УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Проектирование вычислительных систем»

**Лабораторная работа №2**

Вариант 11

Студент

*Митрофанов Е. Ю.*

*Любкин А. С.*

*P34101*

Преподаватель

*Пинкевич В. Ю.*

Санкт-Петербург, 2022 г.

Задание лабораторной работы

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 72 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 задания

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. 73 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

Исходный код

*Объявления переменных:*

uint16\_t GREEN = GPIO\_PIN\_13;

uint16\_t YELLOW = GPIO\_PIN\_14;

uint16\_t RED = GPIO\_PIN\_15;

uint16\_t BLINKING\_GREEN = 0;

uint8\_t INT\_ON = 1;

uint8\_t INT\_OFF = 0;

uint16\_t current\_light = 1;

uint32\_t start\_time;

uint32\_t duration = 5000;

uint32\_t duration\_for\_red;

uint32\_t duration\_for\_yellow = 3000;

uint32\_t blink\_duration = 500;

uint32\_t blink\_count = 0;

uint8\_t button\_flag = 0;

uint8\_t interrupts\_mode = 0;

uint8\_t is\_writing\_now = 0;

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

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

uint8\_t status;

**char**\* cur\_process\_char = read\_buffer;

**char**\* cur\_read\_char = read\_buffer;

**char**\* transmit\_from\_pointer = write\_buffer;

**char**\* write\_to\_pointer = write\_buffer;

*Функция ожидания заданного временного интервала:*

**void** **wait**(uint32\_t duration)

{

uint32\_t startTime = HAL\_GetTick();

**while**((HAL\_GetTick() - startTime) < duration)

{

}

}

*Функция выключения всех световых индикаторов:*

**void** **turnAllOff**()

{

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14, *GPIO\_PIN\_RESET*);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15, *GPIO\_PIN\_RESET*);

}

*Функция включения заданного цвета:*

**void** **turnSpecificLightOn**(uint16\_t light)

{

turnAllOff();

HAL\_GPIO\_WritePin(GPIOD, light, *GPIO\_PIN\_SET*);

}

*Функция мигания заданным цветом определенное количество раз:*

**void** **blinkSpecificLight**(uint32\_t count, uint16\_t light, uint32\_t blinkDuration)

{

**for**(uint32\_t i = 0; i < count; i++){

turnAllOff();

wait(blinkDuration);

turnSpecificLightOn(light);

wait(blinkDuration);

}

}

*Функции работы со строками и вывода результата:*

**int** **check\_starts\_with**(**char**\* a, **char**\* b) {

size\_t i = 0;

**while** (a[i] != '\0') {

**if** (a[i] != b[i]) **return** 0;

i++;

}

**return** 1;

}

**char**\* **concat**(**char** \*s1, **char** \*s2) {

**char** \*result = **malloc**(**strlen**(s1) + **strlen**(s2) + 1);

**strcpy**(result, s1);

**strcat**(result, s2);

**return** result;

}

**void** **next**(**char** \*\*pointer, **char** \*buffer) {

**if**(\*pointer >= buffer + 100){

\*pointer = buffer;

}

**else** {

(\*pointer)++;

}

}

**void** **write\_char\_to\_buff**(**char** c) {

\*write\_to\_pointer = c;

next(&write\_to\_pointer, write\_buffer);

}

**void** **write**(**char**\* str) {

**char**\* str\_with\_newlines = concat("\r\n", str);

str\_with\_newlines = concat(str\_with\_newlines, "\r\n");

**int** size = **sizeof**(**char**)\***strlen**(str\_with\_newlines);

**if** (interrupts\_mode == 0) {

HAL\_UART\_Transmit(&huart6, (uint8\_t \*) str\_with\_newlines, size, 10);

} **else** {

**for**(size\_t i = 0; str\_with\_newlines[i] != '\0'; i++) {

write\_char\_to\_buff(str\_with\_newlines[i]);

}

}

}

**void** **write\_command\_not\_found**() {

**char**\* str = "Command not found. Available: '?', set mode 1/2, set timeout X, set interrupts on/off";

write(str);

}

**int** **is\_number**(**char**\* str) {

**for** (size\_t i = 0; str[i] != '\0'; i++) {

**if** (!isdigit(str[i])) **return** 0;

}

**return** 1;

}

*Функция обработки ввода команд:*

**void** **process\_symbol**() {

**if** (\*cur\_process\_char == '\r') {

\*cur\_process\_char = '\0';

**char**\* command = **strtok**(read\_buffer, " ");

**if** (**strcmp**(command, "?") == 0) {

**char** answer[100];

**char**\* light;

**switch** (current\_light) {

**case** GPIO\_PIN\_15:

light = "red";

**break**;

**case** GPIO\_PIN\_14:

light = "yellow";

**break**;

**case** GPIO\_PIN\_13:

light = "green";

**break**;

**case** 0:

light = "blinking green";

**break**;

}

uint8\_t mode;

**char** interrupts;

**if** (button\_flag == 0 && current\_light == RED) {

mode = 1;

} **else** mode = 2;

**if** (interrupts\_mode == 1) {

interrupts = 'I';

} **else** {

interrupts = 'P';

}

**sprintf**(answer, "Light: %s, Mode: %d, Timeout: %d, Interrupts: %c", light, mode, duration\*4, interrupts);

write(answer);

