Lab 4

Tues 14:15

Jack Landers Henry Fang

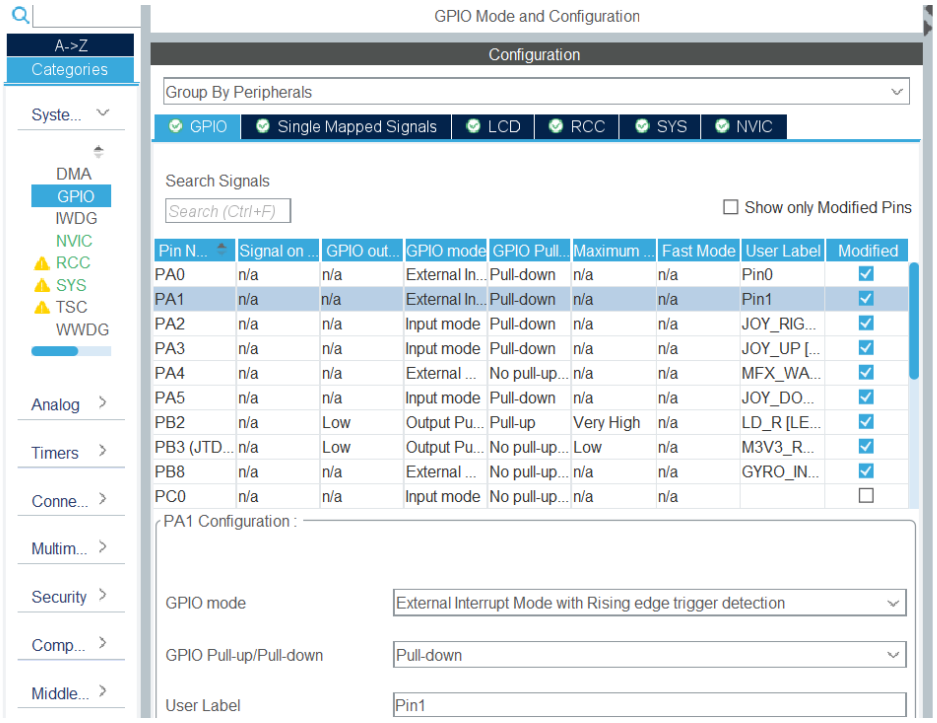
Introduction

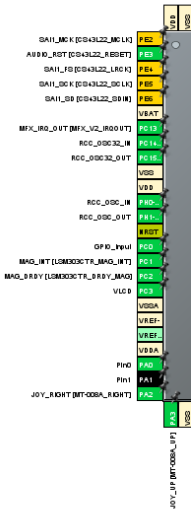
We will create an LCD timer in this lab.

Procedure

Step 1

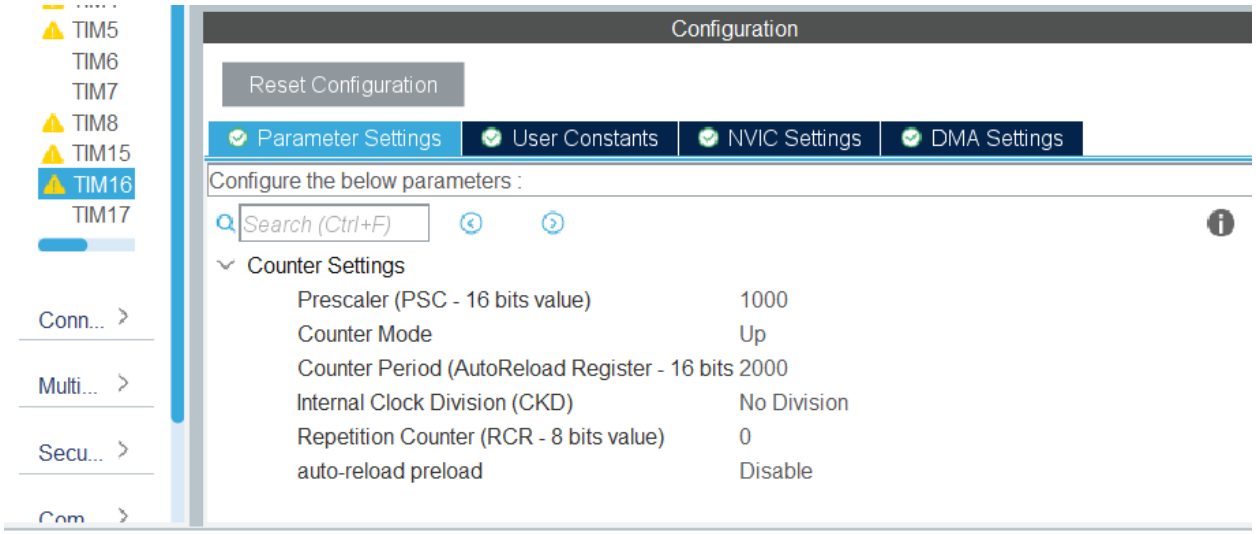
Create a new STM32CubeMX project and name it LCD Timer. Configure the GPIO pins to get it ready to send interrupts.





