



Lab 4

Tues 14:15

Jack Landers Henry Fang

Introduction

We will create an LCD timer in this lab.

Procedure

Step 1

Create a new STM32CubeMX project and name it LCD Timer. Configure the GPIO pins to get it ready to send interrupts.

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO Single Mapped Signals LCD RCC SYS NVIC

Search Signals

Search (Ctrl+F) ☐ Show only Modified Pins

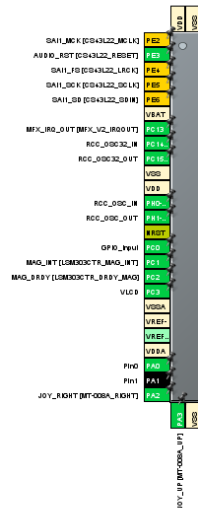
Pin N...	Signal on	GPIO out...	GPIO mode	GPIO Pull...	Maximum ...	Fast Mode	User Label	Modified
PA0	n/a	n/a	External In...	Pull-down	n/a	n/a	Pin0	<input checked="" type="checkbox"/>
PA1	n/a	n/a	External In...	Pull-down	n/a	n/a	Pin1	<input checked="" type="checkbox"/>
PA2	n/a	n/a	Input mode	Pull-down	n/a	n/a	JOY_RIG...	<input checked="" type="checkbox"/>
PA3	n/a	n/a	Input mode	Pull-down	n/a	n/a	JOY_UP [...]	<input checked="" type="checkbox"/>
PA4	n/a	n/a	External ...	No pull-up...	n/a	n/a	MFX_WA...	<input checked="" type="checkbox"/>
PA5	n/a	n/a	Input mode	Pull-down	n/a	n/a	JOY_DO...	<input checked="" type="checkbox"/>
PB2	n/a	Low	Output Pu...	Pull-up	Very High	n/a	LD_R [LE...	<input checked="" type="checkbox"/>
PB3 (JTD...	n/a	Low	Output Pu...	No pull-up...	Low	n/a	M3V3_R...	<input checked="" type="checkbox"/>
PB8	n/a	n/a	External ...	No pull-up...	n/a	n/a	GYRO_IN...	<input checked="" type="checkbox"/>
PC0	n/a	n/a	Input mode	No pull-up...	n/a	n/a		<input type="checkbox"/>

PA1 Configuration :

