EEE3096S Prac 1

HWKLAW001 (Lawrence Hawke) and DWDZAI001 (Zainodine Dawood)

August 2024

1 Code

1.1 Variables

```
/* USER CODE BEGIN PV */
// TODO: Define input variables

    uint8_t patterns[9][8] = {
{0,0,0,0,0,0,0,0},
{1,1,1,0,1,0,0,1},
{1,1,0,1,0,0,1,0},
{1,0,1,0,0,1,0,0},
{0,1,0,0,1,0,0,0},
{1,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{1,0,0,0,0,0,0,0}
};

    uint8_t counterPattern=0; //counter

    void SetLEDs(uint8_t *pattern); //defining function //*pattern makes 1d
array type
    /* USER CODE END PV */
```

1.2 Start Timer

```
/* USER CODE BEGIN 2 */

    // TODO: Start timer TIM16

    HAL_TIM_Base_Start_IT(htim16);

    /* USER CODE END 2 */
```

1.3 Push Buttons

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

    /* USER CODE BEGIN 3 */

    // TODO: Check pushbuttons to change timer delay

if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_RESET) {
__HAL_TIM_SET_AUTORELOAD(htim16, (1000/2)-1); //0.5s delay COMMENT
init_LCD(); //initialise and clear LCD for adding a sentence to LCD.
lcd_command(CLEAR);
lcd_putstring(”0.5s TIMER”);
}
else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == GPIO_PIN_RESET){
__HAL_TIM_SET_AUTORELOAD(htim16, (2000)-1); //2s delay COMMENT
init_LCD();
lcd_command(CLEAR);
lcd_putstring(”2s TIMER”);
}
else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2) == GPIO_PIN_RESET){
__HAL_TIM_SET_AUTORELOAD(htim16, (1000)-1); //1s delay COMMENT
init_LCD();
lcd_command(CLEAR);
lcd_putstring(”1s TIMER”);
}
else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_3) == GPIO_PIN_RESET){
counterPattern = 1; //for resetting the the patterns. COMMENT
SetLEDs(patterns[counterPattern]);
init_LCD();
lcd_command(CLEAR);
lcd_putstring(”RESET PATTERN...”);

    HAL_Delay(10); //Small Delay to debounce the buttons
}

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

1.4 setLEDs() and TIM Handler

```
/* USER CODE BEGIN 4 */

    // Function to set LEDs to the values held in the pattern variable.

    void SetLEDs(uint8_t *pattern){
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, pattern[0]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, pattern[1]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, pattern[2]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, pattern[3]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, pattern[4]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, pattern[5]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, pattern[6]);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, pattern[7]);
}

    // Timer rolled over
void TIM16_IRQHandler(void)
{
// Acknowledge interrupt
HAL_TIM_IRQHandler(htim16);

    // TODO: Change LED pattern
// print something
__HAL_TIM_CLEAR_IT(htim16, TIM_IT_UPDATE); //CLEAR AND UPDATE THE TIMER

    //update pattern
counterPattern = (counterPattern + 1)%9; //Update the pattern via counterPattern variable
SetLEDs(patterns[counterPattern]); //set LEDs

    }
/* USER CODE END 4 */
```

2 Git Hub Link

https://github.com/Lawrenceismyname/EEE3096Spracs

3 Code Summary

The first variable created is a 9x8 array called ‘patterns’ that contains the 9 patterns required for display in practical 1. The display

**starts with the second row in the pattern array. This is because the array cycle starts when the timer starts, which calls an interrupt incrementing the counterPattern variable, hence the first iteration of the 'pattern' variable displayed is the second row of the array.
The second variable is 'counterPattern' and is used as a counter to cycle 0 through 8. Using this line of code:
counterPattern = (counterPattern + 1)%9;**

A function is then defined called 'SetLEDs' that uses the variable '*pattern', which is a 1d array type created from the variable 'patterns' at index 'counterPattern'. The function itself writes to the output pins of the STM32. Each pattern is cycled for the the time specified by the timer.

**Our solution then makes use of the infinite while loop to check if any of the buttons are pressed and then reacts accordingly.
The first if statement checks if the button 0 is pushed and if so, delays the timer to 0.5 seconds.
The second if statement checks if button is 1 is pushed and changes the timer to a 2 second delay.
If button 2 is pushed it restores the delay to 1 second.
If button 3 is pushed the patter is resets and starts again at pattern 1 while displaying a message on the LCD. A 10ms delay is used to make sure once the reset button is pushed any other delays do not interfere with it.**

