Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьтерной техники

Лабораторная работа №2 по Проектированию вычислительных систем

Вариант 1

Роман Юнусов

Алексей Лапин  
Группа P34102

Преподаватель Василий Пинкевич

Санкт-Петербург 2024

Оглавление

[Задание 3](#_Toc181968277)

[Схемы 3](#_Toc181968278)

[Решение 4](#_Toc181968279)

[Последовательный uart. 4](#_Toc181968280)

[Uart через прерывания. 5](#_Toc181968281)

[Парсер(анализ строки) 6](#_Toc181968282)

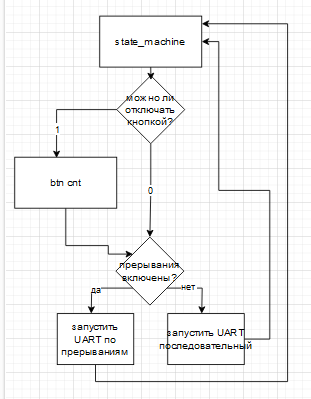
[Выводы 7](#_Toc181968283)

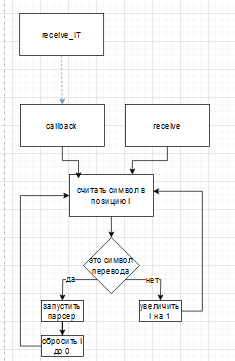
# Задание

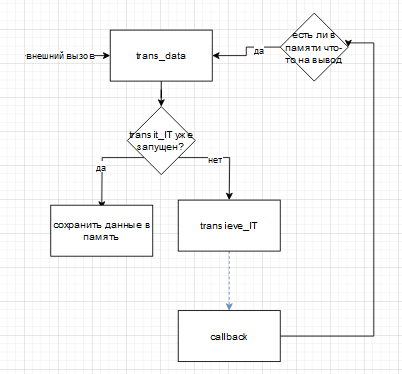
Доработать программу «светофор», добавив возможность отключения кнопки и  
задание величины тайм-аута (период, в течение которого горит красный).  
Должны обрабатываться следующие команды, посылаемые через UART:  
– *?* – в ответ стенд должен прислать состояние, которое отображается в данный момент  
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.

# Схемы

Общая







# Решение

Последовательный uart.  
ничего необычного, считываем символы пока может, при этом выводим их назад, при нажатии энтера делаем операцию анализа строки

**if**(!fsm\_obj.interupt\_enable) {

first\_iter = 0;

**if**(*HAL\_OK* == HAL\_UART\_Receive( &huart6, (uint8\_t \*) &in\_buffer[in\_curlen], 1, 10 )){

HAL\_UART\_Transmit( &huart6, (uint8\_t \*) &in\_buffer[in\_curlen], 1, 10 );

**if**(in\_buffer[in\_curlen] == '\r') {

in\_buffer[in\_curlen] = '\0';

deshifr(in\_buffer, &in\_curlen, &fsm\_obj);

in\_curlen = 0;

}

**else** {

in\_curlen++;

}

}

}

## Uart через прерывания.

Для начала определим операции включение и выключения прерываний, главное не забыть, что нужно сделать AbortReceive всех получений.

**void** **enable\_interrupt**(**struct** mech\_data\* tfl\_obj) {

HAL\_NVIC\_EnableIRQ(*USART6\_IRQn*);

tfl\_obj->interupt\_enable = 1;

}

**void** **disable\_interrupt**(**struct** mech\_data\* tfl\_obj) {

HAL\_UART\_AbortReceive(&huart6);

HAL\_NVIC\_DisableIRQ(*USART6\_IRQn*);

tfl\_obj->interupt\_enable = 0;

}

Просто запустили бесконечное считывание по 1 символу.

**else** {

**if**(first\_iter == 0){

init\_vals\_IT();

first\_iter = 1;

}

**else**

HAL\_UART\_Receive\_IT( &huart6, (uint8\_t \*) &el, 1);

}

Непосредственно в callbacke будет происходить заполнение буффера, в целом всё похоже на последовательный юарт

**void** **HAL\_UART\_RxCpltCallback**(UART\_HandleTypeDef \*huart) {

in\_buffer\_IT[len\_in\_buf] = el;

transend\_data\_IT(&el, 1);

**if**(in\_buffer\_IT[len\_in\_buf] == '\r'){

**int** local\_len = len\_in\_buf;

deshifr(in\_buffer\_IT, &local\_len, &fsm\_obj);

len\_in\_buf = 0;

}

**else**{

len\_in\_buf = len\_in\_buf + 1;

}

}

А вот вывод в последовательном юарте хитрее. Для начала нужно вызвать функцию вывода transend\_data\_IT, если нет других операции вывода(за это отвечает send\_req), то мы просто выведем наши данные установив флаг и сбросив его в callback.  
Если же уже есть запрос на вывод, то мы просто сохраним наши данные в память, а потом после завершения операции вывода через print\_collected\_data после callbackа проверим приходили ли нам ещё запросы на вывод, пока мы выполняли наш, и если приходили, то мы их выполним