GPIO mode External Interrupt Mode with Rising edge trigger detection

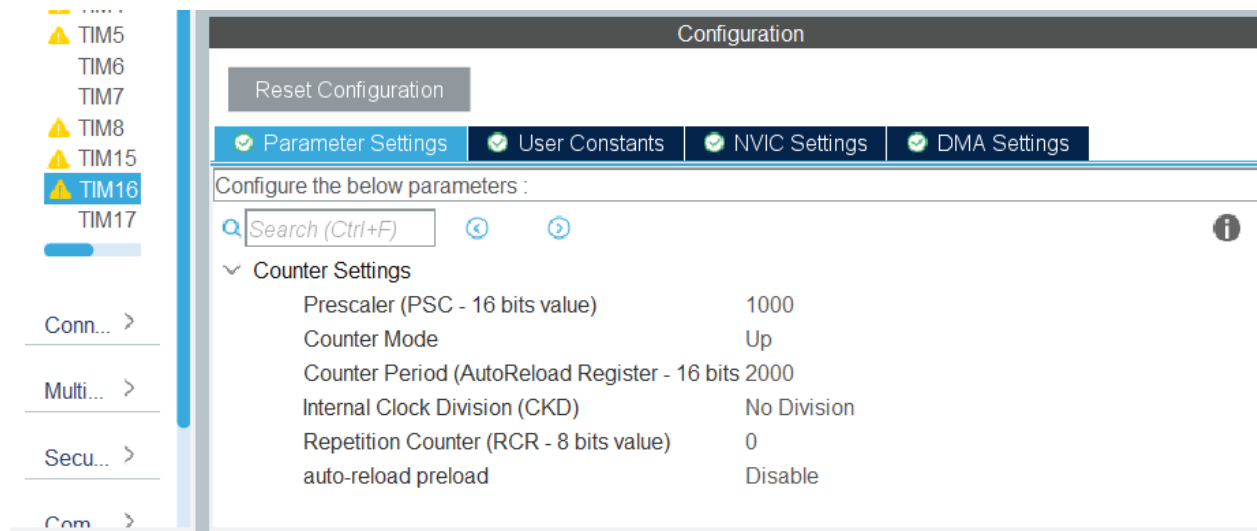
GPIO Pull-up/Pull-down Pull-down

User Label Pin1



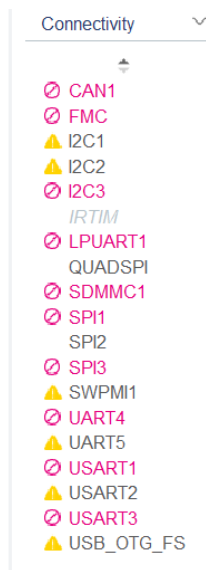
Step 2

Enable Timer TIM16, and set up its prescaler and counter period values.



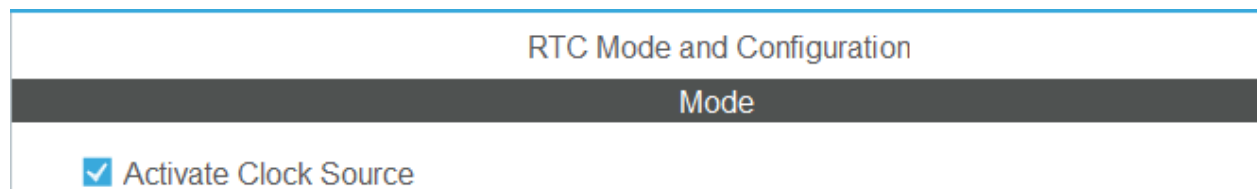
Step 3

Disable unused hardware.



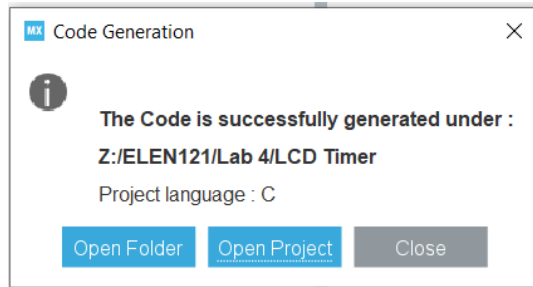
Step 4

enable the real-time clock (RTC) in the Timers section and select "Activate Clock Source."



Step 5

Generate the Keil project. Check the Keil setting and compile.



Step 6

Find the interrupt service routine. Test their functionality with Red and Green LEDs.

