BL       HAL_Init
    149
    150            /* USER CODE BEGIN Init */
    151
    152            /* USER CODE END Init */
    153
    154            /* Configure the system clock */
    155            SystemClock_Config();
   \        0xA   0x....'....        BL       SystemClock_Config
    156
    157            /* USER CODE BEGIN SysInit */
    158
    159            /* USER CODE END SysInit */
    160
    161            /* Initialize all configured peripherals */
    162            MX_GPIO_Init();
   \        0xE   0x2114             MOVS     R1,#+20
   \       0x10   0x4668             MOV      R0,SP
   \       0x12   0x....'....        BL       __aeabi_memclr4
   \       0x16   0x2000             MOVS     R0,#+0
   \       0x18   0x9005             STR      R0,[SP, #+20]
   \       0x1A   0xF44F 0x5980      MOV      R9,#+4096
   \       0x1E   0x....'....        LDR.W    R0,??DataTable8_3  ;; 0x40023830
   \       0x22   0x....'....        LDR.W    R4,??DataTable8_1  ;; 0x40021000
   \       0x26   0x6801             LDR      R1,[R0, #+0]
   \       0x28   0x....'....        LDR.W    R5,??DataTable8_4  ;; 0x40020c00
   \       0x2C   0x....'....        LDR.W    R6,??DataTable8_5  ;; 0x40021800
   \       0x30   0xF041 0x0110      ORR      R1,R1,#0x10
   \       0x34   0x6001             STR      R1,[R0, #+0]
   \       0x36   0x2700             MOVS     R7,#+0
   \       0x38   0x6802             LDR      R2,[R0, #+0]
   \       0x3A   0xF002 0x0210      AND      R2,R2,#0x10
   \       0x3E   0x9205             STR      R2,[SP, #+20]
   \       0x40   0x2200             MOVS     R2,#+0
   \       0x42   0x9905             LDR      R1,[SP, #+20]
   \       0x44   0x9205             STR      R2,[SP, #+20]
   \       0x46   0x6803             LDR      R3,[R0, #+0]
   \       0x48   0xF043 0x0380      ORR      R3,R3,#0x80
   \       0x4C   0x6003             STR      R3,[R0, #+0]
   \       0x4E   0x6801             LDR      R1,[R0, #+0]
   \       0x50   0xF001 0x0180      AND      R1,R1,#0x80
```

```
\       0x54   0x9105             STR      R1,[SP, #+20]
\       0x56   0x9905             LDR      R1,[SP, #+20]
\       0x58   0x9205             STR      R2,[SP, #+20]
\       0x5A   0x6803             LDR      R3,[R0, #+0]
\       0x5C   0xF043 0x0301      ORR      R3,R3,#0x1
\       0x60   0x6003             STR      R3,[R0, #+0]
\       0x62   0x6801             LDR      R1,[R0, #+0]
\       0x64   0xF001 0x0101      AND      R1,R1,#0x1
\       0x68   0x9105             STR      R1,[SP, #+20]
\       0x6A   0x9905             LDR      R1,[SP, #+20]
\       0x6C   0x9205             STR      R2,[SP, #+20]
\       0x6E   0x6803             LDR      R3,[R0, #+0]
\       0x70   0xF043 0x0308      ORR      R3,R3,#0x8
\       0x74   0x6003             STR      R3,[R0, #+0]
\       0x76   0x6801             LDR      R1,[R0, #+0]
\       0x78   0xF001 0x0108      AND      R1,R1,#0x8
\       0x7C   0x9105             STR      R1,[SP, #+20]
\       0x7E   0x9905             LDR      R1,[SP, #+20]
\       0x80   0x9205             STR      R2,[SP, #+20]
\       0x82   0x2118             MOVS     R1,#+24
\       0x84   0x6803             LDR      R3,[R0, #+0]
\       0x86   0xF043 0x0340      ORR      R3,R3,#0x40
\       0x8A   0x6003             STR      R3,[R0, #+0]
\       0x8C   0x6800             LDR      R0,[R0, #+0]
\       0x8E   0xF000 0x0040      AND      R0,R0,#0x40
\       0x92   0x9005             STR      R0,[SP, #+20]
\       0x94   0x9805             LDR      R0,[SP, #+20]
\       0x96   0x....'....        BL       ??Subroutine4_0
\                                 ??CrossCallReturnLabel_19: (+1)
\       0x9A   0x2200             MOVS     R2,#+0
\       0x9C   0x210F             MOVS     R1,#+15
\       0x9E   0x....'....        BL       ??Subroutine1_1
\                                 ??CrossCallReturnLabel_3: (+1)
\       0xA2   0x2118             MOVS     R1,#+24
\       0xA4   0x9100             STR      R1,[SP, #+0]
\       0xA6   0x2201             MOVS     R2,#+1
\       0xA8   0x2100             MOVS     R1,#+0
\       0xAA   0x9102             STR      R1,[SP, #+8]
\       0xAC   0x9103             STR      R1,[SP, #+12]
\       0xAE   0x9201             STR      R2,[SP, #+4]
\       0xB0   0x4669             MOV      R1,SP
\       0xB2   0x4620             MOV      R0,R4
\       0xB4   0x....'....        BL       HAL_GPIO_Init
\       0xB8   0x210F             MOVS     R1,#+15
\       0xBA   0x9100             STR      R1,[SP, #+0]
\       0xBC   0x2201             MOVS     R2,#+1
\       0xBE   0x9201             STR      R2,[SP, #+4]
\       0xC0   0x2100             MOVS     R1,#+0
\       0xC2   0x9102             STR      R1,[SP, #+8]
\       0xC4   0x2202             MOVS     R2,#+2
\       0xC6   0x9203             STR      R2,[SP, #+12]
\       0xC8   0x4669             MOV      R1,SP
\       0xCA   0x4628             MOV      R0,R5
\       0xCC   0x....'....        BL       HAL_GPIO_Init
\       0xD0   0x....'....        LDR.W    R1,??DataTable8_6  ;; 0x10110000
\       0xD4   0x9101             STR      R1,[SP, #+4]
\       0xD6   0x2200             MOVS     R2,#+0
\       0xD8   0xF44F 0x7400      MOV      R4,#+512
\       0xDC   0x9202             STR      R2,[SP, #+8]
\       0xDE   0x9400             STR      R4,[SP, #+0]
\       0xE0   0x4669             MOV      R1,SP
```

```
\       0xE2   0x4630            MOV      R0,R6
\       0xE4   0x....'....       BL       HAL_GPIO_Init
\       0xE8   0xF44F 0x40D0     MOV      R0,#+26624
\       0xEC   0x2100            MOVS     R1,#+0
\       0xEE   0x9000            STR      R0,[SP, #+0]
\       0xF0   0x9101            STR      R1,[SP, #+4]
\       0xF2   0x9102            STR      R1,[SP, #+8]
\       0xF4   0x4630            MOV      R0,R6
\       0xF6   0x4669            MOV      R1,SP
\       0xF8   0x....'....       BL       HAL_GPIO_Init
\       0xFC   0x....'....       LDR.W    R1,??DataTable8_7  ;; 0x10310000
\      0x100   0x9101            STR      R1,[SP, #+4]
\      0x102   0x2200            MOVS     R2,#+0
\      0x104   0x9202            STR      R2,[SP, #+8]
\      0x106   0xF8CD 0x9000     STR      R9,[SP, #+0]
\      0x10A   0x4669            MOV      R1,SP
\      0x10C   0x4630            MOV      R0,R6
\      0x10E   0x....'....       BL       HAL_GPIO_Init
\      0x112   0x2200            MOVS     R2,#+0
\      0x114   0x2100            MOVS     R1,#+0
\      0x116   0x2017            MOVS     R0,#+23
\      0x118   0x....'....       BL       HAL_NVIC_SetPriority
\      0x11C   0x2017            MOVS     R0,#+23
\      0x11E   0x....'....       BL       HAL_NVIC_EnableIRQ
\      0x122   0x2200            MOVS     R2,#+0
\      0x124   0x2100            MOVS     R1,#+0
\      0x126   0x2028            MOVS     R0,#+40
\      0x128   0x....'....       BL       HAL_NVIC_SetPriority
\      0x12C   0x2028            MOVS     R0,#+40
\      0x12E   0x....'....       BL       HAL_NVIC_EnableIRQ
 163           MX_TIM5_Init();
\      0x132   0xA806            ADD      R0,SP,#+24
\      0x134   0x2100            MOVS     R1,#+0
\      0x136   0x2200            MOVS     R2,#+0
\      0x138   0x2300            MOVS     R3,#+0
\      0x13A   0xE880 0x008E     STM      R0,{R1-R3,R7}
\      0x13E   0xA802            ADD      R0,SP,#+8
\      0x140   0xE880 0x008E     STM      R0,{R1-R3,R7}
\      0x144   0xE9CD 0x1200     STRD     R1,R2,[SP, #+0]
\      0x148   0xF04F 0x30FF     MOV      R0,#-1
\      0x14C   0x....'....       LDR.W    R7,??DataTable8_8
\      0x150   0x....'....       LDR.W    R1,??DataTable8_9  ;; 0x40000c00
\      0x154   0x65F8            STR      R0,[R7, #+92]
\      0x156   0x6539            STR      R1,[R7, #+80]
\      0x158   0x2080            MOVS     R0,#+128
\      0x15A   0x66B8            STR      R0,[R7, #+104]
\      0x15C   0x657A            STR      R2,[R7, #+84]
\      0x15E   0x65BA            STR      R2,[R7, #+88]
\      0x160   0x663A            STR      R2,[R7, #+96]
\      0x162   0xF107 0x0050     ADD      R0,R7,#+80
\      0x166   0x....'....       BL       HAL_TIM_Base_Init
\      0x16A   0xB108            CBZ.N    R0,??main_0
\      0x16C   0x....'....       BL       Error_Handler
\                       ??main_0: (+1)
\      0x170   0xF8CD 0x9018     STR      R9,[SP, #+24]
\      0x174   0xA906            ADD      R1,SP,#+24
\      0x176   0xF107 0x0050     ADD      R0,R7,#+80
\      0x17A   0x....'....       BL       HAL_TIM_ConfigClockSource
\      0x17E   0xB108            CBZ.N    R0,??main_1
\      0x180   0x....'....       BL       Error_Handler
\                       ??main_1: (+1)
```

