Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1 по Проектированию вычислительных систем

Вариант 1

Роман Юнусов

Алексей Лапин
Группа P34102

Преподаватель Василий Пинкевич

Санкт-Петербург 2024

Оглавление

Задание 3

Основные этапы вычисления ..... **Ошибка! Закладка не определена.**

Вывод..... **Ошибка! Закладка не определена.**

Документация..... **Ошибка! Закладка не определена.**

Задание

Сымитировать работу светофора пешеходного перехода. Светофор циклически
переключает цвета в следующем порядке (порядок условный, соответствие реальному
светофору не соблюдается): красный, зелёный, зелёный мигающий, жёлтый, снова красный и т. д. По умолчанию период горения красного в четыре раза больше периода горения зеленого. Если во время горения зеленого мигающего, желтого или красного нажимается кнопка, светофор запоминает необходимость скорейшего переключения на зелёный. После нажатия кнопки общий цикл работы светофора не нарушается, но период горения красного должен быть сокращен до ¼ своего обычного периода. Если кнопка нажата во время горения красного, когда он уже горит более ¼ периода, то сразу происходит переключение на зеленый.

А также дополнительное задание

Модифицировать программу: отключить в графическом конфигураторе настройку портов GPIO. Разработать собственные функции для инициализации и использования портов GPIO (можно с использованием объявленных в стандартной библиотеке структур и констант). Заменить в программе использование библиотечных функций работы с GPIO на вызов собственных реализаций.

Решение

Для реализации первой части задания был сделан конечный автомат переключающий. Во время прихода в стейт мы определяем время через которое мы должны закончить, постоянно сравнивая нынешнее время с эталоном и проверяя нажата ли кнопка. Если кнопка нажата, то мы изменяем время действия красного сигнала, если же нажата на красном сигнале то ещё изменяем эталонное время выхода.

Во время проверки времени в мигающем зеленом стейте ,собственно происходит сам процесс мигания, мы взяли постоянные 8 изменений состояний за время.

```
int button_pressed = 0;
int start_time = 0;
int waiting_time = 0;
int cnt = 0;
```

```
    switch (*light) {
        case RED:
             reset_LEDS();
             set_LED(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
             start_time = HAL_GetTick();
             waiting_time = start_time + (*red_t);
             while(HAL_GetTick() < waiting_time) {
                    button_pressed = read_button(&cnt);
                    if(button_pressed){
                           (*red_t) = (*other_t);
                           waiting_time = start_time + (*red_t);
                    }
             }
             *light = GREEN;
            break;
        case GREEN:
             reset_LEDS();
             set_LED(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
             start_time = HAL_GetTick();
             waiting_time = start_time + (*other_t);
             while(HAL_GetTick() < waiting_time){
                    button_pressed = read_button(&cnt);
                    if(button_pressed){
                           (*red_t) = (*other_t);
                    }
             }
             *light = GREEN_BLINK;
            break;
        case GREEN_BLINK:
             reset_LEDS();
             set_LED(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
             start_time = HAL_GetTick();
             int prev = start_time;
             waiting_time = start_time + (*other_t);
             while(HAL_GetTick() < waiting_time) {
                    button_pressed = read_button(&cnt);
                    if(button_pressed){
                           (*red_t) = (*other_t);
                    }
                    if(HAL_GetTick() - prev >= (*other_t) / 8){
                           prev = HAL_GetTick();
                           //inline toggle
                           uint32_t odr = GPIOD->ODR;
                           GPIOD->BSRR = ((odr & GPIO_PIN_13) << 16U) | (~odr &
GPIO_PIN_13);
                    }
             }
            *light = YELLOW;
            break;
        case YELLOW:
             reset_LEDS();
             set_LED(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
             start_time = HAL_GetTick();
             waiting_time = start_time + (*other_t);
             while(HAL_GetTick() < waiting_time){
                    button_pressed = read_button(&cnt);
                    if(button_pressed){
                           (*red_t) = (*other_t);
                    }
             }
            *light = RED;
```

```
            break;
        default:
             reset_LEDS();
             *light = RED;
            break;
    }
```

Отдельно разберем функцию чтения кнопки. Тут важны следующие моменты, кнопка считается нажатой в состоянии 0, а также количество тактов антидрезбега. При стандартном времени дребезга в 5-10мс, то при частоте 16 мега герц, нам нужно примерно 80-160 тактов, взяли 200 для надежности.

```
int read_button(int* cnt){
      GPIO_PinState res = read_PIN(GPIOC, GPIO_PIN_15);
      if(res == 0){
            (*cnt) = (*cnt) + 1;
      }
      else
        (*cnt) = 0;
      if((*cnt) >= 200){
            (*cnt) = 0;
            return 1;
      }
      return 0;
}
```

В рамках выполнения последующего задания были реализованы следующие функции

Инициализация пинов, тут относительно сгенерированного варианта были отключены ненужные в нашем решении пины GPIOA и GPIOB.

```
void my_GPIO_INIT(){
        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_GPIOC_CLK_ENABLE();
        __HAL_RCC_GPIOD_CLK_ENABLE();

        /*Configure GPIO pin Output Level */
        set_LED(GPIOD, GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);

        /*Configure GPIO pin : PC15 */
        GPIO_InitStruct.Pin = GPIO_PIN_15;
        GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

        /*Configure GPIO pins : PD13 PD14 PD15 */
        GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
        GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        // судя по справке и нашей частоте нам нужна средняя скорость, а не low как
было
```

```
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
        HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}
```

Были также реализованы функции чтения и записи в PIN. В этих функциях относительно встроенных я избавился от асертов(мой код гарантированно даёт нужные значения). Также немного изменил if для чтения пина, убрав лишнюю конвертацию enuma

```
void set_LED(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState){
        if(PinState != GPIO_PIN_RESET)
        {
          GPIOx->BSRR = GPIO_Pin;
        }
        else
        {
          GPIOx->BSRR = (uint32_t)GPIO_Pin << 16U;
        }
};

GPIO_PinState read_PIN(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin){
      GPIO_PinState bitstatus;
      if((GPIOx->IDR & GPIO_Pin))
        bitstatus = GPIO_PIN_SET;
      else
        bitstatus = GPIO_PIN_RESET;
      return bitstatus;
}
```