Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1
“Интерфейсы ввода-вывода общего назначения (GPIO)”

Дисциплина: Проектирование вычислительных систем
Вариант 4

Выполнили: студенты группы P34131

Бусыгин Дмитрий Алексеевич

Гиря Максим Дмитриевич

Преподаватель:

Пинкевич Василий Юрьевич

Санкт-Петербург
2024

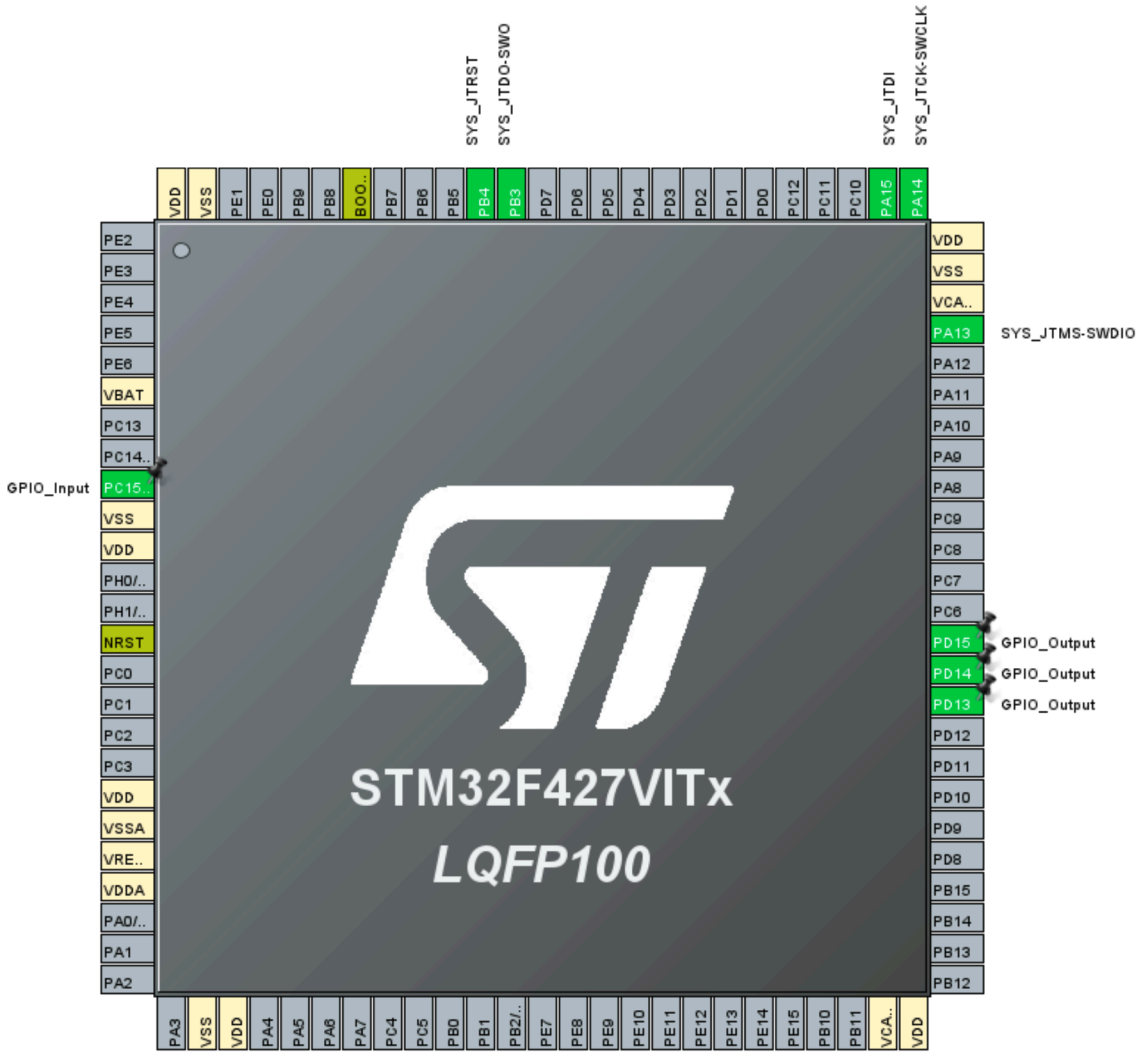
Задание:

- Разработать и реализовать драйверы управления светодиодными индикаторами и чтения состояния кнопки стенда SDK-1.1M (индикаторы и кнопка расположены на боковой панели стенда).
- Написать программу с использованием разработанных драйверов в соответствии с вариантом задания.

