ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

VI XỬ LÝ - VI ĐIỀU KHIỂN

Lab 2

Timer Interrupt and LED Scanning

GVHD: **Huỳnh Phúc Nghị**
Thành viên : Đỗ Xuân Thơ 1713365

TP. HỒ CHÍ MINH, THÁNG 09/2021

Mục lục

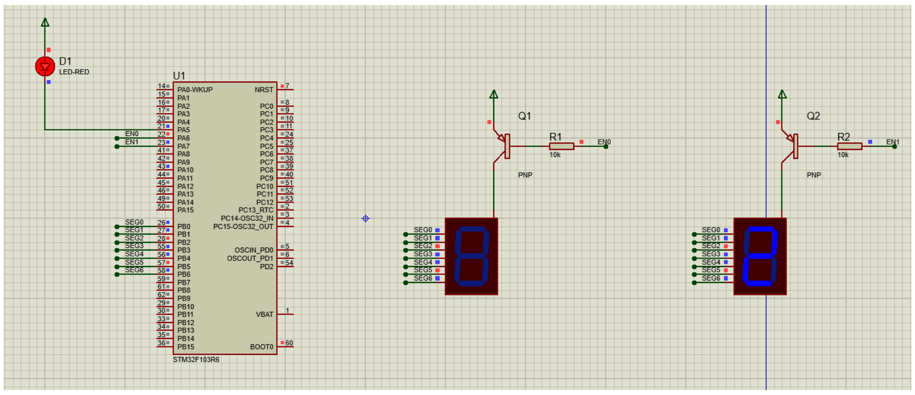
1 Execises 1

The first exercise show how to interface for multiple seven segment LEDs to STM32F103C6 micro-controller (MCU). Seven segment displays are common anode type, meaning that the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins.
In order to save the resource of the MCU, individual cathode pins from all the seven segment LEDs are connected together, and connect to 7 pins of the MCU. These pins are popular known as the signal pins. Meanwhile, the anode pin of each seven segment LEDs are controlled under a power enabling circuit, for instance, an PNP transistor. At a given time, only one seven segment LED is turned on. However, if the delay is small enough, it seems that all LEDs are enabling.

1.1 Source code

```
int counter = 50;
FlagStatus is_7segment_1 = SET;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
   counter -- ;
   if(counter <= 0){
      counter = 50;
      is_7segment_1 = !is_7segment_1;
      if(is_7segment_1){
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
         display7SEG(1);
      }
      else{
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
         display7SEG(2);
      }
   }
}
```

1.2 Schematic



What is the frequency of the scanning process? : 50 timer interrupt for context switch. Cycle time is 100 timer interrupt = 1s -> Frequency = 1.

2 Execises 2

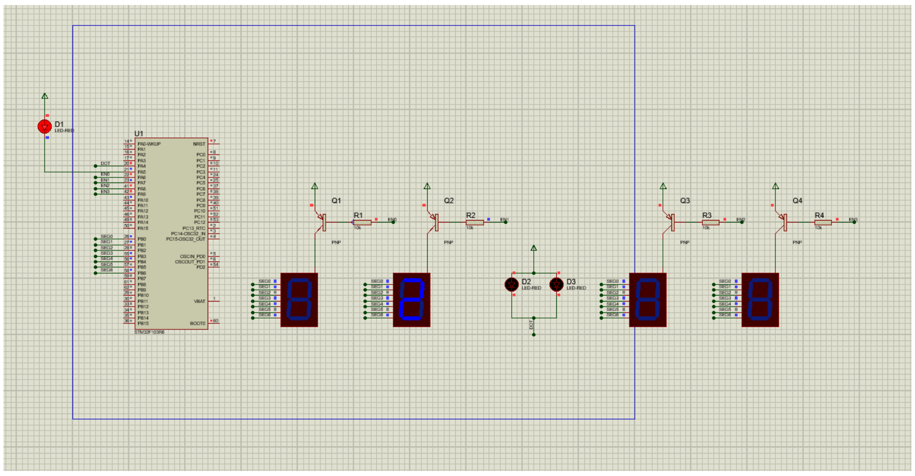
Blink the two LEDs every second. Meanwhile, number 3 is displayed on the third seven segment and number 0 is displayed on the last one (to present 12 hour and a half). The switching time for each seven segment LED is also a half of second (500ms). Implement your code in the timer interrupt function.

