Національний технічний університет України

“ Київський політехнічний інститут ”



Лабораторна робота №1

По курсу: ” МІКРОПРОЦЕСОРНІ СИСТЕМИ НА БАЗІ ARM ПРОЦЕСОРІВ*”*

Тема: “ ”

Виконав:Студент І курсу

Групи ДС-31мп

Рєзнік Олександр Сергійович

**Київ 2024 р.**

***Мета роботи:*** ознайомлення з середовищем розробки STM32CubeIDE. Ініціалізація, налаштування та використання портів та таймерів

Завдання

Порти введення-виведення:

1. Ініціалізувати вивід порту PA, за номером варіанту = PA6,на вихід, та вивід РВ0 на вхід у режимі підтяжки “No pull-up, no pull-down” (floating input).

2. Створити програмний код, що реалізує нескінченне опитування виводу РВ0 на предмет присутності вхідного сигналу. За присутності вхідного сигналу, відбувається підключення периферії підключеної до ініціалізованого виводу порту РА. Врахувати необхідність усунення явища «тремтіння контактів» на програмному рівні.

3. Ініціалізувати вивід порту PA за подвоєним номером варіанту =PA12 на вихід, та вивід порту РВ за номером варіанту =PB6 на вхід у режимі генерації зовнішнього переривання “EXTI”, підтяжка в режимі “Pull-down”.

4. Створити програмний код, що реалізує підключення периферії підключеної до ініціалізованого виводу порту РА за виникнення зовнішнього переривання від ініціалізованого вхідного виводу порту РВ.

5. Створити модель у середовищі “Proteus”, що демонструє роботу створеного коду. Врахувати необхідність усунення явища «тремтіння контактів» на апаратному рівні.

Таймери:

1. Налаштувати таймер ТІМ 1. Часовий інтервал розраховується за формулою: N\*2000+1000\*mod(N,2) мс, де N – номер варіанту. Підключити роботу з перериваннями. 6\*2000+1000\*mod(6,2)=12000 мс.

2. Ініціалізувати вивід порту PA за подвоєним номером варіанту =PA12 на вихід, вивід порту PB0 – на вихід. (вхід)

3. Створити програмний код, що реалізує запуск таймеру за появи вхідного сигналу на виводі РВ0. Таймер, за досягнення заданого часового інтервалу ініціалізує запуск периферії підключеної до ініціалізованого виводу порту РА. Врахувати необхідність усунення явища «тремтіння контактів».

4. Створити програмний код, що реалізує зміну часового інтервалу вимірювання таймера через певний часовий інтервал після запуску таймеру відповідно до п.3 на половину від величини інтервалу визначеної в п.3.

5. Створити модель у середовищі “Proteus”, що демонструє роботу створеного коду.

Зміст звіту

1. Титульна сторінка

2. Завдання варіанту, визначене через список груп студентів за порядковим номером

3. Лістинг програмного коду з коментарями

4. Скріншоти реалізації моделі у середовищі “Proteus”, що демонструє її працездатність

5. Короткі висновки по виконаній роботі.

Лістинг програми

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2024 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 \*/

/\* 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** htim1;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_TIM1\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* 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\_TIM1\_Init**();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

//if(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0)==GPIO\_PIN\_SET)

**if**(GPIOB->IDR & GPIO\_IDR\_IDR0) //якщо кнопка на РВ0 натиснута

{

**HAL\_Delay**(250); // встановлюємо затримку для нейтралізації "тремтіння контактів"

**if**(GPIOB->IDR & GPIO\_IDR\_IDR0) // якщо кнопка все ще нетиснута то виконуємо код

{

**HAL\_GPIO\_WritePin**(GPIOB, GPIO\_PIN\_9, *GPIO\_PIN\_SET*); // включаємо світлодіод на піні РВ9

**HAL\_TIM\_Base\_Start\_IT**(&htim1); // запускаємо таймер та переривання від таймера ТІМ1

**HAL\_Delay**(20000); // затримка протягом якої період роботи таймера не змінюється

**HAL\_TIM\_Base\_Stop\_IT**(&htim1); // зупиняємо таймер ТІМ1 і забороняємо його перериваання

\_\_HAL\_TIM\_SET\_AUTORELOAD(&htim1, 0x03E7); //завантажуємо нове значення регістру перезавантаження таймеру у 999 у hex-форматі.При нашому подільнику частоти період таймеру складатиме 1 секунду.

