Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет ПИиКТ, кафедра ВТ

ОТЧЁТ

По лабораторной работе №2

“Последовательный интерфейс UART”

По предмету: Проектирование вычислительных систем

Вариант 1

Студенты:

Андрейченко Леонид Вадимович

Степанов Михаил Алексеевич

Группа P34301

Преподаватель:

Пинкевич Василий Юрьевич

Санкт-Петербург

2023

Цель работы

1. Изучить протокол передачи данных по интерфейсу UART.
2. Получить базовые знания об организации системы прерываний в микроконтроллерах на примере микроконтроллера STM32.
3. Изучить устройство и принципы работы контроллера интерфейса UART, получить навыки организации обмена данными по UART в режимах опроса и прерываний.

Задание

Разработать и реализовать два варианта драйверов UART для стенда SDK-1.1M: с использованием и без использования прерываний. Драйверы, использующие прерывания, должны обеспечивать работу в «неблокирующем» режиме (возврат из функции происходит сразу же, без ожидания окончания приема/отправки), а также буферизацию данных для исключения случайной потери данных. В драйвере, не использующем прерывания, функция приема данных также должна быть «неблокирующей», то есть она не должна зависать до приема данных (которые могут никогда не поступить). При использовании режима «без прерываний» прерывания от соответствующего блока UART должны быть запрещены.

Написать с использованием разработанных драйверов программу, которая выполняет
определенную вариантом задачу. Для всех вариантов должно быть реализовано два режима работы программы: с использованием и без использования прерываний. Каждый принимаемый стендом символ должен отсылаться обратно, чтобы он был выведен в консоли (так называемое «эхо»). Каждое новое сообщение от стенда должно выводиться с новой строки. Если вариант предусматривает работу с командами, то на каждую команду должен выводиться ответ, определенный в задании или «ОК», если ответ не требуется. Если введена команда, которая не поддерживается, должно быть выведено сообщение об этом.

Вариант

Доработать программу «светофор», добавив возможность отключения кнопки и задание величины тайм-аута (период, в течение которого горит красный). Должны обрабатываться следующие команды, посылаемые через UART:

- **?** – в ответ стенд должен прислать состояние, которое отображается в данный момент на светодиодах: red, yellow, green, blinking green, режим – mode 1 или mode 2 (см. далее), величину тайм-аута (сколько горит красный) – timeout …, и задействованы ли прерывания – символ I (interrupt) или P (polling);
- **set mode 1** или **set mode 2** – установить режим работы светофора, когда обрабатываются или игнорируются нажатия кнопки;
- **set timeout X** – установить тайм-аут (X – длина периода в секундах);
- **set interrupts on** или **set interrupts off** – включить или выключить прерывания.

Скорость обмена данными по UART: 57600 бит/с.

Организация программы

Основная программа представляет из себя цикл, в котором происходит обновление состояния устройств (uart, светодиодов, кнопки) в зависимости от времени прошедшего с момента последнего состояния устройства.

Функции обновления светодиодов и кнопки проверяют момент последнего обновления состояния и принимают решение об его изменении. Таким образом мы можем контролировать моргание светодиодами с непроизвольной задержкой, а также предотвращать дребезг кнопки.

Блок-Схема алгоритма (рисунок 1)