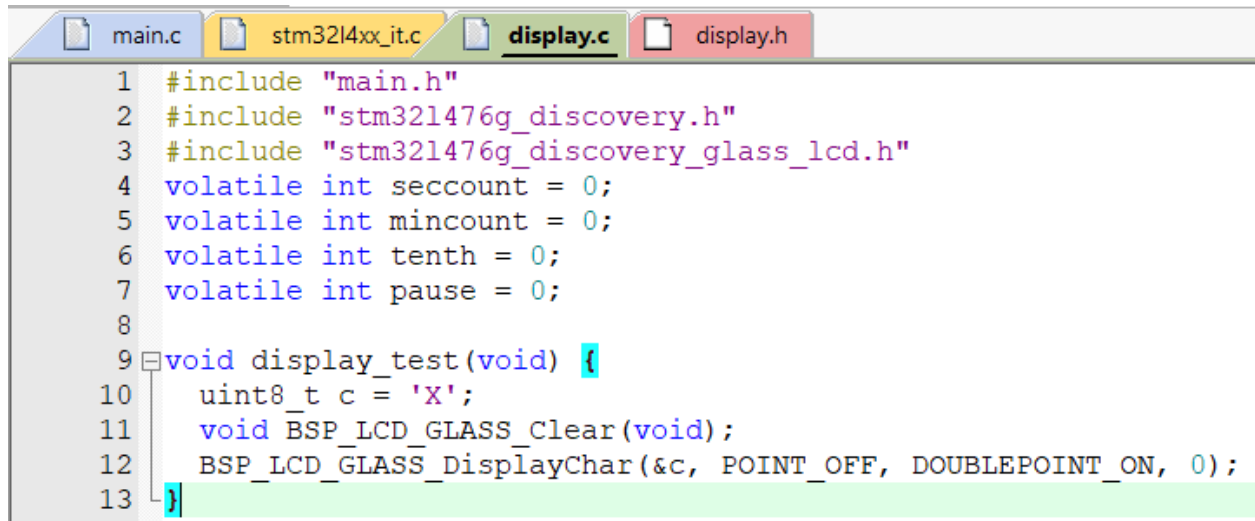
```
209 void EXTI0_IRQHandler(void)
210 {
211     /* USER CODE BEGIN EXTI0_IRQn 0 */
212
213     /* USER CODE END EXTI0_IRQn 0 */
214     HAL_GPIO_EXTI_IRQHandler(Pin0_Pin);
215     /* USER CODE BEGIN EXTI0_IRQn 1 */
216     HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
217     pause++;
218     pause%=2;
219     /* USER CODE END EXTI0_IRQn 1 */
220 }
221
222 /**
223  * @brief This function handles EXTI line1 interrupt.
224  */
225 void EXTI1_IRQHandler(void)
226 {
227     /* USER CODE BEGIN EXTI1_IRQn 0 */
228
229     /* USER CODE END EXTI1_IRQn 0 */
230     HAL_GPIO_EXTI_IRQHandler(Pin1_Pin);
231     /* USER CODE BEGIN EXTI1_IRQn 1 */
232     HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
233     seccount = 0;
234     mincount = 0;
235     tenth = 0;
236     /* USER CODE END EXTI1_IRQn 1 */
237 }
238
```

Step 7

Import the display.c to the project Src directory. Add some variables to display.c. Also include the main.h to display.c.

Also, add those variables to the stm32l4xx_it.c.

Import the "stm32l476g_discovery_glass_lcd_wolfemod.c" and "stm32l476g_discovery_glass_lcd_wolfemod.h" to the project.