2.1 Source code

```c
int counter = 50;
int counter_2 = 100;
uint8_t led_index = 0;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
   counter -- ;
   counter_2 --;
   if(counter <= 0){
      counter = 50;
      switch (led_index) {
         case 0:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(1);
            break;
         case 1:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(2);
            break;
         case 2:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(3);
            break;
         case 3:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, RESET);
            display7SEG(0);
            break;
         default:
            break;
      }
      led_index = (led_index+1)%4;
   }
   if(counter_2<=0){
      counter_2 = 100;
```

```
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
  }
}
```

2.2 Schematic



What is the frequency of the scanning process? : 50 timer interrupt for context switch. Cycle time is 200 timer interrupt = 2s -> Frequency = 0.5.

3 Execises 3

Implement a function named update7SEG(int index).

3.1 Source code

```
const int MAX_LED = 4;
int index_led = 0;
int led_buffer[4] = {1,2,3,4};

void update7SEG(int index){
   switch (index) {
      case 0:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
         break;
      case 1:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
         break;
      case 2:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
         break;
      case 3:
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, RESET);

         break;
      default:
         break;
   }
   display7SEG(led_buffer[index]);
}


int counter = 50;
uint8_t led_index = 0;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
   counter --;
   if(counter<=0){
      counter = 50;
      update7SEG(led_index);
      led_index = (led_index+1)%MAX_LED;
```

```
    }
}
```

4 Execises 4

Change the period of invoking update7SEG function in order to set the frequency of 4 seven segment LEDs to 1Hz. The DOT is still blinking every second.

4.1 Source code

```
int counter = 100;
int counter_2 = 100;
uint8_t led_index = 0;
void HAL_TIM_PeriodElapsedCallback ( TIM_HandleTypeDef * htim ){
   counter -- ;
   counter_2 --;
   if(counter <= 0){
      counter = 100;
      switch (led_index) {
         case 0:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(1);
            break;
         case 1:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(2);
            break;
         case 2:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, RESET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, SET);
            display7SEG(3);
            break;
         case 3:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, SET);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, RESET);
            display7SEG(0);
            break;
         default:
            break;
      }
      led_index = (led_index+1)%4;
   }
   if(counter_2<=0){
      counter_2 = 100;
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
   }
}
```

5 Execises 5

Implement a digital clock with hour and minute information displayed by 2 seven segment LEDs.

5.1 Source code

```c
void updateClockBuffer(){
   led_buffer[0] = (int)(hour/10);
   led_buffer[1] = hour%10;
   led_buffer[2] = (int)(minute/10);
   led_buffer[3] = minute%10;
}
```

6 Execises 6

The main target from this exercise to reduce the complexity (or reduce code processing) in the timer interrupt. The time consumed in the interrupt can lead to the nested interrupt issue, which can crash the whole system. A simple solution can disable the timer whenever the interrupt occurs, the enable it again. However, the real-time processing is not guaranteed anymore. In this exercise, a software timer is created and its counter is count down every timer interrupt is raised (every 10ms). By using this timer, the HAL_Delay(1000) in the main function is removed. In a MCU system, non-blocking delay is better than blocking delay. The details to create a software timer are presented bellow. The source code is added to your current program, do not delete the source code you have on Exercise 5.