Вариант:

- Реализовать двоичный двухразрядный счетчик на светодиодах с возможностью вычитания (использовать зеленый светодиод и один из двух цветов двухцветного).
- Быстрое нажатие кнопки должно прибавлять единицу к отображаемому на светодиодах двоичному числу.
- По переполнению счетчика должна отображаться простая анимация: мигание обоими светодиодами, затем количество миганий зеленым светодиодом, равное количеству переполнений с момента перезагрузки микроконтроллера.
- Долгое нажатие кнопки должно вычитать единицу из отображаемого на светодиодах двоичного числа.
- Если происходит вычитание из нуля, количество переполнений уменьшается на единицу, и отображается анимация, аналогичная анимации при переполнении.

Используемые контакты:



Описание контактов:

PC15 - перехватывает нажатие кнопки (настроен на GPIO_Input)

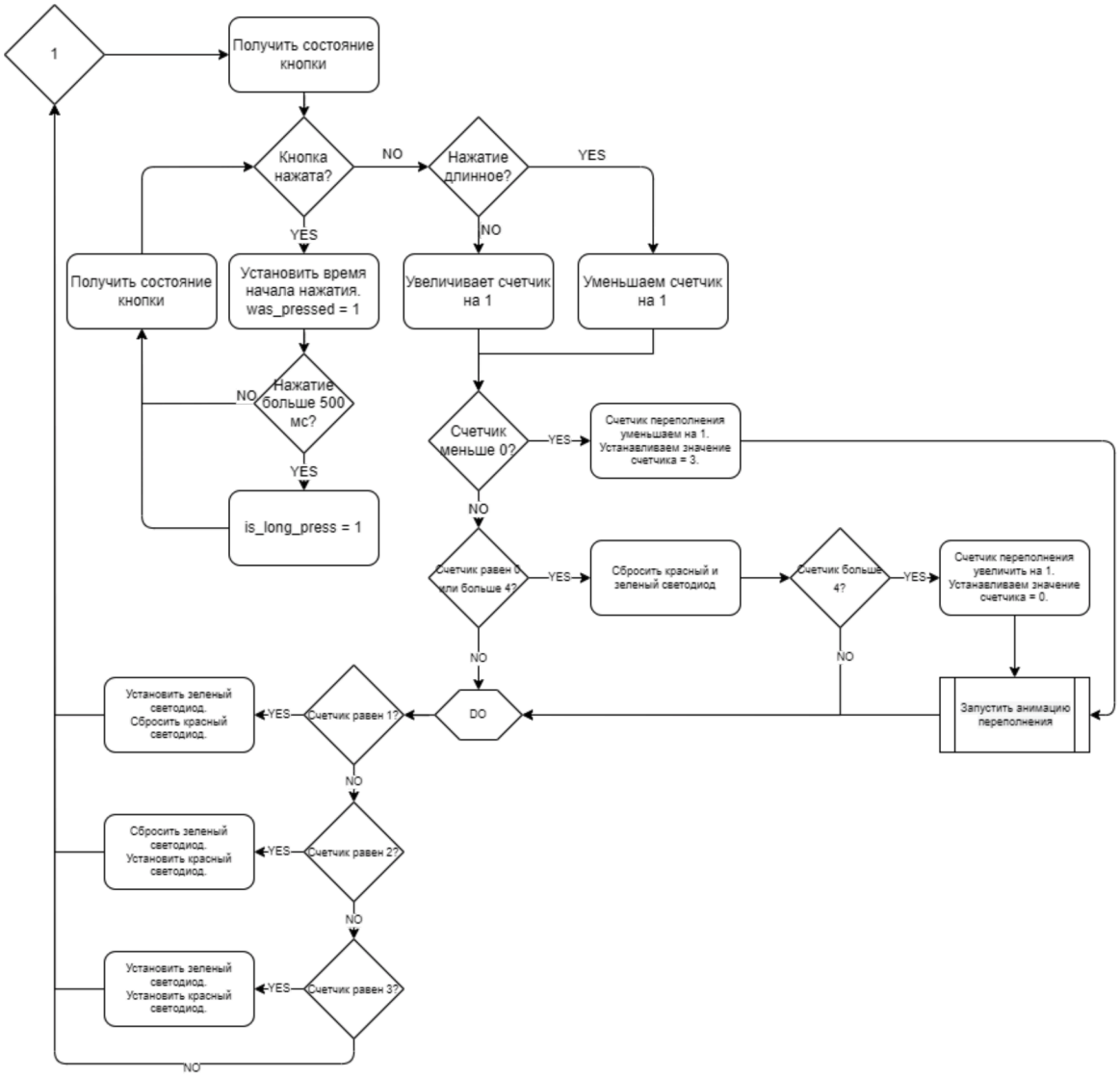
PD13 - управляет зеленым светодиодом (настроен на GPIO_Output)

