MNGLUT008

EEE3096S

CBXLIS001

GitHub Link

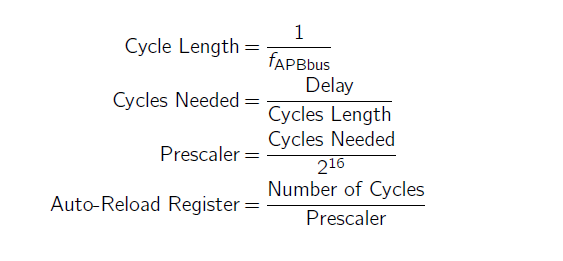
<https://github.com/LuthoYRN/MNGLUT008_CBXLIS001_EEE3096S/blob/main/Prac1/main.c>

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.



Calculation used to configure Prescaler and ARR values

Appendix

/\* 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 \*/

/\*\*

\* @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 \*/