```
\      0x184   0xF107 0x0050      ADD      R0,R7,#+80
\      0x188   0x....'....        BL       HAL_TIM_IC_Init
\      0x18C   0xB108             CBZ.N    R0,??main_2
\      0x18E   0x....'....        BL       Error_Handler
\                      ??main_2: (+1)
\      0x192   0x2100             MOVS     R1,#+0
\      0x194   0x9100             STR      R1,[SP, #+0]
\      0x196   0x9101             STR      R1,[SP, #+4]
\      0x198   0xF107 0x0050      ADD      R0,R7,#+80
\      0x19C   0x4669             MOV      R1,SP
\      0x19E   0x....'....        BL       HAL_TIMEx_MasterConfigSynchronization
\      0x1A2   0xB108             CBZ.N    R0,??main_3
\      0x1A4   0x....'....        BL       Error_Handler
\                      ??main_3: (+1)
\      0x1A8   0x2201             MOVS     R2,#+1
\      0x1AA   0x9203             STR      R2,[SP, #+12]
\      0x1AC   0x2100             MOVS     R1,#+0
\      0x1AE   0x9102             STR      R1,[SP, #+8]
\      0x1B0   0x9104             STR      R1,[SP, #+16]
\      0x1B2   0x2200             MOVS     R2,#+0
\      0x1B4   0x9205             STR      R2,[SP, #+20]
\      0x1B6   0xA902             ADD      R1,SP,#+8
\      0x1B8   0xF107 0x0050      ADD      R0,R7,#+80
\      0x1BC   0x....'....        BL       HAL_TIM_IC_ConfigChannel
\      0x1C0   0xB108             CBZ.N    R0,??main_4
\      0x1C2   0x....'....        BL       Error_Handler
 164            MX_TIM4_Init();
\                      ??main_4: (+1)
\      0x1C6   0xA809             ADD      R0,SP,#+36
\      0x1C8   0x2100             MOVS     R1,#+0
\      0x1CA   0x2200             MOVS     R2,#+0
\      0x1CC   0x2300             MOVS     R3,#+0
\      0x1CE   0x468C             MOV      R12,R1
\      0x1D0   0xE880 0x100E      STM      R0,{R1-R3,R12}
\      0x1D4   0xE9CD 0x1200      STRD     R1,R2,[SP, #+0]
\      0x1D8   0xA802             ADD      R0,SP,#+8
\      0x1DA   0x211C             MOVS     R1,#+28
\      0x1DC   0x....'....        BL       __aeabi_memclr4
\      0x1E0   0x2000             MOVS     R0,#+0
\      0x1E2   0x60F8             STR      R0,[R7, #+12]
\      0x1E4   0x2100             MOVS     R1,#+0
\      0x1E6   0xF640 0x30B7      MOVW     R0,#+2999
\      0x1EA   0x6178             STR      R0,[R7, #+20]
\      0x1EC   0x6139             STR      R1,[R7, #+16]
\      0x1EE   0x61B9             STR      R1,[R7, #+24]
\      0x1F0   0x6239             STR      R1,[R7, #+32]
\      0x1F2   0xF107 0x0008      ADD      R0,R7,#+8
\      0x1F6   0x....'....        LDR.W    R8,??DataTable8_10  ;; 0x40000800
\      0x1FA   0xF8C7 0x8008      STR      R8,[R7, #+8]
\      0x1FE   0x....'....        BL       HAL_TIM_Base_Init
\      0x202   0xB108             CBZ.N    R0,??main_5
\      0x204   0x....'....        BL       Error_Handler
\                      ??main_5: (+1)
\      0x208   0xF8CD 0x9024      STR      R9,[SP, #+36]
\      0x20C   0xA909             ADD      R1,SP,#+36
\      0x20E   0xF107 0x0008      ADD      R0,R7,#+8
\      0x212   0x....'....        BL       HAL_TIM_ConfigClockSource
\      0x216   0xB108             CBZ.N    R0,??main_6
\      0x218   0x....'....        BL       Error_Handler
\                      ??main_6: (+1)
\      0x21C   0xF107 0x0008      ADD      R0,R7,#+8
```

```
\      0x220   0x....'....        BL       HAL_TIM_PWM_Init
\      0x224   0xB108             CBZ.N    R0,??main_7
\      0x226   0x....'....        BL       Error_Handler
\                     ??main_7: (+1)
\      0x22A   0x2100             MOVS     R1,#+0
\      0x22C   0x9100             STR      R1,[SP, #+0]
\      0x22E   0x9101             STR      R1,[SP, #+4]
\      0x230   0xF107 0x0008      ADD      R0,R7,#+8
\      0x234   0x4669             MOV      R1,SP
\      0x236   0x....'....        BL       HAL_TIMEx_MasterConfigSynchronization
\      0x23A   0xB108             CBZ.N    R0,??main_8
\      0x23C   0x....'....        BL       Error_Handler
\                     ??main_8: (+1)
\      0x240   0x2160             MOVS     R1,#+96
\      0x242   0x9102             STR      R1,[SP, #+8]
\      0x244   0x2200             MOVS     R2,#+0
\      0x246   0x9203             STR      R2,[SP, #+12]
\      0x248   0x9204             STR      R2,[SP, #+16]
\      0x24A   0x9206             STR      R2,[SP, #+24]
\      0x24C   0x....'....        BL       ?Subroutine9
\                     ??CrossCallReturnLabel_39: (+1)
\      0x250   0xB108             CBZ.N    R0,??main_9
\      0x252   0x....'....        BL       Error_Handler
\                     ??main_9: (+1)
\      0x256   0x2204             MOVS     R2,#+4
\      0x258   0x....'....        BL       ?Subroutine9
\                     ??CrossCallReturnLabel_38: (+1)
\      0x25C   0xB108             CBZ.N    R0,??main_10
\      0x25E   0x....'....        BL       Error_Handler
\                     ??main_10: (+1)
\      0x262   0xF107 0x0008      ADD      R0,R7,#+8
\      0x266   0x....'....        BL       HAL_TIM_MspPostInit
 165            MX_SPI4_Init();
\      0x26A   0xF44F 0x7282      MOV      R2,#+260
\      0x26E   0x....             LDR.N    R0,??DataTable8_2
\      0x270   0x....             LDR.N    R1,??DataTable8_11  ;; 0x40013400
\      0x272   0x6042             STR      R2,[R0, #+4]
\      0x274   0x2300             MOVS     R3,#+0
\      0x276   0x2238             MOVS     R2,#+56
\      0x278   0x61C2             STR      R2,[R0, #+28]
\      0x27A   0x6001             STR      R1,[R0, #+0]
\      0x27C   0x220A             MOVS     R2,#+10
\      0x27E   0x6083             STR      R3,[R0, #+8]
\      0x280   0x60C3             STR      R3,[R0, #+12]
\      0x282   0x6103             STR      R3,[R0, #+16]
\      0x284   0x6143             STR      R3,[R0, #+20]
\      0x286   0x6203             STR      R3,[R0, #+32]
\      0x288   0x6243             STR      R3,[R0, #+36]
\      0x28A   0x6283             STR      R3,[R0, #+40]
\      0x28C   0x62C2             STR      R2,[R0, #+44]
\      0x28E   0x6184             STR      R4,[R0, #+24]
\      0x290   0x....'....        BL       HAL_SPI_Init
\      0x294   0xB108             CBZ.N    R0,??main_11
\      0x296   0x....'....        BL       Error_Handler
 166            /* USER CODE BEGIN 2 */
 167            // Initialize the LCD:
 168            startup();
\                     ??main_11: (+1)
\      0x29A   0x....'....        BL       startup
\      0x29E   0xF107 0x0998      ADD      R9,R7,#+152
\      0x2A2   0xF240 0x5ADC      MOVW     R10,#+1500
```

```
\      0x2A6   0xF242 0x7B10      MOVW     R11,#+10000
\      0x2AA   0xE012             B.N      ??main_12
 169           /* USER CODE END 2 */
 170
 171           /* Infinite loop */
 172           /* USER CODE BEGIN WHILE */
 173           while (1)
 174           {
 175
 176            if (E_STOP)
 177            {
 178
 179               if (START == 0 && state == 0 && E_STOP) // START is active low
 180               {
 181                   HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);
 182                   TIM4->ARR = 20000-1;
 183                   highduty = 10000;
 184                   lowduty = 10000;
 185                   HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);
 186
 187                   for(int i = 0; i < 6; i++) // Blink LED 6 times
 188                   {
 189                     LED_ON;
 190                     HAL_Delay(500); // on for .5 seconds
 191                     LED_OFF;
 192                     HAL_Delay(500); // off for .5 seconds
 193                   }
 194                   state = 1;
 195                   HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1);
 196               }
 197
 198
 199               if (START == 0 && state == 1 && E_STOP)
 200               {
 201                   IR_LED_ON;
 202                   TIM4->ARR = 3000-1;
 203                   highduty = 1500;
 204                   lowduty = 1500;
 205                   HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_2);
 206                   state = 2;
 207               }
 208
 209
 210               if ((state == 2) && E_STOP)
 211               {
 212                  // calculate the frequency:
 213                  if (time1 > time2)
 214                   {
 215                     diff = (time1-time2);
 216                     f = 90000000/diff;
 217                   }
 218                   else if (time2 > time1)
 219                   {
 220                     diff = (time2-time1);
 221                     f = 90000000/diff;
 222                   }
 223
 224                   // update the duty cycle according to the captured frequency:
 225                   dutycycle = (-f/20 + 325);
 226                   highduty  = (uint16_t)dutycycle*3000/100 + 24;
 227            //     lowduty   = (100 - (uint16_t)dutycycle)*3000/100 - 24;
```