- **Report 1:** - If in line 1 of the code is miss, so LED_RED never be toggled. Because in time_run, It will check timer0_counter > 0 but we dont set value different from 0 to timer0_counter. So timer0_flag will never be set to 1.
- **Report 2:** - If in line 1 is setTimer0(1), then timer0_counter is 1. After that the first interrupt execute, It will check line "If (timer0_counter > 0) -> It's true 1 > 0" then "timer0_counter –" -> timer0_counter = 0; then check "timer0_counter == 0" -> It's true -> Set timer0_flag = 1; So then, In While(1) loop in Main function, LED_RED will be toggled. Then setTimer0(2000) will reset timer0_counter and timer0_flag.
- **Report 3:** - Timer0_counter will be changed from 0 -> 1 -> 10. It mean never toggle or 1 -> 10 interrupt time   10 -> 100ms to the first LED_RED toggle.

7 Execises 7

Upgrade the source code in Exercise 5 (update values for hour, minute and second) by using the software timer and remove the HAL_Delay function at the end. Moreover, the DOT (connected to PA4) of the digital clock is also moved to main function.

7.1 Source code

```
setTimer0(1000);
while (1)
{
  /* USER CODE END WHILE */
   if(timer0_flag){
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
      second ++;
      if(second>=60){
         second = 0;
         minute ++;
      }
      if(minute>=60){
         minute =0;
         hour++;
      }
      if(hour>=24){
         hour = 0;
      }
      updateClockBuffer();
      setTimer0(1000);
   }

  /* USER CODE BEGIN 3 */
}
```

8 Execises 8

Move also the update7SEG() function from the interrupt timer to the main. Finally, the timer interrupt only used to handle software timers. All processing (or complex computations) is move to an infinite loop on the main function, optimizing the complexity of the interrupt handler function.

8.1 Source code

```
  setTimer0(1000);
  setTimer1(500);
  while (1)
  {
    /* USER CODE END WHILE */
     if(timer0_flag){
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
        second ++;
        if(second>=60){
           second = 0;
           minute ++;
        }
        if(minute>=60){
           minute =0;
           hour++;
        }
        if(hour>=24){
           hour = 0;
        }
        updateClockBuffer();
        setTimer0(1000);
     }
     if(timer1_flag){
        update7SEG(led_index);
        led_index = (led_index+1)%4;
        setTimer1(500);
     }

    /* USER CODE BEGIN 3 */
  }
  void setTimer1(int duration){
   timer1_counter = duration/TIMER_CYCLE;
   timer1_flag = 0;
}
```

9 Execises 9

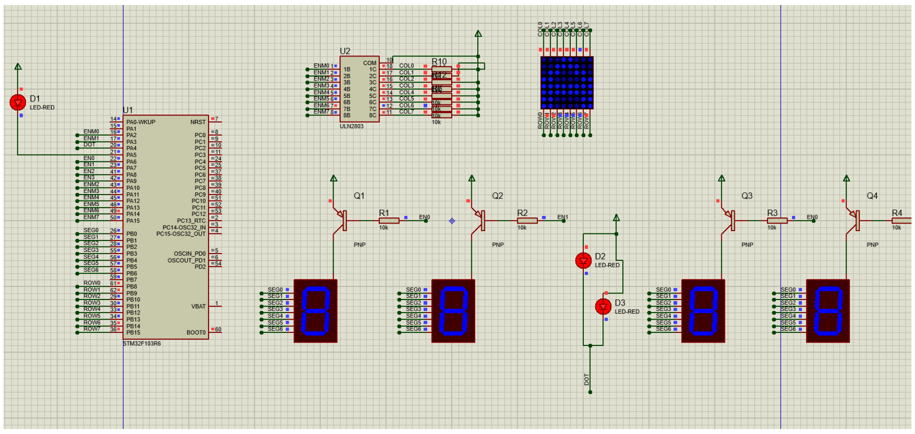
Implement a function named clearNumberOnClock(int num). The input for this function is from 0 to 11 and an appropriate LED is turn off.