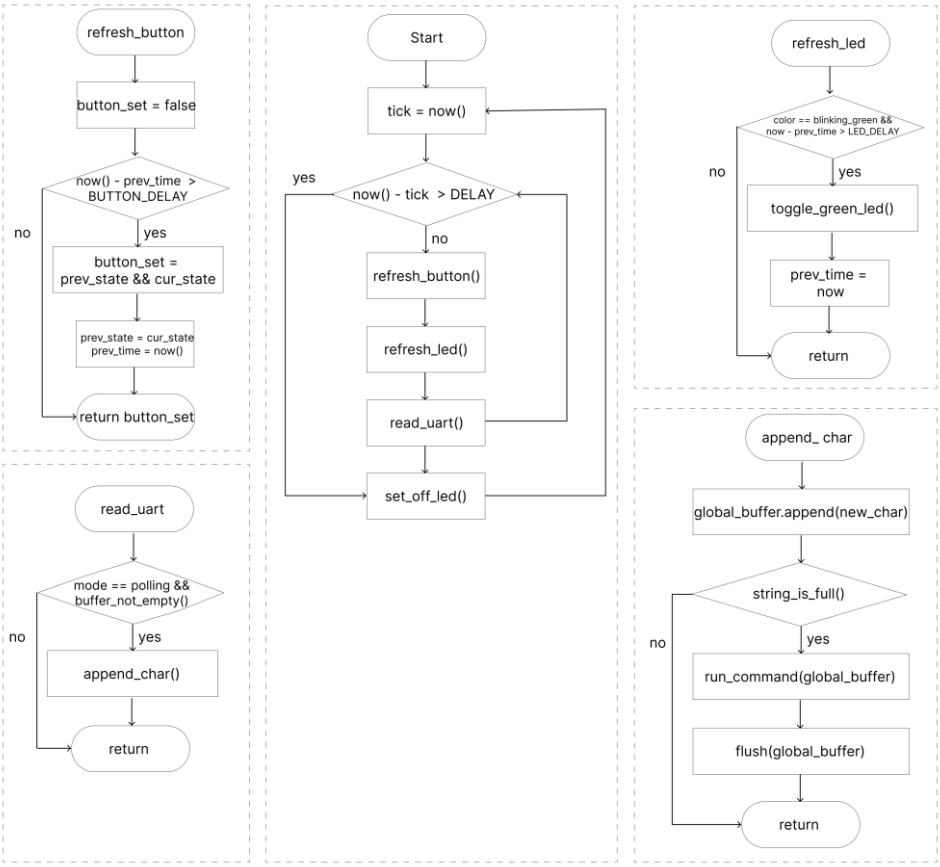

рисунок 1 (Блок-схема алгоритма)

Исходный код

```c
void send_uart(UART_HandleTypeDef *huart, char *s, enum InterruptState state, size_t len) {
    if (state == NonInterrupt)
        HAL_UART_Transmit(huart, (uint8_t *)s, len, UART_MAX_TIMEOUT);
    else
        HAL_UART_Transmit_IT(huart, (uint8_t *)s, len);
}
```

```
while (1)
{
    tick_loop = HAL_GetTick();
    tick_led = HAL_GetTick();
    set_wait(&wait);

    if (state == Green)
            reduced_red = FALSE;

    while((HAL_GetTick() - tick_loop) < wait)
    {
        set_wait(&wait);                                // update wait value if it was changed
        set_led(state, &tick_led);                      // update led state (color and blinking)
        read_uart(&huart6, interrupt_state);            // check for new char at UART port

        // check button and if it press remember to reduced red timeout
        if (button_working && !reduced_red && state != Green && !check_button())
                reduced_red = TRUE;
    }

    reset_led();
    state = next_state[state];
```

```
void push_to_global_string(UART_HandleTypeDef *huart, enum InterruptState state) {
    char c = global_char[0];
    if (cur_strlen >= MAX_STRING_SIZE - 1) {
        cur_strlen = 0;
        send_overflow(huart, state);
    }
    if (c != '\r') {
        send_uart(huart, global_char, state, 1);
        global_string[cur_strlen++] = global_char[0];
    }
    if (c == '\r') {
        global_string[cur_strlen++] = '\0';
        run_command(huart, state, global_string);
        cur_strlen = 0;
    }
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    push_to_global_string(huart, Interrupt);
    HAL_UART_Receive_IT(huart, (uint8_t *)global_char, 1);
}

HAL_StatusTypeDef read_uart(UART_HandleTypeDef *huart, enum InterruptState state) {
    if (state == NonInterrupt) {
        HAL_StatusTypeDef status = HAL_UART_Receive(huart, (uint8_t *)global_char, 1, UART_MAX_TIMEOUT);
        if (status == HAL_OK)
            push_to_global_string(huart, state);
        return status;

    }
    return HAL_TIMEOUT;
}
```

Выводы

1. Изучили протокол передачи данных по интерфейсу UART.
2. Получили базовые знания об организации системы прерываний в микроконтроллерах на примере микроконтроллера STM32.
3. Изучили устройство и принципы работы контроллера интерфейса UART, получить навыки организации обмена данными по UART в режимах опроса и прерываний.