```
228
229                     // if reversing, display "Moving Backward" on the LCD:
230                     if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
231                     {
232                     HAL_Delay(100);
233                     Sendcmd(0x90);
234                     writeString("Moving Backward");
235                 //      Sendcmd(0x88);
236                 //      distance = (int)0.5*count + 0x30;
237                 //      distance = (count % 25 );
238                 //      sprintf(distanceString, "%x", distance);
239                 //      writeString(distanceString);
240                 //      remainder = distance;
241                 //      firstDigit = remainder%10;
242                 //      remainder = (remainder - firstDigit)/10;
243                 //      secondDigit = remainder;
244                 //      distance = 'y';
245                 //      sendData(secondDigit + 0x30);
246                 //      sendData(firstDigit  + 0x30);
247                     }
248
249                     // otherwise display "Moving Forward":
250                     else
251                     {
252                     HAL_Delay(100);
253                     Sendcmd(0x90);
254                     writeString("Moving Forward");
255                 //      Sendcmd(0x88);
256                 //      distance = (count % 25 );
257                 //      sprintf(distanceString, "%x", distance);
258                 //      writeString(distanceString);
259                     }
260
261                     // check if STOP is active. If so, the conveyer belt halts
262                     // and a buzzer sounds. A message is also displayed:
263                     if (STOP == 0) // STOP is active low
264                     {
265                       HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
266                       HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture
267
268                       TIM4->ARR = 25000-1;                        // update auto reload
register
269                       highduty = 12500;                           // set 50% duty cycle
270                 //      lowduty = 12500;
271                       HAL_TIM_PWM_Start_IT(&htim4, TIM_CHANNEL_1);// start buzzer
272
273                       Sendcmd(0x01);                              // clear screen;
274                       HAL_Delay(100);                             // brief delay
275                       Sendcmd(0x90);                              // send command to
write on line 2
276                       writeString("Halted");                      // display "Halted"
message
277
278                       for(int i = 0; i < 10; i++)                 // blink LED 10 times
279                       {
280                         LED_ON;
281                         HAL_Delay(250);                           // on for .5 seconds
282                         LED_OFF;
283                         HAL_Delay(250);                           // off for .5 seconds
284                       }
285                       IR_LED_OFF;                                 // turn off IR LED
```

```
 286                     HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
 287                     state = 0;                                  // reset state to 0
 288                     Sendcmd(0x01);                              // clear screen;
 289                   }
 290                }
 291              }
 292
 293              else
 294              {
 295                   HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_1); // stop buzzer
\                     ??main_13: (+1)
\      0x2AC   0x....'....        BL       ?Subroutine6
 296                   HAL_TIM_PWM_Stop_IT(&htim4, TIM_CHANNEL_2); // stop motor
\                     ??CrossCallReturnLabel_28: (+1)
\      0x2B0   0x....'....        BL       ?Subroutine10
 297                   HAL_TIM_IC_Stop_IT(&htim5, TIM_CHANNEL_1);  // stop input capture
\                     ??CrossCallReturnLabel_41: (+1)
\      0x2B4   0x....'....        BL       ?Subroutine8
 298                   LED_OFF;                                    // turn off LED
\                     ??CrossCallReturnLabel_37: (+1)
\      0x2B8   0x....'....        BL       ?Subroutine1
 299                   IR_LED_OFF;                                 // turn off IR LED
\                     ??CrossCallReturnLabel_9: (+1)
\      0x2BC   0x2200             MOVS     R2,#+0
\      0x2BE   0x2102             MOVS     R1,#+2
\      0x2C0   0x....'....        BL       ??Subroutine1_1
 300                   HAL_Delay(100);                             // brief delay
\                     ??CrossCallReturnLabel_2: (+1)
\      0x2C4   0x....'....        BL       ?Subroutine2
 301                   Sendcmd(0x90);                              // write on line 2
\                     ??CrossCallReturnLabel_13: (+1)
\      0x2C8   0x....'....        BL       ?Subroutine5
 302                   writeString("Emergency stop ");             // display message
\                     ??CrossCallReturnLabel_25: (+1)
\      0x2CC   0x....             ADR.N    R0,?_3
\      0x2CE   0x....'....        BL       writeString
 303              }
\                     ??main_12: (+1)
\      0x2D2   0x....'....        BL       ?Subroutine7
\                     ??CrossCallReturnLabel_35: (+1)
\      0x2D6   0x2800             CMP      R0,#+0
\      0x2D8   0xD0E8             BEQ.N    ??main_13
\      0x2DA   0xF44F 0x5100      MOV      R1,#+8192
\      0x2DE   0x....'....        BL       ??Subroutine7_0
\                     ??CrossCallReturnLabel_31: (+1)
\      0x2E2   0x2800             CMP      R0,#+0
\      0x2E4   0xBF04             ITT      EQ
\      0x2E6   0x7838             LDRBEQ   R0,[R7, #+0]
\      0x2E8   0x2800             CMPEQ    R0,#+0
\      0x2EA   0xD128             BNE.N    ??CrossCallReturnLabel_27
\      0x2EC   0x....'....        BL       ?Subroutine7
\                     ??CrossCallReturnLabel_34: (+1)
\      0x2F0   0xB328             CBZ.N    R0,??CrossCallReturnLabel_27
\      0x2F2   0x2100             MOVS     R1,#+0
\      0x2F4   0xF107 0x0050      ADD      R0,R7,#+80
\      0x2F8   0x....'....        BL       HAL_TIM_IC_Start_IT
\      0x2FC   0xF644 0x601F      MOVW     R0,#+19999
\      0x300   0xF8C8 0x002C      STR      R0,[R8, #+44]
\      0x304   0xF8A7 0xB004      STRH     R11,[R7, #+4]
\      0x308   0xF8A7 0xB006      STRH     R11,[R7, #+6]
\      0x30C   0x2100             MOVS     R1,#+0
```

```
\      0x30E   0xF107 0x0008      ADD      R0,R7,#+8
\      0x312   0x....'....        BL       HAL_TIM_PWM_Start_IT
\      0x316   0x2406             MOVS     R4,#+6
\                     ??main_14: (+1)
\      0x318   0x2201             MOVS     R2,#+1
\      0x31A   0x....'....        BL       ??Subroutine1_0
\                     ??CrossCallReturnLabel_6: (+1)
\      0x31E   0xF44F 0x70FA      MOV      R0,#+500
\      0x322   0x....'....        BL       HAL_Delay
\      0x326   0x....'....        BL       ?Subroutine1
\                     ??CrossCallReturnLabel_8: (+1)
\      0x32A   0xF44F 0x70FA      MOV      R0,#+500
\      0x32E   0x....'....        BL       HAL_Delay
\      0x332   0x1E64             SUBS     R4,R4,#+1
\      0x334   0xD1F0             BNE.N    ??main_14
\      0x336   0x2001             MOVS     R0,#+1
\      0x338   0x7038             STRB     R0,[R7, #+0]
\      0x33A   0x....'....        BL       ?Subroutine6
\                     ??CrossCallReturnLabel_27: (+1)
\      0x33E   0xF44F 0x5100      MOV      R1,#+8192
\      0x342   0x....'....        BL       ??Subroutine7_0
\                     ??CrossCallReturnLabel_30: (+1)
\      0x346   0x2800             CMP      R0,#+0
\      0x348   0xBF04             ITT      EQ
\      0x34A   0x7838             LDRBEQ   R0,[R7, #+0]
\      0x34C   0x2801             CMPEQ    R0,#+1
\      0x34E   0xD116             BNE.N    ??main_15
\      0x350   0x....'....        BL       ?Subroutine7
\                     ??CrossCallReturnLabel_33: (+1)
\      0x354   0xB198             CBZ.N    R0,??main_15
\      0x356   0x2201             MOVS     R2,#+1
\      0x358   0x2102             MOVS     R1,#+2
\      0x35A   0x....'....        BL       ??Subroutine1_1
\                     ??CrossCallReturnLabel_1: (+1)
\      0x35E   0xF640 0x30B7      MOVW     R0,#+2999
\      0x362   0xF8C8 0x002C      STR      R0,[R8, #+44]
\      0x366   0xF8A7 0xA004      STRH     R10,[R7, #+4]
\      0x36A   0xF8A7 0xA006      STRH     R10,[R7, #+6]
\      0x36E   0x2104             MOVS     R1,#+4
\      0x370   0xF107 0x0008      ADD      R0,R7,#+8
\      0x374   0x....'....        BL       HAL_TIM_PWM_Start_IT
\      0x378   0x2002             MOVS     R0,#+2
\      0x37A   0x7038             STRB     R0,[R7, #+0]
\      0x37C   0xE002             B.N      ??main_16
\                     ??main_15: (+1)
\      0x37E   0x7839             LDRB     R1,[R7, #+0]
\      0x380   0x2902             CMP      R1,#+2
\      0x382   0xD1A6             BNE.N    ??main_12
\                     ??main_16: (+1)
\      0x384   0x....'....        BL       ?Subroutine7
\                     ??CrossCallReturnLabel_32: (+1)
\      0x388   0x2800             CMP      R0,#+0
\      0x38A   0xD0A2             BEQ.N    ??main_12
\      0x38C   0xF8D9 0x1004      LDR      R1,[R9, #+4]
\      0x390   0xF8D9 0x2000      LDR      R2,[R9, #+0]
\      0x394   0xF8D9 0x0008      LDR      R0,[R9, #+8]
\      0x398   0x4291             CMP      R1,R2
\      0x39A   0xD206             BCS.N    ??main_17
\      0x39C   0x1A50             SUBS     R0,R2,R1
\      0x39E   0x....             LDR.N    R1,??DataTable8_12  ;; 0x55d4a80
\      0x3A0   0xFBB1 0xF1F0      UDIV     R1,R1,R0
```

