EEE3096S 2023 PRAC 4
DGMROB001 – KNXTHO003

Github link:

https://github.com/rothdu/EEE3096S
https://github.com/rothdu/EEE3096S/tree/main/Prac4

Code:

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

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include "stm32f0xx.h"
#include <lcd_stm32f0.c>
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
// TODO: Add values for below variables
#define NS 500 // Number of samples in LUT
#define TIM2CLK 8000000 // STM Clock frequency
#define F_SIGNAL 50 // Frequency of output analog signal
/* USER CODE END PD */
```

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
DMA_HandleTypeDef hdma_tim2_ch1;

/* USER CODE BEGIN PV */
// TODO: Add code for global variables, including LUTs

uint32_t Sin_LUT[NS] = {
512, 518, 524, 531, 537, 544, 550, 556, 563, 569, 576, 582, 588, 595, 601, 607,
614, 620, 626, 632, 639, 645, 651, 657, 663, 670, 676, 682, 688, 694, 700, 706,
712, 718, 723, 729, 735, 741, 747, 752, 758, 764, 769, 775, 780, 786, 791, 796,
802, 807, 812, 817, 822, 828, 833, 838, 842, 847, 852, 857, 862, 866, 871, 875,
880, 884, 889, 893, 897, 901, 906, 910, 914, 918, 922, 925, 929, 933, 936, 940,
943, 947, 950, 953, 957, 960, 963, 966, 969, 972, 974, 977, 980, 982, 985, 987,
989, 992, 994, 996, 998,1000,1002,1004,1005,1007,1008,1010,1011,1013,1014,1015,
1016,1017,1018,1019,1020,1020,1021,1022,1022,1022,1023,1023,1023,1023,1023,1023,
1023,1022,1022,1022,1021,1020,1020,1019,1018,1017,1016,1015,1014,1013,1011,1010,
1008,1007,1005,1004,1002,1000, 998, 996, 994, 992, 989, 987, 985, 982, 980, 977,
974, 972, 969, 966, 963, 960, 957, 953, 950, 947, 943, 940, 936, 933, 929, 925,
922, 918, 914, 910, 906, 901, 897, 893, 889, 884, 880, 875, 871, 866, 862, 857,
852, 847, 842, 838, 833, 828, 822, 817, 812, 807, 802, 796, 791, 786, 780, 775,
769, 764, 758, 752, 747, 741, 735, 729, 723, 718, 712, 706, 700, 694, 688, 682,
676, 670, 663, 657, 651, 645, 639, 632, 626, 620, 614, 607, 601, 595, 588, 582,
576, 569, 563, 556, 550, 544, 537, 531, 524, 518, 512, 505, 499, 492, 486, 479,
473, 467, 460, 454, 447, 441, 435, 428, 422, 416, 409, 403, 397, 391, 384, 378,
372, 366, 360, 353, 347, 341, 335, 329, 323, 317, 311, 305, 300, 294, 288, 282,
276, 271, 265, 259, 254, 248, 243, 237, 232, 227, 221, 216, 211, 206, 201, 195,
190, 185, 181, 176, 171, 166, 161, 157, 152, 148, 143, 139, 134, 130, 126, 122,
117, 113, 109, 105, 101, 98, 94, 90, 87, 83, 80, 76, 73, 70, 66, 63,
60, 57, 54, 51, 49, 46, 43, 41, 38, 36, 34, 31, 29, 27, 25, 23,
21, 19, 18, 16, 15, 13, 12, 10, 9, 8, 7, 6, 5, 4, 3, 3,
2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 3,
3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 18, 19, 21, 23,
25, 27, 29, 31, 34, 36, 38, 41, 43, 46, 49, 51, 54, 57, 60, 63,
66, 70, 73, 76, 80, 83, 87, 90, 94, 98, 101, 105, 109, 113, 117, 122,
126, 130, 134, 139, 143, 148, 152, 157, 161, 166, 171, 176, 181, 185, 190, 195,
201, 206, 211, 216, 221, 227, 232, 237, 243, 248, 254, 259, 265, 271, 276, 282,
288, 294, 300, 305, 311, 317, 323, 329, 335, 341, 347, 353, 360, 366, 372, 378,
384, 391, 397, 403, 409, 416, 422, 428, 435, 441, 447, 454, 460, 467, 473, 479,
486, 492, 499, 505