4 Appendix

Full main.c file is attached below:

```
/* 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 <stdint.h>
#include "stm32f0xx.h"

#include <lcd_stm32f0.c>
/* wadddup */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

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

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim16;

/* USER CODE BEGIN PV */
// TODO: Define input variables

uint8_t patterns[9][8] = {
                {0,0,0,0,0,0,0,0},
                {1,1,1,0,1,0,0,1},
                {1,1,0,1,0,0,1,0},
                {1,0,1,0,0,1,0,0},
                {0,1,0,0,1,0,0,0},
                {1,0,0,1,0,0,0,0},
                {0,0,1,0,0,0,0,0},
                    {0,1,0,0,0,0,0,0},
                    {1,0,0,0,0,0,0,0}
};

uint8_t counterPattern=0;   //counter

void SetLEDs(uint8_t *pattern);     //defining function //*pattern makes 1d array type



/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM16_Init(void);
/* USER CODE BEGIN PFP */
void TIM16_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 */
  /* 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_TIM16_Init();
  /* USER CODE BEGIN 2 */

  // TODO: Start timer TIM16

  HAL_TIM_Base_Start_IT(&htim16);

  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    // TODO: Check pushbuttons to change timer delay
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_RESET) {
            __HAL_TIM_SET_AUTORELOAD(&htim16, (1000/2)-1);  //0.5s delay
            init_LCD();                          //initialise and clear LCD for adding a
sentence to LCD.
            lcd_command(CLEAR);
            lcd_putstring("0.5s TIMER");
        }
            else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) ==
GPIO_PIN_RESET){
                __HAL_TIM_SET_AUTORELOAD(&htim16, (2000)-1);  //2s
delay
                init_LCD();
                lcd_command(CLEAR);
                lcd_putstring("2s TIMER");
            }
            else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2) ==
GPIO_PIN_RESET){
                __HAL_TIM_SET_AUTORELOAD(&htim16, (1000)-1);  //1s
delay
                init_LCD();
                lcd_command(CLEAR);
                lcd_putstring("1s TIMER");
            }
            else if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_3) ==
GPIO_PIN_RESET){
                counterPattern = 1; //for resetting the the patterns.
                SetLEDs(patterns[counterPattern]);
                init_LCD();
                lcd_command(CLEAR);
                lcd_putstring("RESET PATTERN...");



                HAL_Delay(10);          //Small Delay to debounce the
buttons
            }

  }
  /* 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 TIM16 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM16_Init(void)
{

  /* USER CODE BEGIN TIM16_Init 0 */

  /* USER CODE END TIM16_Init 0 */

  /* USER CODE BEGIN TIM16_Init 1 */

  /* USER CODE END TIM16_Init 1 */
  htim16.Instance = TIM16;
  htim16.Init.Prescaler = 8000-1;
  htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim16.Init.Period = 1000-1;
  htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim16.Init.RepetitionCounter = 0;
  htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM16_Init 2 */
  NVIC_EnableIRQ(TIM16_IRQn);
  /* USER CODE END TIM16_Init 2 */

}
```

```
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_GPIO_InitTypeDef GPIO_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_GPIO_ResetOutputPin(LED0_GPIO_Port, LED0_Pin);

  /**/
  LL_GPIO_ResetOutputPin(LED1_GPIO_Port, LED1_Pin);

  /**/
  LL_GPIO_ResetOutputPin(LED2_GPIO_Port, LED2_Pin);

  /**/
  LL_GPIO_ResetOutputPin(LED3_GPIO_Port, LED3_Pin);
```

```
/**/
LL_GPIO_ResetOutputPin(LED4_GPIO_Port, LED4_Pin);

/**/
LL_GPIO_ResetOutputPin(LED5_GPIO_Port, LED5_Pin);

/**/
LL_GPIO_ResetOutputPin(LED6_GPIO_Port, LED6_Pin);

/**/
LL_GPIO_ResetOutputPin(LED7_GPIO_Port, LED7_Pin);

/**/
GPIO_InitStruct.Pin = Button0_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button0_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = Button1_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button1_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = Button2_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button2_GPIO_Port, &GPIO_InitStruct);
```

```
/**/
GPIO_InitStruct.Pin = Button3_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = LL_GPIO_PULL_UP;
LL_GPIO_Init(Button3_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED0_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED0_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED1_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED1_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED2_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
```

```
LL_GPIO_Init(LED2_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED3_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED3_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED4_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED4_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED5_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
LL_GPIO_Init(LED5_GPIO_Port, &GPIO_InitStruct);

/**/
GPIO_InitStruct.Pin = LED6_Pin;
GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
```

```
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LED6_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = LED7_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(LED7_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = LL_GPIO_PIN_9;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = LL_GPIO_PULL_NO;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

void SetLEDs(uint8_t *pattern){
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, pattern[0]);
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, pattern[1]);
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, pattern[2]);
```

```
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, pattern[3]);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, pattern[4]);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, pattern[5]);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, pattern[6]);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, pattern[7]);
}

// Timer rolled over
void TIM16_IRQHandler(void)
{
        // Acknowledge interrupt
        HAL_TIM_IRQHandler(&htim16);

        // TODO: Change LED pattern
        // print something
        __HAL_TIM_CLEAR_IT(&htim16, TIM_IT_UPDATE);

        //update pattern
        counterPattern = (counterPattern + 1)%9;
        SetLEDs(patterns[counterPattern]);

}

/* 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 */
```