```
\      0x3A4   0xF8C9 0x100C      STR      R1,[R9, #+12]
\      0x3A8   0xE007             B.N      ??main_18
\                     ??main_17: (+1)
\      0x3AA   0x428A             CMP      R2,R1
\      0x3AC   0xD205             BCS.N    ??main_18
\      0x3AE   0x1A88             SUBS     R0,R1,R2
\      0x3B0   0x....             LDR.N    R2,??DataTable8_12  ;; 0x55d4a80
\      0x3B2   0xFBB2 0xF2F0      UDIV     R2,R2,R0
\      0x3B6   0xF8C9 0x200C      STR      R2,[R9, #+12]
\                     ??main_18: (+1)
\      0x3BA   0x2214             MOVS     R2,#+20
\      0x3BC   0xF8D9 0x100C      LDR      R1,[R9, #+12]
\      0x3C0   0xF8C9 0x0008      STR      R0,[R9, #+8]
\      0x3C4   0x4249             RSBS     R1,R1,#+0
\      0x3C6   0xFBB1 0xF1F2      UDIV     R1,R1,R2
\      0x3CA   0xF201 0x1145      ADDW     R1,R1,#+325
\      0x3CE   0xB28B             UXTH     R3,R1
\      0x3D0   0xF640 0x30B8      MOVW     R0,#+3000
\      0x3D4   0x4343             MULS     R3,R0,R3
\      0x3D6   0x2464             MOVS     R4,#+100
\      0x3D8   0xFB93 0xF4F4      SDIV     R4,R3,R4
\      0x3DC   0x8079             STRH     R1,[R7, #+2]
\      0x3DE   0x3418             ADDS     R4,R4,#+24
\      0x3E0   0x80BC             STRH     R4,[R7, #+4]
\      0x3E2   0x2108             MOVS     R1,#+8
\      0x3E4   0x4628             MOV      R0,R5
\      0x3E6   0x....'....        BL       HAL_GPIO_ReadPin
\      0x3EA   0xB128             CBZ.N    R0,??main_19
\      0x3EC   0x....'....        BL       ?Subroutine2
\                     ??CrossCallReturnLabel_12: (+1)
\      0x3F0   0x....'....        BL       ?Subroutine5
\                     ??CrossCallReturnLabel_24: (+1)
\      0x3F4   0x....             ADR.N    R0,?_0
\      0x3F6   0xE004             B.N      ??main_20
\                     ??main_19: (+1)
\      0x3F8   0x....'....        BL       ?Subroutine2
\                     ??CrossCallReturnLabel_11: (+1)
\      0x3FC   0x....'....        BL       ?Subroutine5
\                     ??CrossCallReturnLabel_23: (+1)
\      0x400   0x....             ADR.N    R0,?_1
\                     ??main_20: (+1)
\      0x402   0x....'....        BL       writeString
\      0x406   0xF44F 0x4180      MOV      R1,#+16384
\      0x40A   0x....'....        BL       ??Subroutine7_0
\                     ??CrossCallReturnLabel_29: (+1)
\      0x40E   0x2800             CMP      R0,#+0
\      0x410   0xD132             BNE.N    ??main_21
\      0x412   0x....'....        BL       ?Subroutine10
\                     ??CrossCallReturnLabel_40: (+1)
\      0x416   0x....'....        BL       ?Subroutine8
\                     ??CrossCallReturnLabel_36: (+1)
\      0x41A   0xF246 0x10A7      MOVW     R0,#+24999
\      0x41E   0xF8C8 0x002C      STR      R0,[R8, #+44]
\      0x422   0xF243 0x01D4      MOVW     R1,#+12500
\      0x426   0x80B9             STRH     R1,[R7, #+4]
\      0x428   0xF107 0x0008      ADD      R0,R7,#+8
\      0x42C   0x2100             MOVS     R1,#+0
\      0x42E   0x....'....        BL       HAL_TIM_PWM_Start_IT
\      0x432   0x2001             MOVS     R0,#+1
\      0x434   0x....'....        BL       Sendcmd
\      0x438   0x....'....        BL       ?Subroutine2
```

```
\                       ??CrossCallReturnLabel_10: (+1)
\      0x43C   0x....'....        BL       ?Subroutine5
\                       ??CrossCallReturnLabel_22: (+1)
\      0x440   0x....             ADR.N    R0,?_2
\      0x442   0x....'....        BL       writeString
\      0x446   0x240A             MOVS     R4,#+10
\                       ??main_22: (+1)
\      0x448   0x2201             MOVS     R2,#+1
\      0x44A   0x....'....        BL       ??Subroutine1_0
\                       ??CrossCallReturnLabel_5: (+1)
\      0x44E   0x20FA             MOVS     R0,#+250
\      0x450   0x....'....        BL       HAL_Delay
\      0x454   0x....'....        BL       ?Subroutine1
\                       ??CrossCallReturnLabel_7: (+1)
\      0x458   0x20FA             MOVS     R0,#+250
\      0x45A   0x....'....        BL       HAL_Delay
\      0x45E   0x1E64             SUBS     R4,R4,#+1
\      0x460   0xD1F2             BNE.N    ??main_22
\      0x462   0x2200             MOVS     R2,#+0
\      0x464   0x2102             MOVS     R1,#+2
\      0x466   0x....'....        BL       ??Subroutine1_1
\                       ??CrossCallReturnLabel_0: (+1)
\      0x46A   0x....'....        BL       ?Subroutine6
\                       ??CrossCallReturnLabel_26: (+1)
\      0x46E   0x2000             MOVS     R0,#+0
\      0x470   0x7038             STRB     R0,[R7, #+0]
\      0x472   0x2001             MOVS     R0,#+1
\      0x474   0x....'....        BL       Sendcmd
\                       ??main_21: (+1)
\      0x478   0xE72B             B.N      ??main_12
 304
 305             /* USER CODE END WHILE */
 306             /* USER CODE BEGIN 3 */
 307           }
 308           /* USER CODE END 3 */
 309         }

\                                  In section .text, align 2, keep-with-next
\                       ?Subroutine10: (+1)
\        0x0   0x2104             MOVS     R1,#+4
\        0x2   0xF107 0x0008      ADD      R0,R7,#+8
\        0x6   0x....'....        B.W      HAL_TIM_PWM_Stop_IT

\                                  In section .text, align 2, keep-with-next
\                       ?Subroutine8: (+1)
\        0x0   0x2100             MOVS     R1,#+0
\        0x2   0xF107 0x0050      ADD      R0,R7,#+80
\        0x6   0x....'....        B.W      HAL_TIM_IC_Stop_IT

\                                  In section .text, align 2, keep-with-next
\                       ?Subroutine7: (+1)
\        0x0   0xF44F 0x6100      MOV      R1,#+2048
\                       ??Subroutine7_0: (+1)
\        0x4   0x4630             MOV      R0,R6
\        0x6   0x....'....        B.W      HAL_GPIO_ReadPin

\                                  In section .text, align 2, keep-with-next
\                       ?Subroutine6: (+1)
\        0x0   0x2100             MOVS     R1,#+0
\        0x2   0xF107 0x0008      ADD      R0,R7,#+8
\        0x6   0x....'....        B.W      HAL_TIM_PWM_Stop_IT
```

```
\                                In section .text, align 2, keep-with-next
\                        ?Subroutine5: (+1)
\        0x0   0x2090            MOVS     R0,#+144
\        0x2   0x....            B.N      Sendcmd
 310
 311          /**
 312            * @brief System Clock Configuration
 313            * @retval None
 314            */

\                                In section .text, align 2, keep-with-next
 315          void SystemClock_Config(void)
 316          {
\                        SystemClock_Config: (+1)
\        0x0   0xB580            PUSH     {R7,LR}
\        0x2   0xB092            SUB      SP,SP,#+72
\        0x4   0x2130            MOVS     R1,#+48
\        0x6   0xA806            ADD      R0,SP,#+24
\        0x8   0x....'....       BL       __aeabi_memclr4
\        0xC   0x2114            MOVS     R1,#+20
\        0xE   0xA801            ADD      R0,SP,#+4
\       0x10   0x....'....       BL       __aeabi_memclr4
 317            RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 318            RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 319
 320            /** Configure the main internal regulator output voltage
 321            */
 322            __HAL_RCC_PWR_CLK_ENABLE();
\       0x14   0x2000            MOVS     R0,#+0
\       0x16   0x9000            STR      R0,[SP, #+0]
 323            __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
 324            /** Initializes the RCC Oscillators according to the specified parameters
 325            * in the RCC_OscInitTypeDef structure.
 326            */
 327            RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
 328            RCC_OscInitStruct.HSEState = RCC_HSE_ON;
\       0x18   0xF44F 0x3380     MOV      R3,#+65536
\       0x1C   0x....            LDR.N    R0,??DataTable8_13  ;; 0x40023840
\       0x1E   0x6801            LDR      R1,[R0, #+0]
\       0x20   0xF041 0x5180     ORR      R1,R1,#0x10000000
\       0x24   0x6001            STR      R1,[R0, #+0]
\       0x26   0x2100            MOVS     R1,#+0
\       0x28   0x6800            LDR      R0,[R0, #+0]
\       0x2A   0xF000 0x5080     AND      R0,R0,#0x10000000
\       0x2E   0x9000            STR      R0,[SP, #+0]
\       0x30   0x9800            LDR      R0,[SP, #+0]
\       0x32   0x....            LDR.N    R0,??DataTable8_14  ;; 0x40007000
\       0x34   0x9100            STR      R1,[SP, #+0]
\       0x36   0x6802            LDR      R2,[R0, #+0]
\       0x38   0xF442 0x4240     ORR      R2,R2,#0xC000
\       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]
 329            RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 330            RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
 331            RCC_OscInitStruct.PLL.PLLM = 4;
 332            RCC_OscInitStruct.PLL.PLLN = 180;
 333            RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
```

