TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

EEE3096S PRACTICAL 4

GITHUB LINK
https://github.com/CHHANK001/3096S-Pracs-CHHANK001-TMBTIN004.git

```
/* 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  150*2      // Number of samples in LUT
#define TIM2CLK  8000000  // STM Clock frequency
#define F_SIGNAL 200  // Frequency of output analog signal
char sine[] ="Sine";
char saw[]="Saw";
char Triangle[]="Triangle";
uint32_t last = 0;
uint32_t first=0;
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```
uint32_t LUT = 0;
uint32_t delay_t=500;

/* USER CODE END PD */

/* 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] = {511, 522, 532, 543, 554, 564, 575, 586, 596, 607, 617, 628,
638, 648, 659, 669, 679, 689, 699, 709, 719, 729, 738, 748, 757, 766, 776, 785, 794,
803, 811, 820, 828, 837, 845, 853, 861, 869, 876, 884, 891, 898, 905, 911, 918, 924,
931, 937, 942, 948, 954, 959, 964, 969, 973, 978, 982, 986, 990, 994, 997, 1000,
1003, 1006, 1008, 1011, 1013, 1015, 1017, 1018, 1019, 1020, 1021, 1022, 1022, 1022,
1022, 1022, 1021, 1020, 1019, 1018, 1017, 1015, 1013, 1011, 1008, 1006, 1003, 1000,
997, 994, 990, 986, 982, 978, 973, 969, 964, 959, 954, 948, 942, 937, 931, 924, 918,
911, 905, 898, 891, 884, 876, 869, 861, 853, 845, 837, 828, 820,811, 803, 794, 785,
776, 767, 757, 748, 738, 729, 719, 709, 699, 689, 679, 669, 659, 648, 638, 628, 617,
607, 596, 586, 575, 564, 554, 543, 532, 522, 511, 500, 490, 479, 468, 458, 447, 436,
426, 415, 405, 394, 384, 374, 363, 353, 343, 333, 323, 313, 303, 293, 284, 274, 265,
256, 246, 237, 228, 219, 211, 202, 194, 185, 177, 169, 161, 153, 146, 138, 131, 124,
117, 111, 104, 98, 91, 85, 80, 74, 68, 63, 58, 53, 49, 44, 40, 36, 32, 28, 25, 22,
19, 16, 14, 11, 9, 7, 5, 4, 3, 2, 1, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 7, 9, 11, 14, 16,
19, 22, 25, 28, 32, 36, 40, 44, 49, 53, 58, 63, 68, 74, 80, 85, 91, 98, 104, 111,
117, 124, 131, 138, 146, 153, 161, 169, 177, 185, 194, 202, 211, 219, 228, 237, 246,
256, 265, 274, 284, 293, 303, 313, 323, 333, 343, 353, 363, 374, 384, 394, 405, 415,
426, 436, 447, 458, 468, 479, 490, 500};
uint32_t saw_LUT[NS] = {0, 3, 6, 10, 13, 17, 20, 23, 27, 30, 34, 37, 40, 44, 47, 51,
54, 57, 61, 64, 68, 71, 75, 78, 81, 85, 88, 92, 95, 98, 102, 105, 109, 112, 115, 119,
122, 126, 129, 132, 136, 139, 143, 146, 150, 153, 156, 160, 163, 167, 170, 173, 177,
180, 184, 187, 190, 194, 197, 201, 204, 208, 211, 214, 218, 221, 225, 228, 231, 235,
238, 242, 245, 248, 252, 255, 259, 262, 265, 269, 272, 276, 279, 283, 286, 289, 293,
296, 300, 303, 306, 310, 313, 317, 320, 323, 327, 330, 334, 337, 341, 344, 347, 351,
354, 358, 361, 364, 368, 371, 375, 378, 381, 385, 388, 392, 395, 398, 402, 405, 409,
412, 416, 419, 422, 426, 429, 433, 436, 439, 443, 446, 450, 453, 456, 460, 463, 467,
470, 473, 477, 480, 484, 487, 491, 494, 497, 501, 504, 508, 511, 514, 518, 521, 525,
528, 531, 535, 538, 542, 545, 549, 552, 555, 559, 562, 566, 569, 572, 576, 579, 583,
586, 589, 593,596, 600, 603, 606, 610, 613, 617, 620, 624, 627, 630, 634, 637, 641,
644, 647, 651, 654, 658, 661, 664, 668, 671, 675, 678, 682, 685, 688, 692, 695, 699,
702, 705, 709, 712, 716, 719, 722, 726, 729, 733, 736, 739, 743, 746, 750, 753, 757,
760, 763, 767, 770, 774, 777, 780, 784, 787, 791, 794, 797, 801, 804, 808, 811, 814,
818, 821, 825, 828, 832, 835, 838, 842, 845, 849, 852, 855, 859, 862, 866, 869, 872,
876, 879, 883, 886, 890, 893, 896, 900, 903, 907, 910, 913, 917, 920, 924, 927, 930,
934, 937, 941, 944, 947, 951, 954, 958, 961, 965, 968, 971, 975, 978, 982, 985, 988,
992, 995, 999, 1002, 1005, 1009, 1012, 1016, 1019};
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```
uint32_t triangle_LUT[NS] = {0, 7, 14, 20, 27, 34, 41, 48, 55, 61, 68, 75, 82, 89,