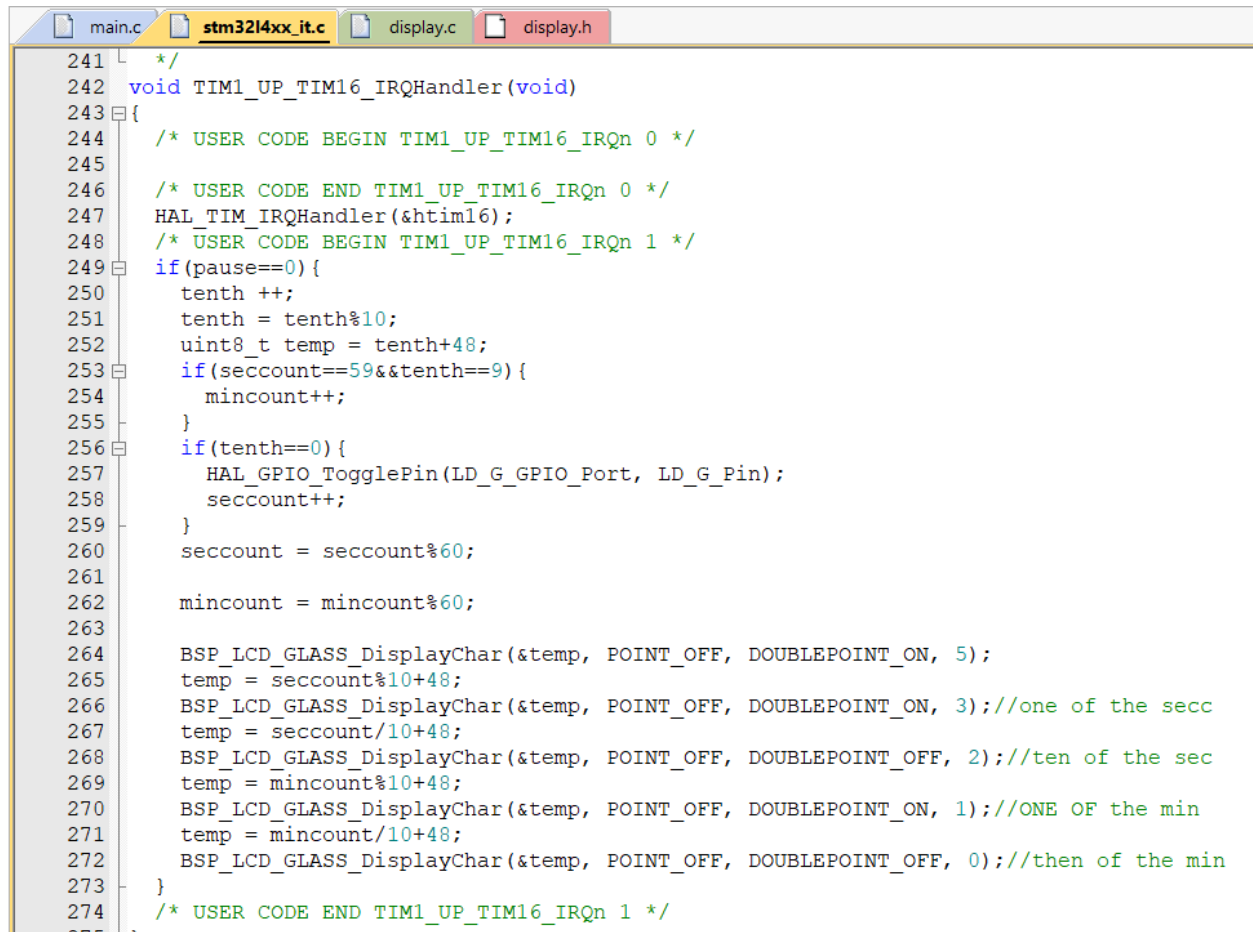


```
1  #include "main.h"
2  #include "stm32l476g_discovery.h"
3  #include "stm32l476g_discovery_glass_lcd.h"
4  volatile int seccount = 0;
5  volatile int mincount = 0;
6  volatile int tenth = 0;
7  volatile int pause = 0;
8
9  void display_test(void) {
10     uint8_t c = 'X';
11     void BSP_LCD_GLASS_Clear(void);
12     BSP_LCD_GLASS_DisplayChar(&c, POINT_OFF, DOUBLEPOINT_ON, 0);
13 }
```