```
 334            RCC_OscInitStruct.PLL.PLLQ = 4;
 335            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   0x22B4             MOVS     R2,#+180
\       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....'....        BL       HAL_RCC_OscConfig
\       0x6E   0xB108             CBZ.N    R0,??SystemClock_Config_0
 336            {
 337              Error_Handler();
\       0x70   0x....'....        BL       Error_Handler
 338            }
 339            /** Activate the Over-Drive mode
 340            */
 341            if (HAL_PWREx_EnableOverDrive() != HAL_OK)
\                     ??SystemClock_Config_0: (+1)
\       0x74   0x....'....        BL       HAL_PWREx_EnableOverDrive
\       0x78   0xB108             CBZ.N    R0,??SystemClock_Config_1
 342            {
 343              Error_Handler();
\       0x7A   0xB672             CPSID    I
\                     ??SystemClock_Config_2: (+1)
\       0x7C   0xE7FE             B.N      ??SystemClock_Config_2
 344            }
 345            /** Initializes the CPU, AHB and APB buses clocks
 346            */
 347            RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
 348                                          |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 349            RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
\                     ??SystemClock_Config_1: (+1)
\       0x7E   0x2102             MOVS     R1,#+2
\       0x80   0x9102             STR      R1,[SP, #+8]
 350            RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
\       0x82   0x2200             MOVS     R2,#+0
\       0x84   0x9203             STR      R2,[SP, #+12]
\       0x86   0x200F             MOVS     R0,#+15
 351            RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
\       0x88   0xF44F 0x51A0      MOV      R1,#+5120
\       0x8C   0x9001             STR      R0,[SP, #+4]
\       0x8E   0x9104             STR      R1,[SP, #+16]
 352            RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
\       0x90   0xF44F 0x5280      MOV      R2,#+4096
\       0x94   0x9205             STR      R2,[SP, #+20]
 353
 354            if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
\       0x96   0x2105             MOVS     R1,#+5
\       0x98   0xA801             ADD      R0,SP,#+4
\       0x9A   0x....'....        BL       HAL_RCC_ClockConfig
\       0x9E   0xB108             CBZ.N    R0,??SystemClock_Config_3
```

```
 355           {
 356             Error_Handler();
\       0xA0   0xB672             CPSID    I
\                      ??SystemClock_Config_4: (+1)
\       0xA2   0xE7FE             B.N      ??SystemClock_Config_4
 357           }
 358         }
\                      ??SystemClock_Config_3: (+1)
\       0xA4   0xB013             ADD      SP,SP,#+76
\       0xA6   0xBD00             POP      {PC}             ;; return
 359
 360         /**
 361           * @brief SPI4 Initialization Function
 362           * @param None
 363           * @retval None
 364           */
 365         static void MX_SPI4_Init(void)
 366         {
 367
 368           /* USER CODE BEGIN SPI4_Init 0 */
 369
 370           /* USER CODE END SPI4_Init 0 */
 371
 372           /* USER CODE BEGIN SPI4_Init 1 */
 373
 374           /* USER CODE END SPI4_Init 1 */
 375           /* SPI4 parameter configuration*/
 376           hspi4.Instance = SPI4;
 377           hspi4.Init.Mode = SPI_MODE_MASTER;
 378           hspi4.Init.Direction = SPI_DIRECTION_2LINES;
 379           hspi4.Init.DataSize = SPI_DATASIZE_8BIT;
 380           hspi4.Init.CLKPolarity = SPI_POLARITY_LOW;
 381           hspi4.Init.CLKPhase = SPI_PHASE_1EDGE;
 382           hspi4.Init.NSS = SPI_NSS_SOFT;
 383           hspi4.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;
 384           hspi4.Init.FirstBit = SPI_FIRSTBIT_MSB;
 385           hspi4.Init.TIMode = SPI_TIMODE_DISABLE;
 386           hspi4.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
 387           hspi4.Init.CRCPolynomial = 10;
 388           if (HAL_SPI_Init(&hspi4) != HAL_OK)
 389           {
 390             Error_Handler();
 391           }
 392           /* USER CODE BEGIN SPI4_Init 2 */
 393
 394           /* USER CODE END SPI4_Init 2 */
 395
 396         }
 397
 398         /**
 399           * @brief TIM4 Initialization Function
 400           * @param None
 401           * @retval None
 402           */
 403         static void MX_TIM4_Init(void)
 404         {
 405
 406           /* USER CODE BEGIN TIM4_Init 0 */
 407
 408           /* USER CODE END TIM4_Init 0 */
 409
```

```
410           TIM_ClockConfigTypeDef sClockSourceConfig = {0};
411           TIM_MasterConfigTypeDef sMasterConfig = {0};
412           TIM_OC_InitTypeDef sConfigOC = {0};
413
414           /* USER CODE BEGIN TIM4_Init 1 */
415
416           /* USER CODE END TIM4_Init 1 */
417           htim4.Instance = TIM4;
418           htim4.Init.Prescaler = 0;
419           htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
420           htim4.Init.Period = 3000-1;
421           htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
422           htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
423           if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
424           {
425             Error_Handler();
426           }
427           sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
428           if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
429           {
430             Error_Handler();
431           }
432           if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
433           {
434             Error_Handler();
435           }
436           sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
437           sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
438           if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) !=
HAL_OK)
439           {
440             Error_Handler();
441           }
442           sConfigOC.OCMode = TIM_OCMODE_PWM1;
443           sConfigOC.Pulse = 0;
444           sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
445           sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
446           if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
447           {
448             Error_Handler();
449           }
450           if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
451           {
452             Error_Handler();
453           }
454           /* USER CODE BEGIN TIM4_Init 2 */
455
456           /* USER CODE END TIM4_Init 2 */
457           HAL_TIM_MspPostInit(&htim4);
458
459         }
460
461         /**
462           * @brief TIM5 Initialization Function
463           * @param None
464           * @retval None
465           */
466         static void MX_TIM5_Init(void)
467         {
468
469           /* USER CODE BEGIN TIM5_Init 0 */
```

```
470
471           /* USER CODE END TIM5_Init 0 */
472
473           TIM_ClockConfigTypeDef sClockSourceConfig = {0};
474           TIM_MasterConfigTypeDef sMasterConfig = {0};
475           TIM_IC_InitTypeDef sConfigIC = {0};
476
477           /* USER CODE BEGIN TIM5_Init 1 */
478
479           /* USER CODE END TIM5_Init 1 */
480           htim5.Instance = TIM5;
481           htim5.Init.Prescaler = 0;
482           htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
483           htim5.Init.Period = 4294967295;
484           htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
485           htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
486           if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
487           {
488             Error_Handler();
489           }
490           sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
491           if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
492           {
493             Error_Handler();
494           }
495           if (HAL_TIM_IC_Init(&htim5) != HAL_OK)
496           {
497             Error_Handler();
498           }
499           sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
500           sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
501           if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) !=
HAL_OK)
502           {
503             Error_Handler();
504           }
505           sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
506           sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
507           sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
508           sConfigIC.ICFilter = 0;
509           if (HAL_TIM_IC_ConfigChannel(&htim5, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
510           {
511             Error_Handler();
512           }
513           /* USER CODE BEGIN TIM5_Init 2 */
514
515           /* USER CODE END TIM5_Init 2 */
516
517         }
518
519         /**
520           * @brief GPIO Initialization Function
521           * @param None
522           * @retval None
523           */
524         static void MX_GPIO_Init(void)
525         {
526           GPIO_InitTypeDef GPIO_InitStruct = {0};
527
528           /* GPIO Ports Clock Enable */
529           __HAL_RCC_GPIOE_CLK_ENABLE();
```

```
    530           __HAL_RCC_GPIOH_CLK_ENABLE();
    531           __HAL_RCC_GPIOA_CLK_ENABLE();
    532           __HAL_RCC_GPIOD_CLK_ENABLE();
    533           __HAL_RCC_GPIOG_CLK_ENABLE();
    534
    535           /*Configure GPIO pin Output Level */
    536           HAL_GPIO_WritePin(GPIOE, RS_Pin|Reset_Pin, GPIO_PIN_RESET);
    537
    538           /*Configure GPIO pin Output Level */
    539           HAL_GPIO_WritePin(GPIOD, LED_Pin|IR_LED_Pin|FORWARD_Pin|REVERSE_Pin,
GPIO_PIN_RESET);
    540
    541           /*Configure GPIO pins : RS_Pin Reset_Pin */
    542           GPIO_InitStruct.Pin = RS_Pin|Reset_Pin;
    543           GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    544           GPIO_InitStruct.Pull = GPIO_NOPULL;
    545           GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    546           HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
    547
    548           /*Configure GPIO pins : LED_Pin IR_LED_Pin FORWARD_Pin REVERSE_Pin */
    549           GPIO_InitStruct.Pin = LED_Pin|IR_LED_Pin|FORWARD_Pin|REVERSE_Pin;
    550           GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    551           GPIO_InitStruct.Pull = GPIO_NOPULL;
    552           GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    553           HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
    554
    555           /*Configure GPIO pin : DISTANCE_Pin */
    556           GPIO_InitStruct.Pin = DISTANCE_Pin;
    557           GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
    558           GPIO_InitStruct.Pull = GPIO_NOPULL;
    559           HAL_GPIO_Init(DISTANCE_GPIO_Port, &GPIO_InitStruct);
    560
    561           /*Configure GPIO pins : E_STOP_Pin START_Pin STOP_Pin */
    562           GPIO_InitStruct.Pin = E_STOP_Pin|START_Pin|STOP_Pin;
    563           GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    564           GPIO_InitStruct.Pull = GPIO_NOPULL;
    565           HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
    566
    567           /*Configure GPIO pin : DIRECTION_Pin */
    568           GPIO_InitStruct.Pin = DIRECTION_Pin;
    569           GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
    570           GPIO_InitStruct.Pull = GPIO_NOPULL;
    571           HAL_GPIO_Init(DIRECTION_GPIO_Port, &GPIO_InitStruct);
    572
    573           /* EXTI interrupt init*/
    574           HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
    575           HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
    576
    577           HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
    578           HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
    579
    580         }
    581
    582         /* USER CODE BEGIN 4 */
    583
    584         /* USER CODE END 4 */
    585
    586         /**
    587           * @brief  This function is executed in case of error occurrence.
    588           * @retval None
    589           */
```

```
   \                                In section .text, align 2, keep-with-next
    590          void Error_Handler(void)
    591          {
    592            /* USER CODE BEGIN Error_Handler_Debug */
    593            /* User can add his own implementation to report the HAL error return state
*/
    594            __disable_irq();
   \                     Error_Handler: (+1)
   \        0x0   0xB672             CPSID    I
    595            while (1)
   \                     ??Error_Handler_0: (+1)
   \        0x2   0xE7FE             B.N      ??Error_Handler_0
    596            {
    597            }
    598            /* USER CODE END Error_Handler_Debug */
    599          }