Step 2

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

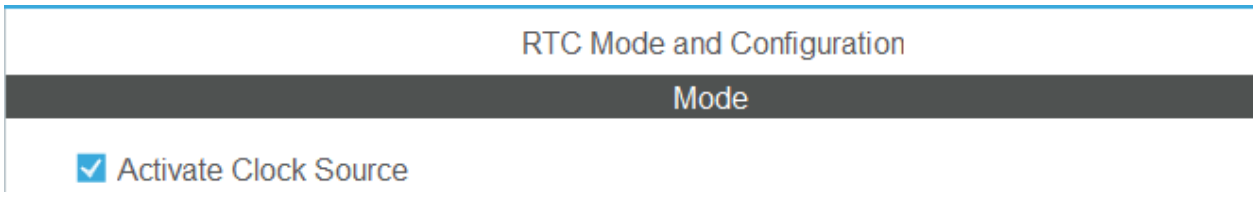


Step 3

Disable unused hardware.

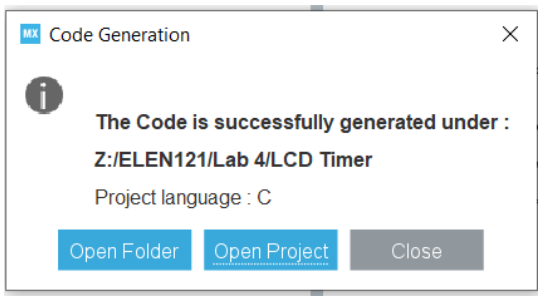
Step 4

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.

```c
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */

  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(Pin0_Pin);
  /* USER CODE BEGIN EXTI0_IRQn 1 */
  HAL_GPIO_TogglePin(LD_R_GPIO_Port, LD_R_Pin);
  pause++;
  pause%=2;
  /* USER CODE END EXTI0_IRQn 1 */
}

/**
  * @brief This function handles EXTI line1 interrupt.
  */
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(Pin1_Pin);
  /* USER CODE BEGIN EXTI1_IRQn 1 */
  HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
  seccount = 0;
  mincount = 0;
  tenth = 0;
  /* USER CODE END EXTI1_IRQn 1 */
}
```

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.

main.c | stm32l4xx_it.c | **display.c** | display.h

```c
#include "main.h"
#include "stm32l476g_discovery.h"
#include "stm32l476g_discovery_glass_lcd.h"
volatile int seccount = 0;
volatile int mincount = 0;
volatile int tenth = 0;
volatile int pause = 0;

void display_test(void) {
  uint8_t c = 'X';
  void BSP_LCD_GLASS_Clear(void);
  BSP_LCD_GLASS_DisplayChar(&c, POINT_OFF, DOUBLEPOINT_ON, 0);
}
```