Step 8

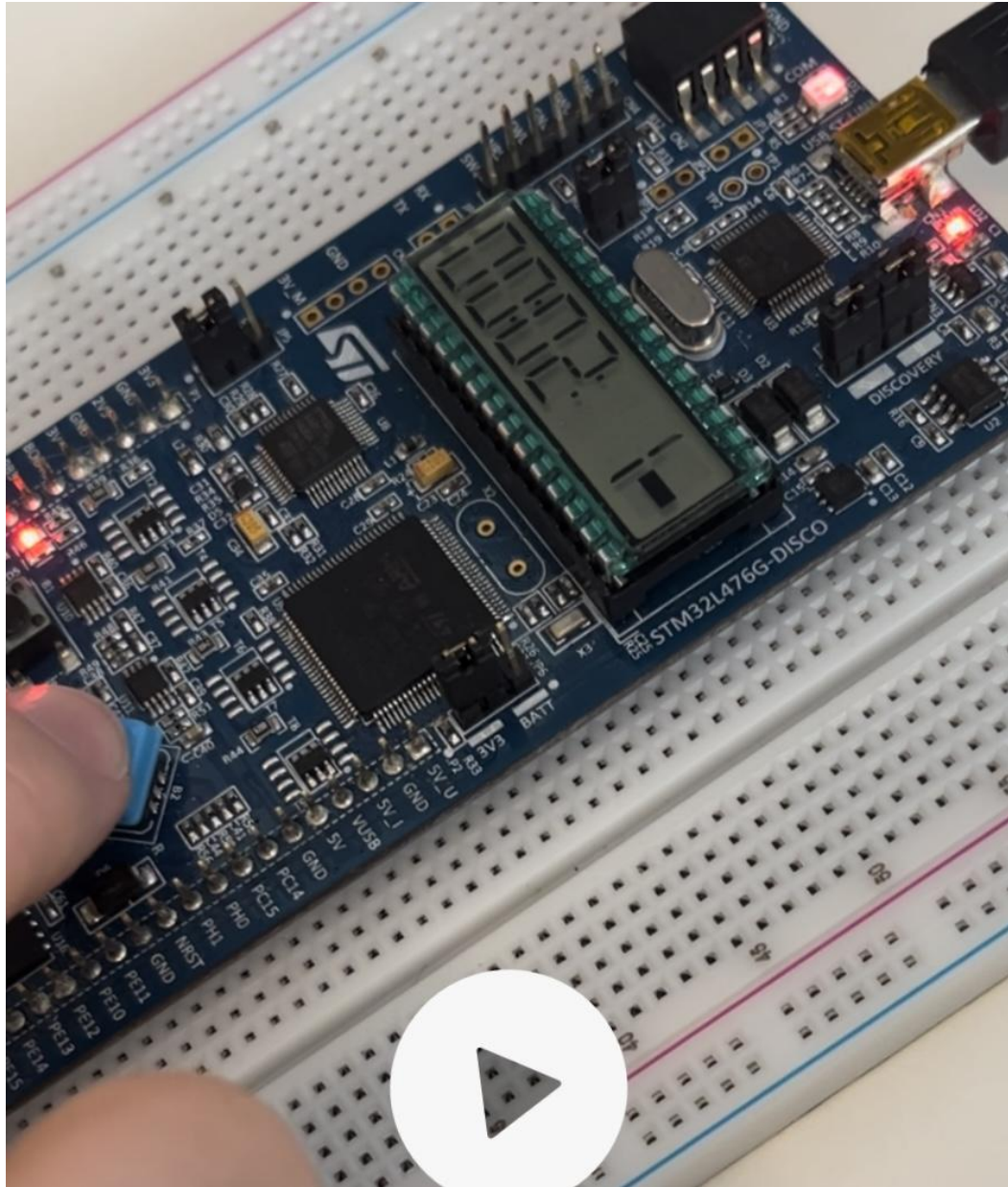
Finish the project.