95, 102, 109, 116, 123, 130, 136, 143, 150, 157, 164, 170, 177, 184, 191, 198, 205,
211, 218, 225, 232, 239, 246, 252, 259, 266, 273, 280, 286, 293, 300, 307, 314, 321,
327, 334, 341, 348, 355, 361, 368, 375, 382, 389, 396, 402, 409, 416, 423, 430, 436,
443, 450, 457, 464, 471, 477, 484, 491, 498, 505, 512, 518, 525, 532, 539, 546, 552,
559, 566, 573, 580, 587, 593, 600, 607, 614, 621, 627, 634, 641, 648, 655, 662, 668,
675, 682, 689, 696, 702, 709, 716, 723, 730, 737, 743, 750, 757, 764, 771, 777, 784,
791, 798, 805, 812, 818, 825, 832, 839, 846, 852, 859, 866, 873, 880, 887, 893, 900,
907, 914, 921, 928, 934, 941, 948, 955, 962, 968, 975, 982, 989, 996, 1003, 1009,
1016, 1023, 1016, 1009, 1003, 996, 989, 982, 975, 968, 962, 955, 948, 941, 934, 928,
921, 914, 907, 900, 893,887, 880, 873, 866, 859, 852, 846, 839, 832, 825, 818, 812,
805, 798, 791, 784, 777, 771, 764, 757, 750, 743, 737, 730, 723, 716, 709, 702, 696,
689, 682, 675, 668, 662, 655, 648, 641, 634, 627, 621, 614, 607, 600, 593, 587, 580,
573, 566, 559, 552, 546, 539, 532, 525, 518, 512, 505, 498, 491, 484, 477, 471, 464,
457, 450, 443, 436, 430, 423, 416, 409, 402, 396, 389, 382, 375, 368, 361, 355, 348,
341, 334, 327, 321, 314, 307, 300, 293, 286, 280, 273, 266, 259, 252, 246, 239, 232,
225, 218, 211, 205, 198, 191, 184, 177, 170, 164, 157, 150, 143, 136, 130, 123, 116,
109, 102, 95, 89, 82, 75, 68, 61, 55, 48, 41, 34, 27, 20, 14, 7};

// 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();
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```
  /* 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, (uint32_t)(&TIM3->CCR3), NS);

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

  // 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;
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```
  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 */

}

/**
  * @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
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```
  * @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)
{      first=HAL_GetTick();


if (HAL_GPIO_ReadPin(Button0_GPIO_Port, Button0_Pin) == GPIO_PIN_SET) {
       // Debounce using HAL_GetTick()
       if (first - last >= delay_t) {
          last = first;

          LUT++;
                if (LUT==3){LUT=0;};

           // Disable DMA transfer and abort IT
           __HAL_TIM_DISABLE_DMA(&htim2, TIM_DMA_CC1);
           HAL_DMA_Abort_IT(&hdma_tim2_ch1);

           // Switch to the next LUT
           switch (LUT) {
```

```
                case 0:
                    // Configure DMA for LUT 1
                    HAL_DMA_Start_IT(&hdma_tim2_ch1, (uint32_t)Sin_LUT, DestAddress,
NS);
                    lcd_command(CLEAR);
                    lcd_putstring(sine);
                    break;

                case 1:
                 // Configure DMA for LUT 0
                  HAL_DMA_Start_IT(&hdma_tim2_ch1, (uint32_t)saw_LUT, DestAddress,
NS);
                  lcd_command(CLEAR);
                  lcd_putstring(saw);
                    break;

                case 2:

                    // Configure DMA for LUT 2
                   HAL_DMA_Start_IT(&hdma_tim2_ch1, (uint32_t)triangle_LUT,
DestAddress, NS);
                    lcd_command(CLEAR);
                    lcd_putstring(Triangle);
                     break;

                default:
                    LUT = 0;
                    break;
            }

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

    }

}
HAL_GPIO_EXTI_IRQHandler(Button0_Pin);
}
/* 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 */
}
```

TMBTIN004-Tinashe Timba
CHHANK001-Ankush Chohan

```c
#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 */
```