   \                                In section .text, align 2, keep-with-next
   \                     ?Subroutine9: (+1)
   \        0x0   0xA902             ADD      R1,SP,#+8
   \        0x2   0xF107 0x0008      ADD      R0,R7,#+8
   \        0x6   0x....'....        B.W      HAL_TIM_PWM_ConfigChannel

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8:
   \        0x0   0x....'....        DC32     array

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_1:
   \        0x0   0x4002'1000        DC32     0x40021000

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_2:
   \        0x0   0x....'....        DC32     hspi4

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_3:
   \        0x0   0x4002'3830        DC32     0x40023830

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_4:
   \        0x0   0x4002'0C00        DC32     0x40020c00

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_5:
   \        0x0   0x4002'1800        DC32     0x40021800

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_6:
   \        0x0   0x1011'0000        DC32     0x10110000

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_7:
   \        0x0   0x1031'0000        DC32     0x10310000

   \                                In section .text, align 4, keep-with-next
   \                     ??DataTable8_8:
   \        0x0   0x....'....        DC32     state

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

```
\                       ??DataTable8_9:
\        0x0   0x4000'0C00        DC32     0x40000c00

\                                In section .text, align 4, keep-with-next
\                       ??DataTable8_10:
\        0x0   0x4000'0800        DC32     0x40000800

\                                In section .text, align 4, keep-with-next
\                       ??DataTable8_11:
\        0x0   0x4001'3400        DC32     0x40013400

\                                In section .text, align 4, keep-with-next
\                       ??DataTable8_12:
\        0x0   0x055D'4A80        DC32     0x55d4a80

\                                In section .text, align 4, keep-with-next
\                       ??DataTable8_13:
\        0x0   0x4002'3840        DC32     0x40023840

\                                In section .text, align 4, keep-with-next
\                       ??DataTable8_14:
\        0x0   0x4000'7000        DC32     0x40007000

\                                In section .text, align 4, keep-with-next
\                       ?_0:
\        0x0   0x4D 0x6F          DC8 "Moving Backward"

\              0x76 0x69

\              0x6E 0x67

\              0x20 0x42

\              0x61 0x63

\              0x6B 0x77

\              0x61 0x72

\              0x64 0x00

\                                In section .text, align 4, keep-with-next
\                       ?_1:
\        0x0   0x4D 0x6F          DC8 "Moving Forward"

\              0x76 0x69

\              0x6E 0x67

\              0x20 0x46

\              0x6F 0x72

\              0x77 0x61

\              0x72 0x64

\              0x00
\        0xF                      DS8 1

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

```
   \                     ?_2:
   \        0x0   0x48 0x61          DC8 "Halted"

   \              0x6C 0x74

   \              0x65 0x64

   \              0x00
   \        0x7                     DS8 1

   \                               In section .text, align 4, keep-with-next
   \                     ?_3:
   \        0x0   0x45 0x6D          DC8 "Emergency stop "

   \              0x65 0x72

   \              0x67 0x65

   \              0x6E 0x63

   \              0x79 0x20

   \              0x73 0x74

   \              0x6F 0x70

   \              0x20 0x00
    600
    601          #ifdef  USE_FULL_ASSERT
    602          /**
    603            * @brief  Reports the name of the source file and the source line number
    604            *         where the assert_param error has occurred.
    605            * @param  file: pointer to the source file name
    606            * @param  line: assert_param error line source number
    607            * @retval None
    608            */
    609          void assert_failed(uint8_t *file, uint32_t line)
    610          {
    611            /* USER CODE BEGIN 6 */
    612            /* User can add his own implementation to report the file name and line
number,
    613               ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
*/
    614            /* USER CODE END 6 */
    615          }
    616          #endif /* USE_FULL_ASSERT */
    617
    618          /************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/

   Maximum stack usage in bytes:

   .cstack Function
   ------- --------
       0   Error_Handler
      16   Sendcmd
         0   -> HAL_GPIO_WritePin
        16   -> HAL_GPIO_WritePin
        16   -> HAL_SPI_Transmit
      80   SystemClock_Config
        80   -> Error_Handler
```

```
      80   -> HAL_PWREx_EnableOverDrive
      80   -> HAL_RCC_ClockConfig
      80   -> HAL_RCC_OscConfig
      80   -> __aeabi_memclr4
    88   main
      88   -> Error_Handler
      88   -> HAL_Delay
      88   -> HAL_GPIO_Init
      88   -> HAL_GPIO_ReadPin
      88   -> HAL_GPIO_WritePin
      88   -> HAL_Init
      88   -> HAL_NVIC_EnableIRQ
      88   -> HAL_NVIC_SetPriority
      88   -> HAL_SPI_Init
      88   -> HAL_TIMEx_MasterConfigSynchronization
      88   -> HAL_TIM_Base_Init
      88   -> HAL_TIM_ConfigClockSource
      88   -> HAL_TIM_IC_ConfigChannel
      88   -> HAL_TIM_IC_Init
      88   -> HAL_TIM_IC_Start_IT
      88   -> HAL_TIM_IC_Stop_IT
      88   -> HAL_TIM_MspPostInit
      88   -> HAL_TIM_PWM_ConfigChannel
      88   -> HAL_TIM_PWM_Init
      88   -> HAL_TIM_PWM_Start_IT
      88   -> HAL_TIM_PWM_Stop_IT
      88   -> Sendcmd
      88   -> SystemClock_Config
      88   -> __aeabi_memclr4
      88   -> startup
      88   -> writeString
    16   sendData
       0   -> HAL_GPIO_WritePin
      16   -> HAL_GPIO_WritePin
      16   -> HAL_SPI_Transmit
     8   startup
       0   -> HAL_Delay
       8   -> HAL_Delay
       8   -> HAL_GPIO_WritePin
       8   -> Sendcmd
    16   writeString
      16   -> sendData
      16   -> strlen


  Section sizes:

  Bytes  Function/Label
  -----  --------------
      4  ??DataTable8
      4  ??DataTable8_1
      4  ??DataTable8_10
      4  ??DataTable8_11
      4  ??DataTable8_12
      4  ??DataTable8_13
      4  ??DataTable8_14
      4  ??DataTable8_2
      4  ??DataTable8_3
      4  ??DataTable8_4
      4  ??DataTable8_5
      4  ??DataTable8_6
```

```
       4  ??DataTable8_7
       4  ??DataTable8_8
       4  ??DataTable8_9
      52  ?Subroutine0
      10  ?Subroutine1
      10  ?Subroutine10
       6  ?Subroutine2
       6  ?Subroutine3
       8  ?Subroutine4
       4  ?Subroutine5
      10  ?Subroutine6
      10  ?Subroutine7
      10  ?Subroutine8
      10  ?Subroutine9
      16  ?_0
      16  ?_1
       8  ?_2
      16  ?_3
       4  Error_Handler
      10  Sendcmd
     168  SystemClock_Config
       4  array
       1  count
       1  distance
       2  distanceString
       1  firstDigit
      88  hspi4
   1'146  main
       1  remainder
       1  secondDigit
       8  sendData
      84  startup
     168  state
           dutycycle
           highduty
           lowduty
           htim4
           htim5
           time1
           time2
           diff
           f
      28  writeString


   267 bytes in section .bss
 1'700 bytes in section .text

 1'700 bytes of CODE memory
   267 bytes of DATA memory

Errors: none
Warnings: none
```

[stm32f4xx_it.lst]

```
###############################################################################
#
# IAR ANSI C/C++ Compiler V8.32.2.178/W32 for ARM         20/Mar/2021  21:47:03
# Copyright 1999-2018 IAR Systems AB.
#
#    Cpu mode     =
#    Endian       =  little
#    Source file  =  C:\Users\ecelab\Desktop\HW4\Core\Src\stm32f4xx_it.c
#    Command line =
#        -f C:\Users\ecelab\AppData\Local\Temp\EW93A.tmp
#        (C:\Users\ecelab\Desktop\HW4\Core\Src\stm32f4xx_it.c -D USE_HAL_DRIVER
#        -D STM32F429xx -lC C:\Users\ecelab\Desktop\HW4\EWARM\HW4\List -o
#        C:\Users\ecelab\Desktop\HW4\EWARM\HW4\Obj --debug --endian=little
#        --cpu=Cortex-M4 -e --fpu=VFPv4_sp --dlib_config "C:\Program Files
#        (x86)\IAR Systems\Embedded Workbench 8.2\arm\inc\c\DLib_Config_Full.h"
#        -I C:\Users\ecelab\Desktop\HW4\EWARM/../Core/Inc\ -I
#        C:\Users\ecelab\Desktop\HW4\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc\
#        -I
#        C:\Users\ecelab\Desktop\HW4\EWARM/../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy\
#        -I
#        C:\Users\ecelab\Desktop\HW4\EWARM/../Drivers/CMSIS/Device/ST/STM32F4xx/Include\
#        -I C:\Users\ecelab\Desktop\HW4\EWARM/../Drivers/CMSIS/Include\ -Ohz)
#    Locale       =  C
#    List file    =
#        C:\Users\ecelab\Desktop\HW4\EWARM\HW4\List\stm32f4xx_it.lst
#    Object file  =  C:\Users\ecelab\Desktop\HW4\EWARM\HW4\Obj\stm32f4xx_it.o
#
###############################################################################

C:\Users\ecelab\Desktop\HW4\Core\Src\stm32f4xx_it.c
      1          /* USER CODE BEGIN Header */
      2          /**
      3
******************************************************************************
      4            * @file    stm32f4xx_it.c
      5            * @brief   Interrupt Service Routines.
      6
******************************************************************************
      7            * @attention
      8            *
      9            * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
     10            * All rights reserved.</center></h2>
     11            *
     12            * This software component is licensed by ST under BSD 3-Clause license,
     13            * the "License"; You may not use this file except in compliance with the
     14            * License. You may obtain a copy of the License at:
     15            *                        opensource.org/licenses/BSD-3-Clause
     16            *
     17
******************************************************************************
     18            */
     19          /* USER CODE END Header */
     20
     21          /* Includes ------------------------------------------------------------------
-*/
     22          #include "main.h"
     23          #include "stm32f4xx_it.h"
     24          /* Private includes ----------------------------------------------------------
-*/
```