PD14 - управляет желтым светодиодом (настроен на GPIO_Output)

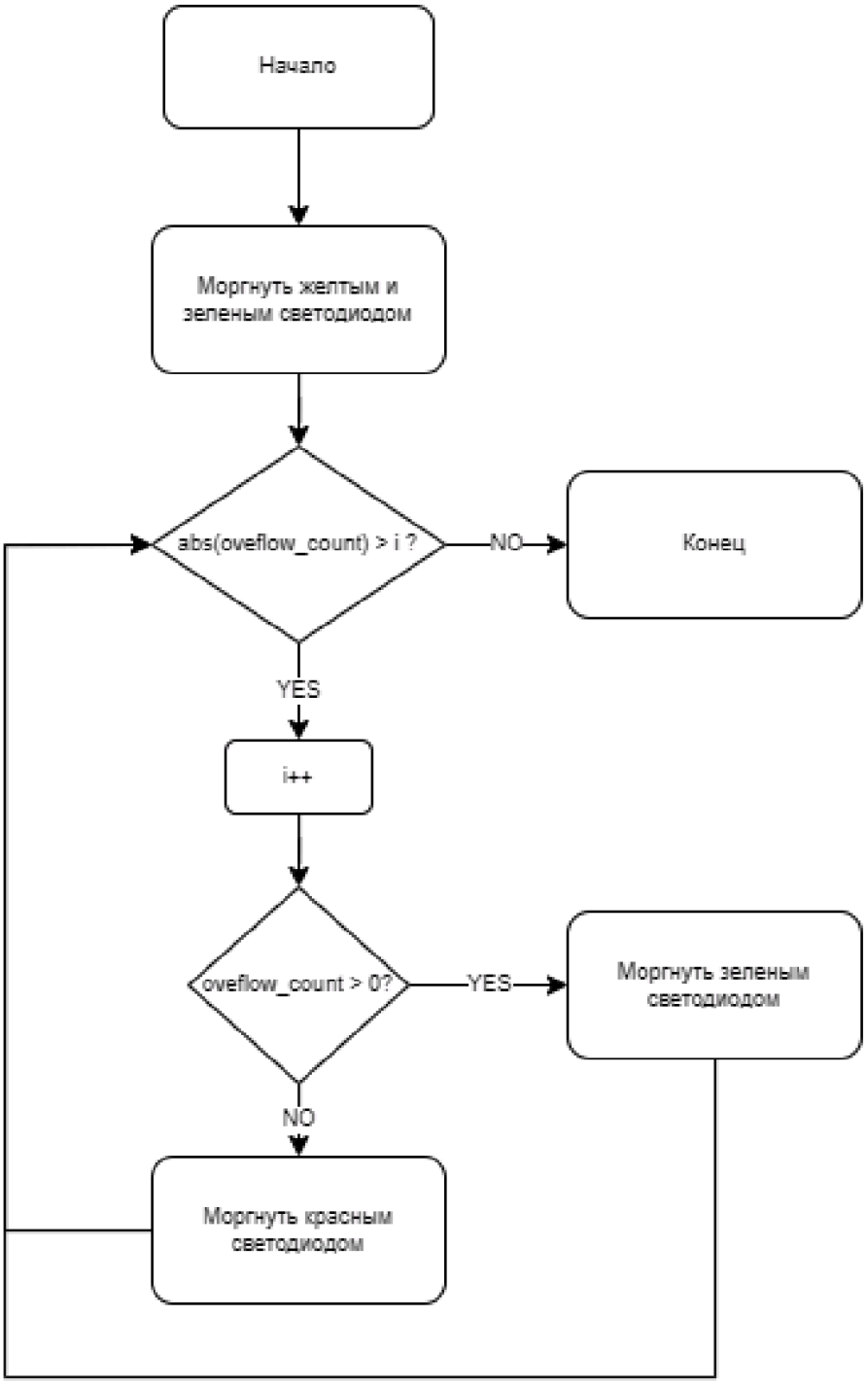
PD15 - управляет красным светодиодом (настроен на GPIO_Output)