\_\_HAL\_TIM\_SET\_COUNTER(&htim1, 0x0000); // завантажуємо нульове значення у розрахунковий регістр для відрахунку часовогг інтервалу спочатку

**HAL\_TIM\_Base\_Start\_IT**(&htim1); // знову запускаємо таймер ТІМ1 і дозволяємо його переривання

}

}

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

**RCC\_OscInitTypeDef** RCC\_OscInitStruct = {0};

**RCC\_ClkInitTypeDef** RCC\_ClkInitStruct = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

**if** (**HAL\_RCC\_OscConfig**(&RCC\_OscInitStruct) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (**HAL\_RCC\_ClockConfig**(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != *HAL\_OK*)

{

**Error\_Handler**();

}

}

/\*\*

\* @brief TIM1 Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_TIM1\_Init**(**void**)

{

/\* USER CODE BEGIN TIM1\_Init 0 \*/

/\* USER CODE END TIM1\_Init 0 \*/

**TIM\_ClockConfigTypeDef** sClockSourceConfig = {0};

**TIM\_MasterConfigTypeDef** sMasterConfig = {0};

/\* USER CODE BEGIN TIM1\_Init 1 \*/

/\* USER CODE END TIM1\_Init 1 \*/

htim1.Instance = TIM1;

htim1.Init.Prescaler = 7999;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 4999;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (**HAL\_TIM\_Base\_Init**(&htim1) != *HAL\_OK*)

{

**Error\_Handler**();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (**HAL\_TIM\_ConfigClockSource**(&htim1, &sClockSourceConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (**HAL\_TIMEx\_MasterConfigSynchronization**(&htim1, &sMasterConfig) != *HAL\_OK*)

{

**Error\_Handler**();

}

/\* USER CODE BEGIN TIM1\_Init 2 \*/

/\* USER CODE END TIM1\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

**GPIO\_InitTypeDef** GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

**HAL\_GPIO\_WritePin**(GPIOB, GPIO\_PIN\_10|GPIO\_PIN\_15|GPIO\_PIN\_9, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : PB0 PB1 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

**HAL\_GPIO\_Init**(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pin : PB2 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_2;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

**HAL\_GPIO\_Init**(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : PB10 PB15 PB9 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_10|GPIO\_PIN\_15|GPIO\_PIN\_9;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

**HAL\_GPIO\_Init**(GPIOB, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

**HAL\_NVIC\_SetPriority**(*EXTI2\_IRQn*, 0, 0);

**HAL\_NVIC\_EnableIRQ**(*EXTI2\_IRQn*);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

/\* 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 \*/

/////////////////

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file stm32f1xx\_it.c

\* @brief Interrupt Service Routines.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2024 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"

**#include** "stm32f1xx\_it.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN TD \*/

/\* USER CODE END TD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\* External variables --------------------------------------------------------\*/

**extern** **TIM\_HandleTypeDef** htim1;

/\* USER CODE BEGIN EV \*/

/\* USER CODE END EV \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Cortex-M3 Processor Interruption and Exception Handlers \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles Non maskable interrupt.

\*/

**void** **NMI\_Handler**(**void**)

{

/\* USER CODE BEGIN NonMaskableInt\_IRQn 0 \*/

/\* USER CODE END NonMaskableInt\_IRQn 0 \*/

/\* USER CODE BEGIN NonMaskableInt\_IRQn 1 \*/

**while** (1)

{

}

/\* USER CODE END NonMaskableInt\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Hard fault interrupt.

\*/

**void** **HardFault\_Handler**(**void**)

{

/\* USER CODE BEGIN HardFault\_IRQn 0 \*/

/\* USER CODE END HardFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_HardFault\_IRQn 0 \*/

/\* USER CODE END W1\_HardFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Memory management fault.

\*/

**void** **MemManage\_Handler**(**void**)

{

/\* USER CODE BEGIN MemoryManagement\_IRQn 0 \*/

/\* USER CODE END MemoryManagement\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_MemoryManagement\_IRQn 0 \*/

/\* USER CODE END W1\_MemoryManagement\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Prefetch fault, memory access fault.