};

uint32_t saw_LUT[NS] = {
0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 21, 23, 25, 27, 29, 31,
33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 62, 64,
66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96,
98, 100, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129,
131, 133, 135, 137, 139, 141, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162,
164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 185, 187, 189, 191, 193, 195,
197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 226, 228,
230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 258, 260,
262, 264, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293,
295, 297, 299, 301, 303, 305, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326,
328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 349, 351, 353, 355, 357, 359,
361, 363, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 390, 392,
394, 396, 398, 400, 402, 404, 406, 408, 410, 412, 414, 416, 418, 420, 422, 424,
426, 428, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455, 457,
459, 461, 463, 465, 467, 469, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490,
492, 494, 496, 498, 500, 502, 504, 506, 508, 510, 513, 515, 517, 519, 521, 523,
525, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 549, 551, 554, 556,
558, 560, 562, 564, 566, 568, 570, 572, 574, 576, 578, 580, 582, 584, 586, 588,
590, 592, 595, 597, 599, 601, 603, 605, 607, 609, 611, 613, 615, 617, 619, 621,
623, 625, 627, 629, 631, 633, 636, 638, 640, 642, 644, 646, 648, 650, 652, 654,
656, 658, 660, 662, 664, 666, 668, 670, 672, 674, 677, 679, 681, 683, 685, 687,
689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 711, 713, 715, 718, 720,
722, 724, 726, 728, 730, 732, 734, 736, 738, 740, 742, 744, 746, 748, 750, 752,
754, 756, 759, 761, 763, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785,
787, 789, 791, 793, 795, 797, 800, 802, 804, 806, 808, 810, 812, 814, 816, 818,
820, 822, 824, 826, 828, 830, 832, 834, 836, 838, 841, 843, 845, 847, 849, 851,
853, 855, 857, 859, 861, 863, 865, 867, 869, 871, 873, 875, 877, 879, 882, 884,
886, 888, 890, 892, 894, 896, 898, 900, 902, 904, 906, 908, 910, 912, 914, 916,
918, 920, 923, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 949,
951, 953, 955, 957, 959, 961, 964, 966, 968, 970, 972, 974, 976, 978, 980, 982,
984, 986, 988, 990, 992, 994, 996, 998,1000,1002,1005,1007,1009,1011,1013,1015,
1017,1019,1021,1023
};

uint32_t triangle_LUT[NS] = {
0, 4, 8, 12, 16, 20, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61,
65, 70, 74, 78, 82, 86, 90, 94, 98, 102, 106, 110, 115, 119, 123, 127,
131, 135, 139, 143, 147, 151, 155, 160, 164, 168, 172, 176, 180, 184, 188, 192,
196, 201, 205, 209, 213, 217, 221, 225, 229, 233, 237, 241, 246, 250, 254, 258,
262, 266, 270, 274, 278, 282, 286, 291, 295, 299, 303, 307, 311, 315, 319, 323,
327, 331, 336, 340, 344, 348, 352, 356, 360, 364, 368, 372, 376, 381, 385, 389,
393, 397, 401, 405, 409, 413, 417, 421, 426, 430, 434, 438, 442, 446, 450, 454,
458, 462, 466, 471, 475, 479, 483, 487, 491, 495, 499, 503, 507, 512, 516, 520,
524, 528, 532, 536, 540, 544, 548, 552, 557, 561, 565, 569, 573, 577, 581, 585,