Используемые контакты:

Основная схема алгоритма



Подпрограмма “анимация переполнения”



Описание работы алгоритма:

1. **Проверка состояния кнопки**:
   Алгоритм циклично проверяет, нажата кнопка или нет.
2. **Фиксация нажатия**:
   Если кнопка нажата:
   - Фиксируем факт нажатия.
   - Измеряем продолжительность нажатия.
3. **Обработка продолжительности нажатия**:
   - Если нажатие было **долгим**, сохраняем это состояние.
   - Если нажатие было **коротким**, фиксируем его.
4. **Действия при отпускании кнопки**:
   После отпускания кнопки:
   - Если нажатие было долгим, уменьшаем счётчик.
   - Если нажатие было коротким, увеличиваем счётчик.
5. **Ограничения счётчика**:
   Счётчик может принимать значения от 0 до 3.
   - При значениях **в пределах** диапазона (0, 1, 2, 3):
     Отображается число с помощью лампочек.
   - При значениях **вне диапазона** (переполнение):
     - При переполнении в большую сторону: запускается анимация с подсветкой **зелёного светодиода**.
     - При переполнении в меньшую сторону: запускается анимация с подсветкой **красного светодиода**.
6. **Анимация переполнения**:
   Анимация показывает, насколько счётчик вышел за пределы диапазона (в большую или меньшую сторону).

Код драйвера для кнопок:

```
#include "button_driver.h"
#include "main.h"
#include "stdbool.h"
#include "stdio.h"
_Bool Button_WasPressed(void) {
    _Bool button_state = !My_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
    if (button_state) {
              HAL_Delay(30);
              _Bool new_button_state = !My_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
              if ((new_button_state == button_state) && button_state) {
               return 1;
              }
    }
    return 0;
}
```

Код драйвера для светодиодов:

```
#include "led_driver.h"
#include "main.h"
void Light_Up_Led(Led_Type led) {
    switch (led)
    {
        case YELLOW_LED:
                    My_GPIO_SetPin(GPIOD, GPIO_PIN_14);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
            break;
        case RED_LED:
                    My_GPIO_SetPin(GPIOD, GPIO_PIN_15);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_14);
            break;
        case GREEN_LED:
                    My_GPIO_SetPin(GPIOD, GPIO_PIN_13);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_14);
                    My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
                    break;
        default:
            break;
    }
}
```