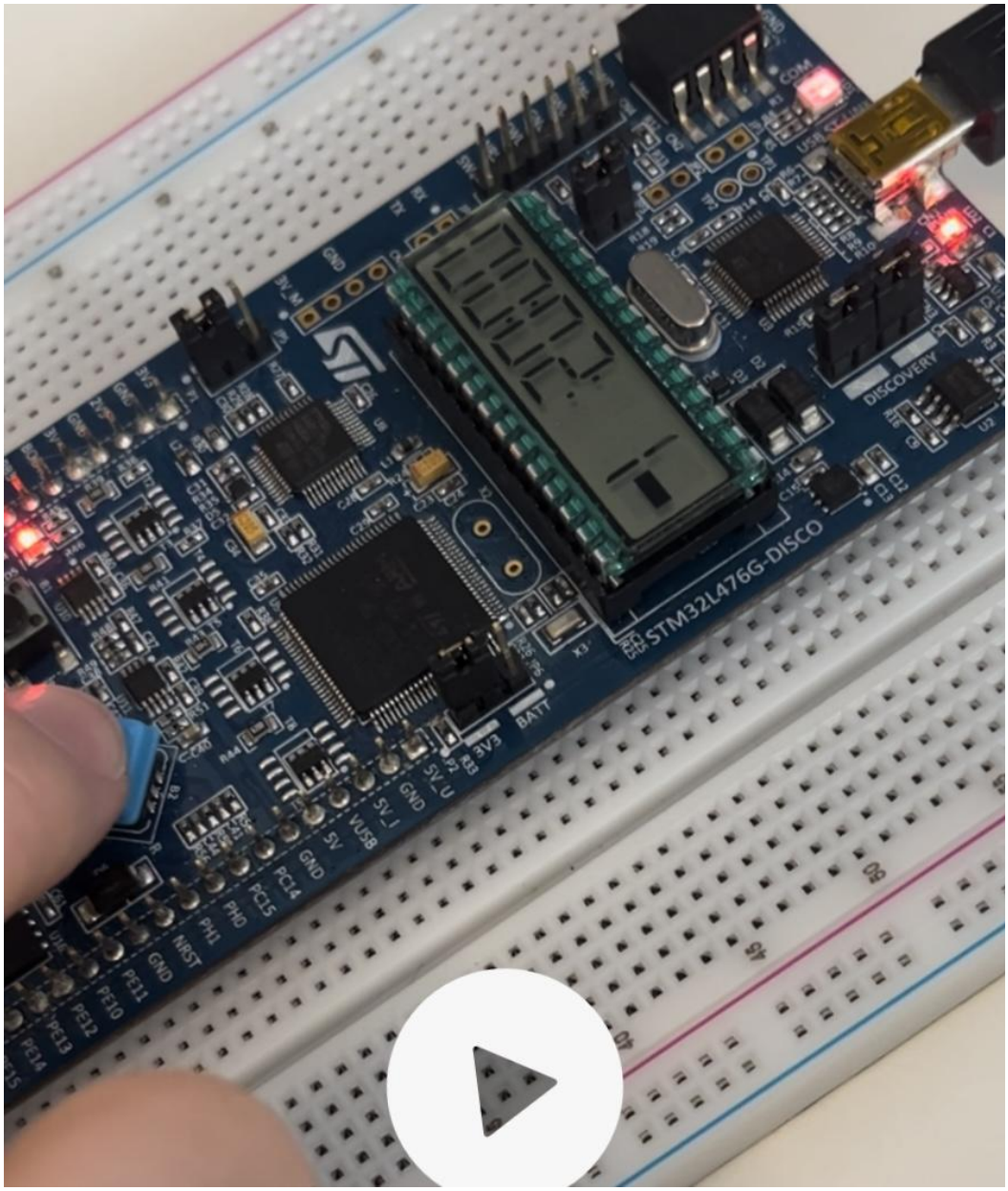
Step 8

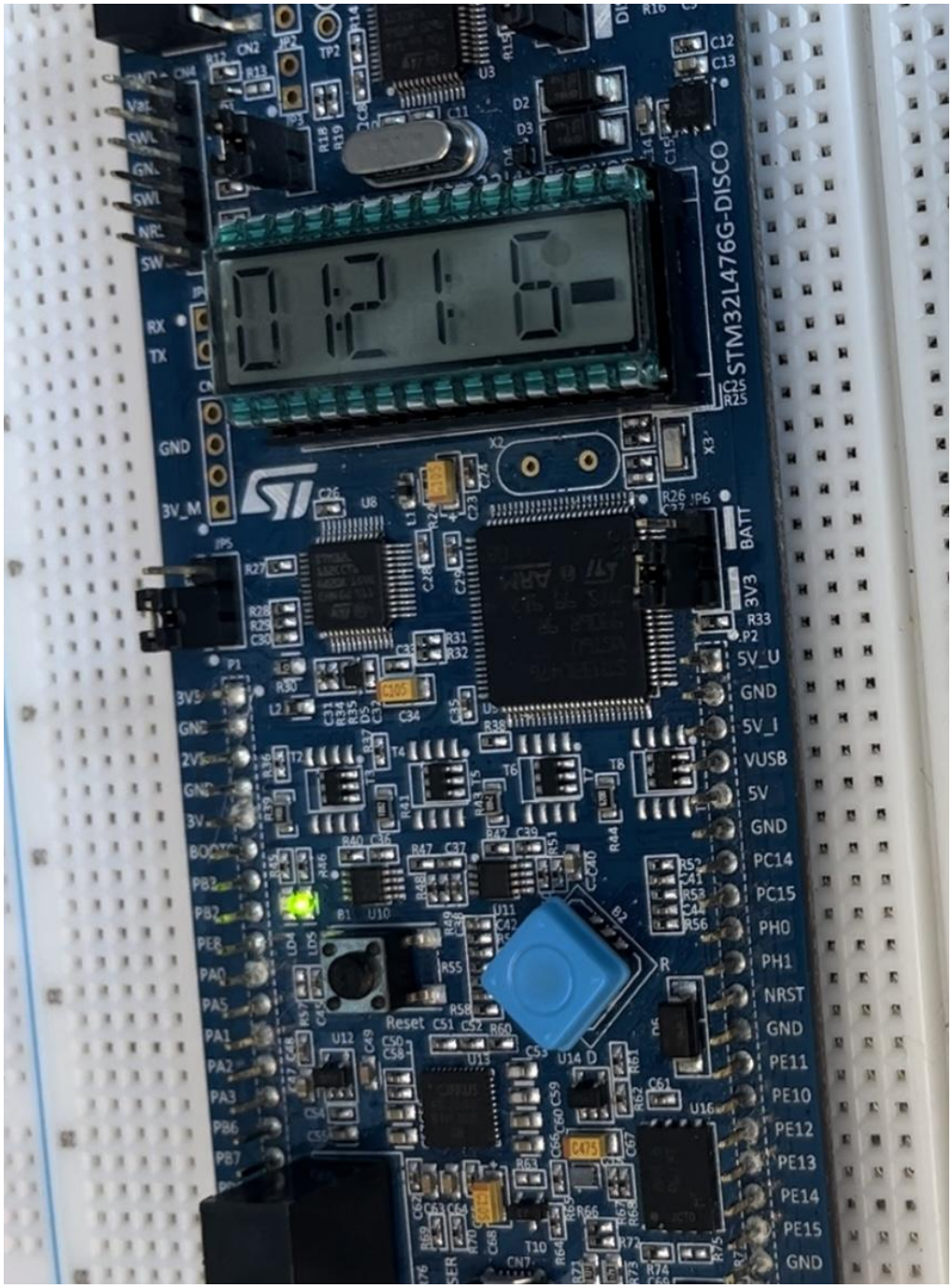
Finish the project.
Connect the interrupts with an algorithm that writes the current numerical time to the LCD Glass.

main.c | stm32l4xx_it.c | display.c | display.h

```c
  */
void TIM1_UP_TIM16_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 0 */

  /* USER CODE END TIM1_UP_TIM16_IRQn 0 */
  HAL_TIM_IRQHandler(&htim16);
  /* USER CODE BEGIN TIM1_UP_TIM16_IRQn 1 */
  if(pause==0){
    tenth ++;
    tenth = tenth%10;
    uint8_t temp = tenth+48;
    if(seccount==59&&tenth==9){
      mincount++;
    }
    if(tenth==0){
      HAL_GPIO_TogglePin(LD_G_GPIO_Port, LD_G_Pin);
      seccount++;
    }
    seccount = seccount%60;

    mincount = mincount%60;

    BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 5);
    temp = seccount%10+48;
    BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 3);//one of the secc
    temp = seccount/10+48;
    BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_OFF, 2);//ten of the sec
    temp = mincount%10+48;
    BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_ON, 1);//ONE OF the min
    temp = mincount/10+48;
    BSP_LCD_GLASS_DisplayChar(&temp, POINT_OFF, DOUBLEPOINT_OFF, 0);//then of the min
  }
  /* USER CODE END TIM1_UP_TIM16_IRQn 1 */
```

Results


STM32L476G-DISCO
DISCOVERY
BATT
3V3
5V_U
GND
5V_I
VUSB
5V
GND
PC14
PC15
PH0
PH1
NRST
GND
PE11
PE10
PE12
PC13
PC14



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