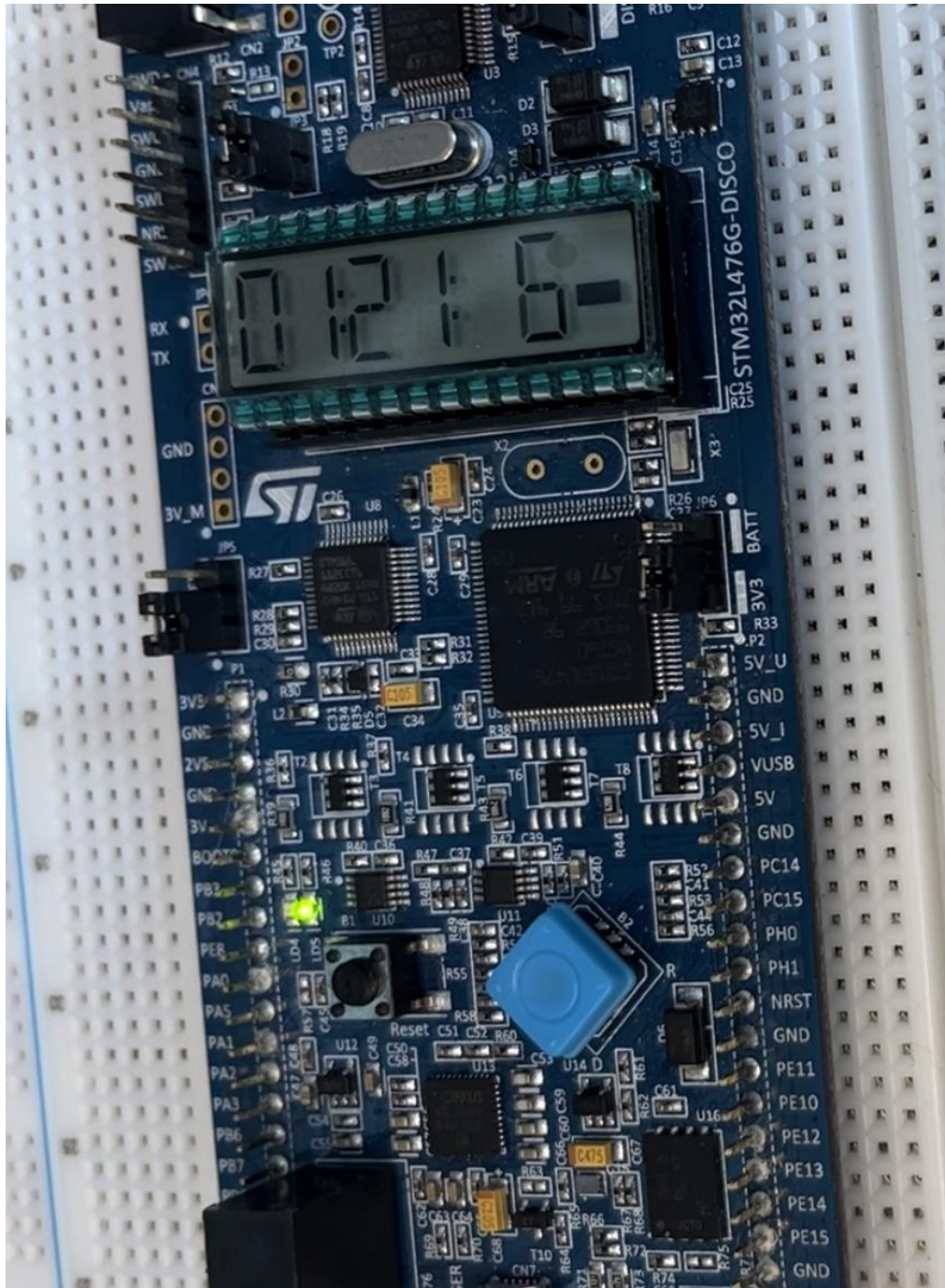
Connect the interrupts with an algorithm that writes the current numerical time to the LCD Glass.

A screenshot of a code editor window with four tabs: 'main.c', 'stm32l4xx_it.c', 'display.c', and 'display.h'. The 'stm32l4xx_it.c' tab is active, showing C code for a TIM16 interrupt handler. The code is numbered from 241 to 274. It includes comments for user code and uses HAL and BSP functions for timer and LCD control. The logic involves incrementing a counter, toggling a pin, and updating the LCD display with the current timer values.

```
241  */
242  void TIM1_UP_TIM16_IRQHandler(void)
243  {
244      /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 0 */
245
246      /* USER CODE END TIM1_UP_TIM16_IRQn 0 */
247      HAL_TIM_IRQHandler(&htim16);
248      /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 1 */
249      if(pause==0){
250          tenth++;
251          tenth = tenth%10;
252          uint8_t temp = tenth+48;
253          if(seccount==59&tenth==9){
254              mincount++;
255          }
256          if(tenth==0){
257              HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
258              seccount++;
259          }
260          seccount = seccount%60;
261
262          mincount = mincount%60;
263
264          BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 5);
265          temp = seccount%10+48;
266          BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 3); //one of the secc
267          temp = seccount/10+48;
268          BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_OFF, 2); //ten of the sec
269          temp = mincount%10+48;
270          BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 1); //ONE OF the min
271          temp = mincount/10+48;
272          BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_OFF, 0); //then of the min
273      }
274      /* USER CODE END TIM1_UP_TIM16_IRQn 1 */
```

Results





Conclusion

In this lab, we made our stopwatch. We use timers to generate external interrupts. Then the CPU reads the interrupts to update variables and update the LCD register with the updated variables. This should be the most basic application of the timer.

Appendix