```
     25          /* USER CODE BEGIN Includes */
     26          /* USER CODE END Includes */
     27
     28          /* Private typedef -----------------------------------------------------------
-*/
     29          /* USER CODE BEGIN TD */
     30
     31          /* USER CODE END TD */
     32
     33          /* Private define ------------------------------------------------------------
-*/
     34          /* USER CODE BEGIN PD */
     35
     36          /* USER CODE END PD */
     37
     38          /* Private macro -------------------------------------------------------------
-*/
     39          /* USER CODE BEGIN PM */
     40
     41          /* USER CODE END PM */
     42
     43          /* Private variables ---------------------------------------------------------
-*/
     44          /* USER CODE BEGIN PV */

   \                                In section .bss, align 1
     45            uint8_t temp0, temp1, temp2;
   \                       temp0:
   \        0x0                     DS8 1

   \                                In section .bss, align 1
   \                       temp1:
   \        0x0                     DS8 1

   \                                In section .bss, align 1
   \                       temp2:
   \        0x0                     DS8 1
     46          /* USER CODE END PV */
     47
     48          /* Private function prototypes -----------------------------------------------
-*/
     49          /* USER CODE BEGIN PFP */
     50
     51          /* USER CODE END PFP */
     52
     53          /* Private user code ---------------------------------------------------------
-*/
     54          /* USER CODE BEGIN 0 */
     55
     56          /* USER CODE END 0 */
     57
     58          /* External variables --------------------------------------------------------
-*/
     59          extern TIM_HandleTypeDef htim4;
     60          extern TIM_HandleTypeDef htim5;
     61          /* USER CODE BEGIN EV */
     62          extern uint32_t time1, time2;
     63          extern uint16_t highduty, lowduty;
     64          extern uint8_t count;
     65          /* USER CODE END EV */
     66
```

```
     67
/******************************************************************************/
     68          /*           Cortex-M4 Processor Interruption and Exception Handlers
*/
     69
/******************************************************************************/
     70          /**
     71            * @brief This function handles Non maskable interrupt.
     72            */

   \                                In section .text, align 2, keep-with-next
     73          void NMI_Handler(void)
     74          {
     75            /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
     76
     77            /* USER CODE END NonMaskableInt_IRQn 0 */
     78            /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
     79            while (1)
   \                      NMI_Handler: (+1)
   \                      ??NMI_Handler_0: (+1)
   \        0x0   0xE7FE             B.N      ??NMI_Handler_0
     80            {
     81            }
     82            /* USER CODE END NonMaskableInt_IRQn 1 */
     83          }
     84
     85          /**
     86            * @brief This function handles Hard fault interrupt.
     87            */

   \                                In section .text, align 2, keep-with-next
     88          void HardFault_Handler(void)
     89          {
     90            /* USER CODE BEGIN HardFault_IRQn 0 */
     91
     92            /* USER CODE END HardFault_IRQn 0 */
     93            while (1)
   \                      HardFault_Handler: (+1)
   \                      ??HardFault_Handler_0: (+1)
   \        0x0   0xE7FE             B.N      ??HardFault_Handler_0
     94            {
     95              /* USER CODE BEGIN W1_HardFault_IRQn 0 */
     96              /* USER CODE END W1_HardFault_IRQn 0 */
     97            }
     98          }
     99
    100          /**
    101            * @brief This function handles Memory management fault.
    102            */

   \                                In section .text, align 2, keep-with-next
    103          void MemManage_Handler(void)
    104          {
    105            /* USER CODE BEGIN MemoryManagement_IRQn 0 */
    106
    107            /* USER CODE END MemoryManagement_IRQn 0 */
    108            while (1)
   \                      MemManage_Handler: (+1)
   \                      ??MemManage_Handler_0: (+1)
   \        0x0   0xE7FE             B.N      ??MemManage_Handler_0
    109            {
```

```
 110             /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
 111             /* USER CODE END W1_MemoryManagement_IRQn 0 */
 112           }
 113         }
 114
 115         /**
 116           * @brief This function handles Pre-fetch fault, memory access fault.
 117           */

\                                In section .text, align 2, keep-with-next
 118         void BusFault_Handler(void)
 119         {
 120           /* USER CODE BEGIN BusFault_IRQn 0 */
 121
 122           /* USER CODE END BusFault_IRQn 0 */
 123           while (1)
\                   BusFault_Handler: (+1)
\                   ??BusFault_Handler_0: (+1)
\        0x0   0xE7FE             B.N      ??BusFault_Handler_0
 124           {
 125             /* USER CODE BEGIN W1_BusFault_IRQn 0 */
 126             /* USER CODE END W1_BusFault_IRQn 0 */
 127           }
 128         }
 129
 130         /**
 131           * @brief This function handles Undefined instruction or illegal state.
 132           */

\                                In section .text, align 2, keep-with-next
 133         void UsageFault_Handler(void)
 134         {
 135           /* USER CODE BEGIN UsageFault_IRQn 0 */
 136
 137           /* USER CODE END UsageFault_IRQn 0 */
 138           while (1)
\                   UsageFault_Handler: (+1)
\                   ??UsageFault_Handler_0: (+1)
\        0x0   0xE7FE             B.N      ??UsageFault_Handler_0
 139           {
 140             /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
 141             /* USER CODE END W1_UsageFault_IRQn 0 */
 142           }
 143         }
 144
 145         /**
 146           * @brief This function handles System service call via SWI instruction.
 147           */

\                                In section .text, align 2, keep-with-next
 148         void SVC_Handler(void)
 149         {
 150           /* USER CODE BEGIN SVCall_IRQn 0 */
 151
 152           /* USER CODE END SVCall_IRQn 0 */
 153           /* USER CODE BEGIN SVCall_IRQn 1 */
 154
 155           /* USER CODE END SVCall_IRQn 1 */
 156         }
\                   SVC_Handler: (+1)
\        0x0   0x4770             BX       LR               ;; return
```

```
    157
    158          /**
    159            * @brief This function handles Debug monitor.
    160            */

   \                                In section .text, align 2, keep-with-next
    161          void DebugMon_Handler(void)
    162          {
    163            /* USER CODE BEGIN DebugMonitor_IRQn 0 */
    164
    165            /* USER CODE END DebugMonitor_IRQn 0 */
    166            /* USER CODE BEGIN DebugMonitor_IRQn 1 */
    167
    168            /* USER CODE END DebugMonitor_IRQn 1 */
    169          }
   \                        DebugMon_Handler: (+1)
   \        0x0   0x4770             BX       LR               ;; return
    170
    171          /**
    172            * @brief This function handles Pendable request for system service.
    173            */

   \                                In section .text, align 2, keep-with-next
    174          void PendSV_Handler(void)
    175          {
    176            /* USER CODE BEGIN PendSV_IRQn 0 */
    177
    178            /* USER CODE END PendSV_IRQn 0 */
    179            /* USER CODE BEGIN PendSV_IRQn 1 */
    180
    181            /* USER CODE END PendSV_IRQn 1 */
    182          }
   \                        PendSV_Handler: (+1)
   \        0x0   0x4770             BX       LR               ;; return
    183
    184          /**
    185            * @brief This function handles System tick timer.
    186            */

   \                                In section .text, align 2, keep-with-next
    187          void SysTick_Handler(void)
    188          {
    189            /* USER CODE BEGIN SysTick_IRQn 0 */
    190
    191            /* USER CODE END SysTick_IRQn 0 */
    192            HAL_IncTick();
   \                        SysTick_Handler: (+1)
   \        0x0   0x....'....        B.W      HAL_IncTick
    193            /* USER CODE BEGIN SysTick_IRQn 1 */
    194
    195            /* USER CODE END SysTick_IRQn 1 */
    196          }
    197
    198
/******************************************************************************/
    199          /* STM32F4xx Peripheral Interrupt Handlers
*/
    200          /* Add here the Interrupt Handlers for the used peripherals.
*/
    201          /* For the available peripheral interrupt handler names,
*/
```

```
    202          /* please refer to the startup file (startup_stm32f4xx.s).
*/
    203
/*******************************************************************************/
    204
    205          /**
    206            * @brief This function handles EXTI line[9:5] interrupts.
    207            */

   \                                In section .text, align 2, keep-with-next
    208          void EXTI9_5_IRQHandler(void)
    209          {
   \                          EXTI9_5_IRQHandler: (+1)
   \        0x0   0xB580             PUSH     {R7,LR}
    210            /* USER CODE BEGIN EXTI9_5_IRQn 0 */
    211
    212
    213
    214
    215            /* USER CODE END EXTI9_5_IRQn 0 */
    216            HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_9);
   \        0x2   0xF44F 0x7000      MOV      R0,#+512
   \        0x6   0x....'....        BL       HAL_GPIO_EXTI_IRQHandler
    217            /* USER CODE BEGIN EXTI9_5_IRQn 1 */
    218                if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
   \        0xA   0x2108             MOVS     R1,#+8
   \        0xC   0x....             LDR.N    R0,??DataTable4  ;; 0x40020c00
   \        0xE   0x....'....        BL       HAL_GPIO_ReadPin
   \       0x12   0x....             LDR.N    R1,??DataTable4_1
   \       0x14   0x780A             LDRB     R2,[R1, #+0]
   \       0x16   0x2800             CMP      R0,#+0
   \       0x18   0xBF14             ITE      NE
   \       0x1A   0x1E52             SUBNE    R2,R2,#+1
   \       0x1C   0x1C52             ADDEQ    R2,R2,#+1
    219                {
    220                count--;
    221                }
    222                else
    223                {
    224                count++;
   \       0x1E   0x700A             STRB     R2,[R1, #+0]
    225                }
    226            /* USER CODE END EXTI9_5_IRQn 1 */
    227          }
   \       0x20   0xBD01             POP      {R0,PC}          ;; return
    228
    229          /**
    230            * @brief This function handles TIM4 global interrupt.
    231            */

