GitHub Link

https://github.com/LuthoYRN/MNGLUT008_CBXLIS001_EEE3096S/blob/main/Prac1/main.c

Description of LED Sequence Implementation

The code initializes an array of 8-bit unsigned integers, **led_patterns[]**, which stores 9 distinct LED patterns. The variable **current** is used to track the index of the currently displayed pattern, starting from 0. The LED patterns are updated within the **TIM16_IRQHandler** function, which is called upon a timer interrupt. When the interrupt occurs, the LED output is updated to match the current pattern, and the **current** index is incremented, looping back to 0 after reaching the last pattern.

Description of Timer Interrupt Functionality

The timer interrupt functionality is handled by the **TIM16_IRQHandler** function. This function checks if the TIM16 update interrupt flag is set and clears it. It then updates the LED outputs to reflect the pattern specified by **current** variable and increments the **current** to point to the next pattern. Additionally, the code includes logic to change the timer delay based on the state of four pushbuttons. The pin states of PA0-PA3, corresponding to the pushbuttons, are checked continuously within an infinite while loop in the main method via if statements. Inside each statement, TIM16 is configured for different delays by changing the Prescaler and ARR value based on calculations (using an 8MHZ clock), except for the last pin PA3, which just resets the **current** index to 0 to restart the pattern sequence.

$$\text{Cycle Length} = \frac{1}{f_{\text{APBbus}}}$$

$$\text{Cycles Needed} = \frac{\text{Delay}}{\text{Cycles Length}}$$

$$\text{Prescaler} = \frac{\text{Cycles Needed}}{2^{16}}$$

$$\text{Auto-Reload Register} = \frac{\text{Number of Cycles}}{\text{Prescaler}}$$

Calculation used to configure Prescaler and ARR values

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

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdint.h>
#include "stm32f0xx.h"
/* 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 variable
      int current = 0;
      uint8_t led_patterns[] = {0b11101001,0b11010010,0b10100100,0b01001000,
                                0b10010000,0b00100000,0b01000000,0b10000000,0x0};

/* 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 */

```

```c
/**
  * @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 (LL_GPIO_IsInputPinSet(GPIOA, LL_GPIO_PIN_0) == 0){
           //0.5 seconds delay
           HAL_TIM_Base_Stop(&htim16);
           if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
           {
             Error_Handler();
           }
           htim16.Init.Period = 64516;
           htim16.Init.Prescaler =61;
           HAL_TIM_Base_Start_IT(&htim16);
     }
      if (LL_GPIO_IsInputPinSet(GPIOA, LL_GPIO_PIN_1)== 0){
           //2 seconds delay
           HAL_TIM_Base_Stop(&htim16);
           if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
           {
             Error_Handler();
           }
           htim16.Init.Period = 65306;
           htim16.Init.Prescaler = 245;
           HAL_TIM_Base_Start_IT(&htim16);
     }
      if (LL_GPIO_IsInputPinSet(GPIOA, LL_GPIO_PIN_2)== 0){
           //1 seconds delay
           HAL_TIM_Base_Stop(&htim16);
           if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
           {
```

```
                Error_Handler();
              }
              htim16.Init.Period = 65040;
              htim16.Init.Prescaler =123 ;

              HAL_TIM_Base_Start_IT(&htim16);
        }
         if (LL_GPIO_IsInputPinSet(GPIOA, LL_GPIO_PIN_3 )== 0){
              //back to pattern 1
              current = 0;
        }
   }
   /* 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 = 123;
  htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim16.Init.Period = 65040;
  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 */

// Timer rolled over
void TIM16_IRQHandler(void)
{
      // Acknowledge interrupt
      HAL_TIM_IRQHandler(&htim16);
      // TODO: Change LED pattern
      if (__HAL_TIM_GET_IT_SOURCE(&htim16, TIM_IT_UPDATE)) {
            __HAL_TIM_CLEAR_FLAG(&htim16, TIM_FLAG_UPDATE);
           LL_GPIO_WriteOutputPort(LED0_GPIO_Port,led_patterns[current]);
             current++;
             if (current>8)current=0;
       }
}
/* 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 */
```