589, 593, 597, 602, 606, 610, 614, 618, 622, 626, 630, 634, 638, 642, 647, 651,
655, 659, 663, 667, 671, 675, 679, 683, 687, 692, 696, 700, 704, 708, 712, 716,
720, 724, 728, 732, 737, 741, 745, 749, 753, 757, 761, 765, 769, 773, 777, 782,
786, 790, 794, 798, 802, 806, 810, 814, 818, 822, 827, 831, 835, 839, 843, 847,
851, 855, 859, 863, 868, 872, 876, 880, 884, 888, 892, 896, 900, 904, 908, 913,
917, 921, 925, 929, 933, 937, 941, 945, 949, 953, 958, 962, 966, 970, 974, 978,
982, 986, 990, 994, 998,1003,1007,1011,1015,1019,1023,1019,1015,1011,1007,1003,
998, 994, 990, 986, 982, 978, 974, 970, 966, 962, 958, 953, 949, 945, 941, 937,
933, 929, 925, 921, 917, 913, 908, 904, 900, 896, 892, 888, 884, 880, 876, 872,
868, 863, 859, 855, 851, 847, 843, 839, 835, 831, 827, 822, 818, 814, 810, 806,
802, 798, 794, 790, 786, 782, 777, 773, 769, 765, 761, 757, 753, 749, 745, 741,
737, 732, 728, 724, 720, 716, 712, 708, 704, 700, 696, 692, 687, 683, 679, 675,
671, 667, 663, 659, 655, 651, 647, 642, 638, 634, 630, 626, 622, 618, 614, 610,
606, 602, 597, 593, 589, 585, 581, 577, 573, 569, 565, 561, 557, 552, 548, 544,
540, 536, 532, 528, 524, 520, 516, 512, 507, 503, 499, 495, 491, 487, 483, 479,
475, 471, 466, 462, 458, 454, 450, 446, 442, 438, 434, 430, 426, 421, 417, 413,
409, 405, 401, 397, 393, 389, 385, 381, 376, 372, 368, 364, 360, 356, 352, 348,
344, 340, 336, 331, 327, 323, 319, 315, 311, 307, 303, 299, 295, 291, 286, 282,
278, 274, 270, 266, 262, 258, 254, 250, 246, 241, 237, 233, 229, 225, 221, 217,
213, 209, 205, 201, 196, 192, 188, 184, 180, 176, 172, 168, 164, 160, 155, 151,
147, 143, 139, 135, 131, 127, 123, 119, 115, 110, 106, 102, 98, 94, 90, 86,
82, 78, 74, 70, 65, 61, 57, 53, 49, 45, 41, 37, 33, 29, 25, 20,
16, 12, 8, 4
};

uint8_t current_LUT_num = 0; // referred to for changing LUTs

uint32_t button0_prev_tick = 0; // initialise previous tick to 0
// for button0 debounce
uint8_t debounce_delay = 50;

// TODO: Equation to calculate TIM2_Ticks
uint32_t TIM2_Ticks = TIM2CLK/(F_SIGNAL * NS); // How often to write new LUT value
uint32_t DestAddress = (uint32_t) &(TIM3->CCR3); // Write LUT TO TIM3->CCR3 to modify
PWM duty cycle

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);

/* USER CODE BEGIN PFP */
void EXTI0_1_IRQHandler(void);
/* 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 */
init_LCD();
/* 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_DMA_Init();
MX_TIM2_Init();
MX_TIM3_Init();

/* USER CODE BEGIN 2 */
// TODO: Start TIM3 in PWM mode on channel 3
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);


// TODO: Start TIM2 in Output Compare (OC) mode on channel 1.
HAL_TIM_OC_Start(&htim2, TIM_CHANNEL_1);


// TODO: Start DMA in IT mode on TIM2->CH1; Source is LUT and Dest is TIM3->CCR3; start
with Sine LUT
HAL_DMA_Start_IT(&hdma_tim2_ch1, (uint32_t)Sin_LUT, DestAddress, NS);

// TODO: Write current waveform to LCD ("Sine")
delay(3000);
lcd_command(CLEAR);
lcd_putstring("Sine");

// TODO: Enable DMA (start transfer from LUT to CCR)
__HAL_TIM_ENABLE_DMA(&htim2, TIM_DMA_CC1);

/* USER CODE END 2 */

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