   \                                In section .text, align 2, keep-with-next
    232          void TIM4_IRQHandler(void)
    233          {
   \                          TIM4_IRQHandler: (+1)
   \        0x0   0xB510             PUSH     {R4,LR}
    234            /* USER CODE BEGIN TIM4_IRQn 0 */
    235
    236            /* USER CODE END TIM4_IRQn 0 */
    237            HAL_TIM_IRQHandler(&htim4);
   \        0x2   0x....             LDR.N    R0,??DataTable4_2
   \        0x4   0x....'....        BL       HAL_TIM_IRQHandler
```

```
 238           /* USER CODE BEGIN TIM4_IRQn 1 */
 239         //  if (temp1 == 1)
 240         //  {
 241             if (TIM4->ARR == 19999) {
\        0x8   0x....              LDR.N    R1,??DataTable4_3
\        0xA   0x8808              LDRH     R0,[R1, #+0]
\        0xC   0x....              LDR.N    R1,??DataTable4_4  ;; 0x4000082c
\        0xE   0x680A              LDR      R2,[R1, #+0]
\       0x10   0xF644 0x631F       MOVW     R3,#+19999
\       0x14   0x429A              CMP      R2,R3
\       0x16   0xBF1E              ITTT     NE
\       0x18   0x680C              LDRNE    R4,[R1, #+0]
\       0x1A   0xF246 0x12A7       MOVWNE   R2,#+24999
\       0x1E   0x4294              CMPNE    R4,R2
 242               TIM4->CCR1 = highduty;
 243               TIM4->CCR2 = 0;
 244             }
 245             else if (TIM4->ARR == 24999)
\       0x20   0xBF07              ITTEE    EQ
\       0x22   0x6088              STREQ    R0,[R1, #+8]
\       0x24   0x2000              MOVEQ    R0,#+0
\       0x26   0x2200              MOVNE    R2,#+0
\       0x28   0x608A              STRNE    R2,[R1, #+8]
 246             {
 247               TIM4->CCR1 = highduty;
 248               TIM4->CCR2 = 0;
 249             }
 250             else
 251             {
 252               TIM4->CCR1 = 0;
 253               TIM4->CCR2 = highduty;
\       0x2A   0x60C8              STR      R0,[R1, #+12]
 254             }
 255         //   temp1 = 0;
 256         //  }
 257         //  else
 258         //  {
 259         //    if (TIM4->ARR == 19999) {
 260         //      TIM4->CCR1 = lowduty;
 261         //      TIM4->CCR2 = 0;
 262         //    }
 263         //    else
 264         //    {
 265         //      TIM4->CCR1 = 0;
 266         //      TIM4->CCR2 = lowduty;
 267         //    }
 268         //   temp1 = 1;
 269         //  }
 270           /* USER CODE END TIM4_IRQn 1 */
 271         }
\       0x2C   0xBD10              POP      {R4,PC}          ;; return
 272
 273         /**
 274           * @brief This function handles EXTI line[15:10] interrupts.
 275           */

\                                  In section .text, align 2, keep-with-next
 276         void EXTI15_10_IRQHandler(void)
 277         {
\                          EXTI15_10_IRQHandler: (+1)
\        0x0   0xB510              PUSH     {R4,LR}
```

```
 278            /* USER CODE BEGIN EXTI15_10_IRQn 0 */
 279
 280            /* USER CODE END EXTI15_10_IRQn 0 */
 281            HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_12);
\        0x2   0xF44F 0x5080      MOV      R0,#+4096
\        0x6   0x....'....        BL       HAL_GPIO_EXTI_IRQHandler
 282            /* USER CODE BEGIN EXTI15_10_IRQn 1 */
 283            HAL_GPIO_TogglePin(GPIOD, REVERSE_Pin);
\        0xA   0x....              LDR.N    R4,??DataTable4  ;; 0x40020c00
\        0xC   0x2108              MOVS     R1,#+8
\        0xE   0x4620              MOV      R0,R4
\       0x10   0x....'....        BL       HAL_GPIO_TogglePin
 284            if (HAL_GPIO_ReadPin(GPIOD, REVERSE_Pin))
\       0x14   0x2108              MOVS     R1,#+8
\       0x16   0x4620              MOV      R0,R4
\       0x18   0x....'....        BL       HAL_GPIO_ReadPin
\       0x1C   0x2800              CMP      R0,#+0
 285            {
 286              HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_RESET);
 287            }
 288            else
 289            {
 290              HAL_GPIO_WritePin(GPIOD, FORWARD_Pin, GPIO_PIN_SET);
\       0x1E   0x4620              MOV      R0,R4
\       0x20   0xE8BD 0x4010      POP      {R4,LR}
\       0x24   0xBF14              ITE      NE
\       0x26   0x2200              MOVNE    R2,#+0
\       0x28   0x2201              MOVEQ    R2,#+1
\       0x2A   0x2104              MOVS     R1,#+4
\       0x2C   0x....'....        B.W      HAL_GPIO_WritePin
 291            }
 292            /* USER CODE END EXTI15_10_IRQn 1 */
 293          }
 294
 295          /**
 296            * @brief This function handles TIM5 global interrupt.
 297            */

\                                  In section .text, align 2, keep-with-next
 298          void TIM5_IRQHandler(void)
 299          {
\                          TIM5_IRQHandler: (+1)
\        0x0   0xB580              PUSH     {R7,LR}
 300            /* USER CODE BEGIN TIM5_IRQn 0 */
 301
 302            /* USER CODE END TIM5_IRQn 0 */
 303            HAL_TIM_IRQHandler(&htim5);
\        0x2   0x....              LDR.N    R0,??DataTable4_5
\        0x4   0x....'....        BL       HAL_TIM_IRQHandler
 304            /* USER CODE BEGIN TIM5_IRQn 1 */
 305
 306              // if the first time hasn't been captured yet:
 307              if(temp2 == 0) {
\        0x8   0x....              LDR.N    R1,??DataTable4_6
\        0xA   0x....              LDR.N    R2,??DataTable4_7  ;; 0x40000c34
\        0xC   0x7808              LDRB     R0,[R1, #+0]
\        0xE   0xB920              CBNZ.N   R0,??TIM5_IRQHandler_0
 308                time1 = TIM5->CCR1;
\       0x10   0x6810              LDR      R0,[R2, #+0]
\       0x12   0x....              LDR.N    R2,??DataTable4_8
\       0x14   0x6010              STR      R0,[R2, #+0]
```

```
 309               temp2 = 1;
\       0x16   0x2301             MOVS     R3,#+1
\       0x18   0xE005             B.N      ??TIM5_IRQHandler_1
 310             }
 311             // otherwise, if the first value has been captured already:
 312             else if (temp2 == 1) {
\                      ??TIM5_IRQHandler_0: (+1)
\       0x1A   0x2801             CMP      R0,#+1
\       0x1C   0xD104             BNE.N    ??TIM5_IRQHandler_2
 313               time2 = TIM5->CCR1;
\       0x1E   0x6810             LDR      R0,[R2, #+0]
\       0x20   0x....             LDR.N    R2,??DataTable4_9
\       0x22   0x6010             STR      R0,[R2, #+0]
 314               temp2 = 0;
\       0x24   0x2300             MOVS     R3,#+0
\                      ??TIM5_IRQHandler_1: (+1)
\       0x26   0x700B             STRB     R3,[R1, #+0]
 315             }
 316
 317           /* USER CODE END TIM5_IRQn 1 */
 318         }
\                      ??TIM5_IRQHandler_2: (+1)
\       0x28   0xBD01             POP      {R0,PC}          ;; return

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4:
\        0x0   0x4002'0C00        DC32     0x40020c00

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_1:
\        0x0   0x....'....        DC32     count

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_2:
\        0x0   0x....'....        DC32     htim4

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

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_4:
\        0x0   0x4000'082C        DC32     0x4000082c

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_5:
\        0x0   0x....'....        DC32     htim5

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_6:
\        0x0   0x....'....        DC32     temp2

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_7:
\        0x0   0x4000'0C34        DC32     0x40000c34

\                                 In section .text, align 4, keep-with-next
\                      ??DataTable4_8:
\        0x0   0x....'....        DC32     time1

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

```
   \                     ??DataTable4_9:
   \        0x0   0x....'....        DC32     time2
    319
    320          /* USER CODE BEGIN 1 */
    321
    322          /* USER CODE END 1 */
    323          /************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/

   Maximum stack usage in bytes:

   .cstack Function
   ------- --------
       0   BusFault_Handler
       0   DebugMon_Handler
       8   EXTI15_10_IRQHandler
         8   -> HAL_GPIO_EXTI_IRQHandler
         8   -> HAL_GPIO_ReadPin
         8   -> HAL_GPIO_TogglePin
         0   -> HAL_GPIO_WritePin
       8   EXTI9_5_IRQHandler
         8   -> HAL_GPIO_EXTI_IRQHandler
         8   -> HAL_GPIO_ReadPin
       0   HardFault_Handler
       0   MemManage_Handler
       0   NMI_Handler
       0   PendSV_Handler
       0   SVC_Handler
       0   SysTick_Handler
         0   -> HAL_IncTick
       8   TIM4_IRQHandler
         8   -> HAL_TIM_IRQHandler
       8   TIM5_IRQHandler
         8   -> HAL_TIM_IRQHandler
       0   UsageFault_Handler


   Section sizes:

   Bytes  Function/Label
   -----  --------------
       4  ??DataTable4
       4  ??DataTable4_1
       4  ??DataTable4_2
       4  ??DataTable4_3
       4  ??DataTable4_4
       4  ??DataTable4_5
       4  ??DataTable4_6
       4  ??DataTable4_7
       4  ??DataTable4_8
       4  ??DataTable4_9
       2  BusFault_Handler
       2  DebugMon_Handler
      48  EXTI15_10_IRQHandler
      34  EXTI9_5_IRQHandler
       2  HardFault_Handler
       2  MemManage_Handler
       2  NMI_Handler
       2  PendSV_Handler
       2  SVC_Handler
       4  SysTick_Handler
```

```
      46  TIM4_IRQHandler
      42  TIM5_IRQHandler
       2  UsageFault_Handler
       1  temp0
       1  temp1
       1  temp2


   3 bytes in section .bss
 230 bytes in section .text

 230 bytes of CODE memory
   3 bytes of DATA memory

Errors: none
Warnings: none
```