Код основной программы:

```c
#include "main.h"
#include "gpio.h"
#include "button_driver.h"
#include "led_driver.h"

void SystemClock_Config(void);

void My_GPIO_Init(void);
void My_GPIO_SetPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void My_GPIO_ResetPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
uint8_t My_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
/* Кастомная функция для инициализации GPIO */
void My_GPIO_Init(void)
{
    // Включаем тактирование портов GPIOC и GPIOD
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN | RCC_AHB1ENR_GPIODEN;
    // Настраиваем PC15 (кнопка) как вход
    GPIOC->MODER &= ~(GPIO_MODER_MODER15);  // Режим входа
    GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPDR15);  // Сбрасываем режим подтяжки
    GPIOC->PUPDR |= GPIO_PUPDR_PUPDR15_0; // Устанвливаем режим подтяжки наверх
    // Настраиваем PD13, PD14, PD15 (светодиоды) как выходы
    GPIOD->MODER &= ~(GPIO_MODER_MODER13 | GPIO_MODER_MODER14 | GPIO_MODER_MODER15);
    GPIOD->MODER |= (GPIO_MODER_MODER13_0 | GPIO_MODER_MODER14_0 | GPIO_MODER_MODER15_0); // Установка в
режим выхода
    GPIOD->OTYPER &= ~(GPIO_OTYPER_OT13 | GPIO_OTYPER_OT14 | GPIO_OTYPER_OT15); // Тип выхода - push-pull
    GPIOD->OSPEEDR &= ~(GPIO_OSPEEDER_OSPEEDR13 | GPIO_OSPEEDER_OSPEEDR14 | GPIO_OSPEEDER_OSPEEDR15); //
Низкая скорость
    GPIOD->PUPDR &= ~(GPIO_PUPDR_PUPDR13 | GPIO_PUPDR_PUPDR14 | GPIO_PUPDR_PUPDR15); // Без подтяжки
    // Устанавливаем начальное состояние светодиодов в 0
    GPIOD->BSRR = (GPIO_BSRR_BR13 | GPIO_BSRR_BR14 | GPIO_BSRR_BR15);
}
void My_GPIO_SetPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
      GPIOx->BSRR = GPIO_Pin;
}
void My_GPIO_ResetPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
      GPIOx->BSRR = (uint32_t)GPIO_Pin << 16U;
}
uint8_t My_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
      return (uint8_t)((GPIOx->IDR & GPIO_Pin) ? 1 : 0);
}
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  HAL_Init();
```

```
SystemClock_Config();

My_GPIO_Init();

int count = 0;
int overflow_count = 0;
_Bool was_pressed = 0;
_Bool is_long_press = 0;
uint32_t wait = 500;
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
	int start_time;
	int button_state = My_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
	was_pressed = 0;
	is_long_press = 0;
	if(!button_state){
		start_time = HAL_GetTick();
	}
	while (!button_state){
		was_pressed = 1;
		if (HAL_GetTick() - start_time >= wait){
			is_long_press = 1;
		}
		button_state = My_GPIO_ReadPin(GPIOC, GPIO_PIN_15);
	}
	if (was_pressed){
		!is_long_press ? count++ : count--;
	}
	if (count < 0){
		count = 3;
		overflow_count--;
		overflow_animation(overflow_count);
	}
	if (count == 0 || count >= 4){
		My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
		My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
		if (count >= 4){
			count = 0;
			overflow_count++;
			overflow_animation(overflow_count);
		}
	}
	if (count == 1){
		My_GPIO_SetPin(GPIOD, GPIO_PIN_13);
		My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
	}
	if (count == 2){
```

```
                My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
                My_GPIO_SetPin(GPIOD, GPIO_PIN_15);
        }
        if (count == 3){
                My_GPIO_SetPin(GPIOD, GPIO_PIN_13);
                My_GPIO_SetPin(GPIOD, GPIO_PIN_15);
        }
  }
  /* USER CODE END 3 */
}

/* USER CODE BEGIN 4 */
void overflow_animation(int overflow_count){
       My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
       My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
       HAL_Delay(250);
       My_GPIO_SetPin(GPIOD, GPIO_PIN_13);
       My_GPIO_SetPin(GPIOD, GPIO_PIN_14);
       HAL_Delay(250);
       My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
       My_GPIO_ResetPin(GPIOD, GPIO_PIN_14);
       HAL_Delay(500);
       for (int i = 0; i < abs(overflow_count); i++) {
              if (overflow_count > 0){
                     My_GPIO_SetPin(GPIOD, GPIO_PIN_13);
                     HAL_Delay(250);
                     My_GPIO_ResetPin(GPIOD, GPIO_PIN_13);
                     HAL_Delay(250);
              } else {
                     My_GPIO_SetPin(GPIOD, GPIO_PIN_15);
                     HAL_Delay(250);
                     My_GPIO_ResetPin(GPIOD, GPIO_PIN_15);
                     HAL_Delay(250);
              }
       }
}

#endif /* USE_FULL_ASSERT */
```

Вывод:

В ходе лабораторной работы мы познакомились с интерфейсами ввода/вывода общего назначения и реализовали “счетчика” с помощью двух светодиодов.