**void** **transend\_data\_IT**(**char**\* buf, **int** len) {

**if**(send\_req){

data\_collecter(buf, len);

**return**;

}

send\_req = 1;

HAL\_UART\_Transmit\_IT( &huart6, (uint8\_t \*) buf, 1);

}

**void** **HAL\_UART\_TxCpltCallback**(UART\_HandleTypeDef \*huart) {

send\_req = 0;

print\_collected\_data();

}

**void** **data\_collecter**(**char**\* buf, **int** len){

**for**(**int** i = 0;i < len; i++){

datas[last\_data] = buf[i];

last\_data++;

}

}

**void** **print\_collected\_data**(){

**if**(already\_printed < last\_data){

transend\_data\_IT(&datas[already\_printed], 1);

already\_printed++;

}

**else** {

already\_printed = 0;

last\_data = 0;

}

}

## Парсер(анализ строки)

Дешифрация простая, получили строчку, сравниваем её с теми что нам нужно, и отсюда изменяем параметры нашей системы. Также для предусмотрел разный вывод информации в зависимости от режима в котором мы сейчас находимся.

Для команды с общей информации о нынешней схеме выделено дополнительное время в timeout.

**void** **deshifr**(**char**\* in\_buf, **int**\* cur\_length, **struct** mech\_data\* tfl\_obj){

**int** cmp\_res = **strncmp**(in\_buf, "!", 1);

**if**(cmp\_res == 0){

**char** out\_buffer[100];

**int** transmitted\_data\_len;

transmitted\_data\_len = **sprintf**(out\_buffer, "current\_state is %s, timeout is %d, mode is %d, I is %d\r", getColorName(tfl\_obj->state), tfl\_obj->red\_time, tfl\_obj->work\_mode, tfl\_obj->interupt\_enable);

**if**(tfl\_obj->interupt\_enable) {

transend\_data\_IT(out\_buffer, transmitted\_data\_len);

}

**else**{

HAL\_UART\_Transmit( &huart6, (uint8\_t \*) out\_buffer, transmitted\_data\_len, 30 );

}

}

**char**\* set\_mode\_flag = **strstr**(in\_buf, "set mode ");

**if**(set\_mode\_flag) {

**int** val = (\*(set\_mode\_flag + 9) - '0');

tfl\_obj->work\_mode = val;

**if**(val == 0)

tfl\_obj->button\_pressed\_flag = 0;

**char** out\_buffer[100];

**int** transmitted\_data\_len;

transmitted\_data\_len = **sprintf**(out\_buffer, "data change to %d\r", val);

**if**(tfl\_obj->interupt\_enable){

transend\_data\_IT(out\_buffer, transmitted\_data\_len);

}

**else**{

HAL\_UART\_Transmit( &huart6, (uint8\_t \*) out\_buffer, transmitted\_data\_len, 10 );

}

}

**char**\* set\_red\_time = **strstr**(in\_buf, "set timeout ");

**if**(set\_red\_time) {

set\_red\_time = set\_red\_time + 12;

**int** val = 0;

**while**(\*set\_red\_time != '\0' && \*set\_red\_time != '\r') {

val = val \* 10 + (\*set\_red\_time - '0');

set\_red\_time++;

}

**if**(tfl\_obj->current\_red == tfl\_obj->red\_time)

tfl\_obj->current\_red = val;

**else**

tfl\_obj->current\_red = val / 4;

tfl\_obj->red\_time = val;

tfl\_obj->other\_time = tfl\_obj-> red\_time / 4;

**char** out\_buffer[100];

**int** transmitted\_data\_len;

transmitted\_data\_len = **sprintf**(out\_buffer, "red timeout change to %d\r", tfl\_obj->current\_red);

**if**(tfl\_obj->interupt\_enable){

transend\_data\_IT(out\_buffer, transmitted\_data\_len);

}

**else**{

HAL\_UART\_Transmit( &huart6, (uint8\_t \*) out\_buffer, transmitted\_data\_len, 10 );

}

}

cmp\_res = **strncmp**(in\_buf, "set interrupts on", 17);

**if**(cmp\_res == 0) {

enable\_interrupt(tfl\_obj);

}

cmp\_res = **strncmp**(in\_buf, "set interrupts off", 18);

**if**(cmp\_res == 0) {

disable\_interrupt(tfl\_obj);

}

**return**;

# Выводы

Мы использовали механизм UART для общения с выполняемой программой на стенде, для вывода справочной информации и задания параметров в runtime. Причём UART был реализован в двух режимах, через асинхронные операции с прерываниями и потоковый, последовательно с основной программой. А значит цель задания была нам полностью выполнена.