9.1 Source code

```
void clear_col(int index){
   HAL_GPIO_WritePin(GPIOA,
        col_mask[0]|col_mask[1]|col_mask[2]|col_mask[3]|col_mask[4]|col_mask[5]|col_mask[6]|col_mask[7],
        GPIO_PIN_RESET);
}
void display_col(int col_index){
   HAL_GPIO_WritePin(GPIOA, col_mask[col_index], GPIO_PIN_SET);
   for (int var = 0; var < 8; ++var) {
      if((0x01)&(maxtrix_buffer[col_index]>>var)){
         HAL_GPIO_WritePin(GPIOB, row_mask[var], GPIO_PIN_SET);
      }
      else{
         HAL_GPIO_WritePin(GPIOB, row_mask[var], GPIO_PIN_RESET);
      }
   }
}
// Update A character
void updateLEDMatrix(int index){
   /*
    * A character
    *            * *
    *          * * * *
    *        * *     * *
    *      * *         * *
    *      * * * * * * * *
    *      * * * * * * * *
    *      * *         * *
    *      * *         * *
    */
   // matrix_buffer[] = { 0x18,0x3C,0x66,0xC3,0xFF,0xFF,0xC3,0xC3}
   // clear all matrix
   clear_col(index);
   switch (index) {
      case 0:
         maxtrix_buffer[0] = 0x18;
         break;
      case 1:
         maxtrix_buffer[1] = 0x3C;
         break;
      case 2:
         maxtrix_buffer[2] = 0x66;
         break;
      case 3:
         maxtrix_buffer[3] = 0xC3;
         break;
      case 4:
         maxtrix_buffer[4] = 0xFF;
         break;
      case 5:
         maxtrix_buffer[5] = 0xFF;
         break;
      case 6:
         maxtrix_buffer[6] = 0xC3;
         break;
      case 7:
         maxtrix_buffer[7] = 0xC3;
         break;
      default:
         break;
   }
   display_col(index);
}
```

```
  set_Timer0(10);
  while(1){
     if(timer0_flag){
        //If all columns are set, delay 5s
        updateLEDMatrix(index_led_matrix);
        index_led_matrix = (index_led_matrix+1)%MAX_LED_MATRIX;
        set_Timer0(10);
     }
  }
```

9.2 Schematic



10 Execises 10

Create an animation on LED matrix, for example, the character is shifted to the left. **Brief**: Character A will shift to left.

10.1 Source code

```
  set_Timer0(10);
  set_Timer1(200);
  while(1){
     if(timer0_flag){
        //If all columns are set, delay 5s
        updateLEDMatrix(index_led_matrix);
        index_led_matrix = (index_led_matrix+1)%MAX_LED_MATRIX;
        set_Timer0(10);
     }
     if(timer1_flag){
        shift_index = (shift_index +1)%MAX_LED_MATRIX;
        set_Timer1(200);
     }
  }



void updateLEDMatrix(int index){
   /*
    * A character
    *            * *
    *          * * * *
    *        * *    * *
    *      * *        * *
    *      * * * * * * * *
    *      * * * * * * * *
    *      * *        * *
    *      * *        * *
    */
   // matrix_buffer[] = { 0x18,0x3C,0x66,0xC3,0xFF,0xFF,0xC3,0xC3}
   // clear all matrix
   clear_col(index);
   switch (index) {
      case 0:
         maxtrix_buffer[0] = 0x18>>shift_index;
         break;
      case 1:
         maxtrix_buffer[1] = 0x3C>>shift_index;
         break;
      case 2:
         maxtrix_buffer[2] = 0x66>>shift_index;
         break;
      case 3:
         maxtrix_buffer[3] = 0xC3>>shift_index;
         break;
      case 4:
         maxtrix_buffer[4] = 0xFF>>shift_index;
         break;
      case 5:
         maxtrix_buffer[5] = 0xFF>>shift_index;
         break;
      case 6:
         maxtrix_buffer[6] = 0xC3>>shift_index;
         break;
      case 7:
         maxtrix_buffer[7] = 0xC3>>shift_index;
         break;
      default:
         break;
   }
   display_col(index);
}
```

10.2 Schematic