\*/

**void** **BusFault\_Handler**(**void**)

{

/\* USER CODE BEGIN BusFault\_IRQn 0 \*/

/\* USER CODE END BusFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_BusFault\_IRQn 0 \*/

/\* USER CODE END W1\_BusFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles Undefined instruction or illegal state.

\*/

**void** **UsageFault\_Handler**(**void**)

{

/\* USER CODE BEGIN UsageFault\_IRQn 0 \*/

/\* USER CODE END UsageFault\_IRQn 0 \*/

**while** (1)

{

/\* USER CODE BEGIN W1\_UsageFault\_IRQn 0 \*/

/\* USER CODE END W1\_UsageFault\_IRQn 0 \*/

}

}

/\*\*

\* @brief This function handles System service call via SWI instruction.

\*/

**void** **SVC\_Handler**(**void**)

{

/\* USER CODE BEGIN SVCall\_IRQn 0 \*/

/\* USER CODE END SVCall\_IRQn 0 \*/

/\* USER CODE BEGIN SVCall\_IRQn 1 \*/

/\* USER CODE END SVCall\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Debug monitor.

\*/

**void** **DebugMon\_Handler**(**void**)

{

/\* USER CODE BEGIN DebugMonitor\_IRQn 0 \*/

/\* USER CODE END DebugMonitor\_IRQn 0 \*/

/\* USER CODE BEGIN DebugMonitor\_IRQn 1 \*/

/\* USER CODE END DebugMonitor\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles Pendable request for system service.

\*/

**void** **PendSV\_Handler**(**void**)

{

/\* USER CODE BEGIN PendSV\_IRQn 0 \*/

/\* USER CODE END PendSV\_IRQn 0 \*/

/\* USER CODE BEGIN PendSV\_IRQn 1 \*/

/\* USER CODE END PendSV\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles System tick timer.

\*/

**void** **SysTick\_Handler**(**void**)

{

/\* USER CODE BEGIN SysTick\_IRQn 0 \*/

/\* USER CODE END SysTick\_IRQn 0 \*/

**HAL\_IncTick**();

/\* USER CODE BEGIN SysTick\_IRQn 1 \*/

/\* USER CODE END SysTick\_IRQn 1 \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* STM32F1xx Peripheral Interrupt Handlers \*/

/\* Add here the Interrupt Handlers for the used peripherals. \*/

/\* For the available peripheral interrupt handler names, \*/

/\* please refer to the startup file (startup\_stm32f1xx.s). \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles EXTI line2 interrupt.

\*/

**void** **EXTI2\_IRQHandler**(**void**)

{

/\* USER CODE BEGIN EXTI2\_IRQn 0 \*/

**HAL\_GPIO\_TogglePin**(GPIOB, GPIO\_PIN\_15);

/\* USER CODE END EXTI2\_IRQn 0 \*/

**HAL\_GPIO\_EXTI\_IRQHandler**(GPIO\_PIN\_2);

/\* USER CODE BEGIN EXTI2\_IRQn 1 \*/

/\* USER CODE END EXTI2\_IRQn 1 \*/

}

/\*\*

\* @brief This function handles TIM1 update interrupt.

\*/

**void** **TIM1\_UP\_IRQHandler**(**void**)

{

/\* USER CODE BEGIN TIM1\_UP\_IRQn 0 \*/

**HAL\_GPIO\_TogglePin**(GPIOB, GPIO\_PIN\_10); // блимаємо світлодіодом при приході переривання переповненням

/\* USER CODE END TIM1\_UP\_IRQn 0 \*/

**HAL\_TIM\_IRQHandler**(&htim1);

/\* USER CODE BEGIN TIM1\_UP\_IRQn 1 \*/

/\* USER CODE END TIM1\_UP\_IRQn 1 \*/

}

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

// Опитування PB0

if(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == GPIO\_PIN\_SET) // Якщо кнопка натиснута

{

HAL\_Delay(50); // Анти-дребезг

if(HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == GPIO\_PIN\_SET) // Перевірка після затримки

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, GPIO\_PIN\_SET); // Активуємо PA6

}

}

else

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, GPIO\_PIN\_RESET); // Деактивуємо PA6, якщо сигнал відсутній

}