```
{
/* USER CODE END WHILE */

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

/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
LL_FLASH_SetLatency(LL_FLASH_LATENCY_0);
while(LL_FLASH_GetLatency() != LL_FLASH_LATENCY_0)
{
}
LL_RCC_HSI_Enable();

/* Wait till HSI is ready */
while(LL_RCC_HSI_IsReady() != 1)
{

}
LL_RCC_HSI_SetCalibTrimming(16);
LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_1);
LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_HSI);

/* Wait till System clock is ready */
while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_HSI)
{

}
LL_SetSystemCoreClock(8000000);

/* Update the time base */
if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
{
Error_Handler();
}
}

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

```
/* USER CODE BEGIN TIM2_Init 0 */

/* USER CODE END TIM2_Init 0 */

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

/* USER CODE BEGIN TIM2_Init 1 */

/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 0;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = TIM2_Ticks - 1;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
Error_Handler();
}
if (HAL_TIM_OC_Init(&htim2) != HAL_OK)
{
Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_TIMING;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_OC_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */

/* USER CODE END TIM2_Init 2 */

}
```

```c
/**
* @brief TIM3 Initialization Function
* @param None
* @retval None
*/
static void MX_TIM3_Init(void)
{

/* USER CODE BEGIN TIM3_Init 0 */

/* USER CODE END TIM3_Init 0 */

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

/* USER CODE BEGIN TIM3_Init 1 */

/* USER CODE END TIM3_Init 1 */
htim3.Instance = TIM3;
htim3.Init.Prescaler = 0;
htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
htim3.Init.Period = 1023;
htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
{
Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
{
Error_Handler();
}
if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
{
Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &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(&htim3, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN TIM3_Init 2 */

/* USER CODE END TIM3_Init 2 */
HAL_TIM_MspPostInit(&htim3);

}

/**
* Enable DMA controller clock
*/
static void MX_DMA_Init(void)
{

/* DMA controller clock enable */
__HAL_RCC_DMA1_CLK_ENABLE();

/* DMA interrupt init */
/* DMA1_Channel4_5_IRQn interrupt configuration */
HAL_NVIC_SetPriority(DMA1_Channel4_5_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(DMA1_Channel4_5_IRQn);

}

/**
* @brief GPIO Initialization Function
* @param None
* @retval None
*/
static void MX_GPIO_Init(void)
{
LL_EXTI_InitTypeDef EXTI_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOF);
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA);
LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOB);

/**/
LL_SYSCFG_SetEXTISource(LL_SYSCFG_EXTI_PORTA, LL_SYSCFG_EXTI_LINE0);

/**/
LL_GPIO_SetPinPull(Button0_GPIO_Port, Button0_Pin, LL_GPIO_PULL_UP);

/**/
LL_GPIO_SetPinMode(Button0_GPIO_Port, Button0_Pin, LL_GPIO_MODE_INPUT);

/**/
EXTI_InitStruct.Line_0_31 = LL_EXTI_LINE_0;
EXTI_InitStruct.LineCommand = ENABLE;
EXTI_InitStruct.Mode = LL_EXTI_MODE_IT;
EXTI_InitStruct.Trigger = LL_EXTI_TRIGGER_RISING;
LL_EXTI_Init(&EXTI_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
HAL_NVIC_SetPriority(EXTI0_1_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI0_1_IRQn);
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
void EXTI0_1_IRQHandler(void)
{
// TODO: Debounce using HAL_GetTick()
uint32_t current_tick = HAL_GetTick();
if (current_tick - button0_prev_tick < debounce_delay) {
return; // premature return if debounce delay has not been passed
}
button0_prev_tick = current_tick;

// TODO: Disable DMA transfer and abort IT, then start DMA in IT mode with new LUT and
re-enable transfer
// HINT: Consider using C's "switch" function to handle LUT changes

__HAL_TIM_DISABLE_DMA(&htim2, TIM_DMA_CC1);
HAL_DMA_Abort_IT(&hdma_tim2_ch1);

uint32_t * new_LUT;
char lcd_string[11];

// switch statement checks prev LUT and assigns new LUT
switch (++current_LUT_num){ // pre-increment current num
case 3:
current_LUT_num = 0; // loop back from 3 to 0
case 0:
new_LUT = Sin_LUT;
strcpy(lcd_string, "Sine");
break;
case 1:
new_LUT = saw_LUT;
strcpy(lcd_string, "Sawtooth");
break;
case 2:
new_LUT = triangle_LUT;
strcpy(lcd_string, "Triangular");
break;
}

// start DMA
HAL_DMA_Start_IT(&hdma_tim2_ch1, (uint32_t)new_LUT, DestAddress, NS);
```

```c
// write current waveform to LCD
delay(3000);
lcd_command(CLEAR);
lcd_putstring(lcd_string);

// re-enable DMA
__HAL_TIM_ENABLE_DMA(&htim2, TIM_DMA_CC1);

HAL_GPIO_EXTI_IRQHandler(Button0_Pin); // Clear interrupt flags
}
/* 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 */
```