} **else** **if** (**strcmp**(command, "set") == 0) {

**char**\* first\_arg = **strtok**(NULL, " ");

**if** (**strcmp**(first\_arg, "mode") == 0) {

**char**\* mode = **strtok**(NULL, " ");

**if** (**strcmp**(mode, "1") == 0) {

button\_flag = 0;

duration\_for\_red = duration \* 4;

write("Entered mode 1");

} **else** **if** (**strcmp**(mode, "2") == 0) {

button\_flag = 1;

duration\_for\_red = duration;

write("Entered mode 2");

} **else** {

write\_command\_not\_found();

}

} **else** **if** (**strcmp**(first\_arg, "timeout") == 0) {

**char**\* timeout = **strtok**(NULL, " ");

**if** (is\_number(timeout)) {

**int** new\_dur = **atoi**(timeout) \* 1000;

**if** (duration\_for\_red == duration) {

duration\_for\_red = new\_dur / 4;

} **else** {

duration\_for\_red = new\_dur;

}

write(concat("New duration is ", timeout));

duration = new\_dur / 4;

} **else** {

write\_command\_not\_found();

}

} **else** **if** (**strcmp**(first\_arg, "interrupts") == 0) {

**char**\* interrupts = **strtok**(NULL, " ");

**if** (**strcmp**(interrupts, "on") == 0) {

interrupts\_mode = 1;

transmit\_from\_pointer = write\_to\_pointer;

cur\_read\_char = read\_buffer;

write("Interrupt mode on");

HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) cur\_read\_char, **sizeof**( **char** ));

} **else** **if** (**strcmp**(interrupts, "off") == 0) {

interrupts\_mode = 0;

HAL\_UART\_Abort\_IT(&huart6);

write("Interrupt mode off");

} **else** {

write\_command\_not\_found();

}

} **else** {

write\_command\_not\_found();

}

} **else** {

write\_command\_not\_found();

}

cur\_process\_char = read\_buffer;

} **else** {

next(&cur\_process\_char, read\_buffer);

}

}

*Основная программа:*

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART6\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

duration\_for\_red = 4 \* duration;

**if** (interrupts\_mode == 1) {

HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) cur\_read\_char, **sizeof**( **char** ));

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

//HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) s, sizeof( char ));

//HAL\_UART\_Transmit\_IT( &huart6, (uint8\_t \*) s, sizeof( s ));

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**switch**(interrupts\_mode) {

**case** 1:

**if** (cur\_process\_char != cur\_read\_char) {

process\_symbol();

}

**if**(is\_writing\_now == 0){

**if**(transmit\_from\_pointer != write\_to\_pointer) {

is\_writing\_now = 1;

HAL\_UART\_Transmit\_IT( &huart6, (uint8\_t \*) transmit\_from\_pointer, **sizeof**( **char** ));

}

}

**break**;

**case** 0:

status = HAL\_UART\_Receive(&huart6, (uint8\_t \*) cur\_process\_char, **sizeof**( **char** ), 100);

**if** (status == *HAL\_OK*) {

HAL\_UART\_Transmit(&huart6, (uint8\_t \*) cur\_process\_char, **sizeof**( **char** ), 10);

process\_symbol();

}

**break**;

}

**switch**(current\_light) {

**case** GPIO\_PIN\_15:

**if** (HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_15) == 0 && button\_flag == 0) {

duration\_for\_red = duration;

button\_flag = 1;

}

**if** ((HAL\_GetTick() - start\_time) >= duration\_for\_red) {

current\_light = GREEN;

duration\_for\_red = 4 \* duration;

button\_flag = 0;

turn\_specific\_light\_on(GREEN);

start\_time = HAL\_GetTick();

}

**break**;

**case** GPIO\_PIN\_14:

**if** ((HAL\_GetTick() - start\_time) >= duration\_for\_yellow) {

current\_light = RED;

turn\_specific\_light\_on(RED);

start\_time = HAL\_GetTick();

}

**break**;

**case** GPIO\_PIN\_13:

**if** ((HAL\_GetTick() - start\_time) >= duration) {

current\_light = BLINKING\_GREEN;

turn\_all\_off();

start\_time = HAL\_GetTick();

}

**break**;

**case** 0:

**if** ((HAL\_GetTick() - start\_time) >= blink\_duration) {

blink\_count++;

**if** (blink\_count < 6) {

blink(GREEN);

} **else** {

blink\_count = 0;

current\_light = YELLOW;

turn\_specific\_light\_on(YELLOW);

}

start\_time = HAL\_GetTick();

}

**break**;

**default**:

current\_light = RED;

turn\_specific\_light\_on(RED);

start\_time = HAL\_GetTick();

}

}

/\* USER CODE END 3 \*/

}

*Callback функции ввода и вывода:*

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

**if** (huart->Instance == huart6.Instance) {

**if** (\*cur\_read\_char == '\r'){

cur\_read\_char = read\_buffer;

}

**else** {

write\_char\_to\_buff(\*cur\_read\_char);

next(&cur\_read\_char, read\_buffer);

}

**if** (interrupts\_mode == 1)

HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) cur\_read\_char, **sizeof**( **char** ));

}

}

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

**if** (huart->Instance == huart6.Instance) {

is\_writing\_now = 0;

next(&transmit\_from\_pointer, write\_buffer);

}

}

Вывод

Во время выполнения лабораторной работы мы изучили работу интерфейса передачи данных UART и получили навыки организации обмена данными в двух режимах, а также углубили знания в системе прерываний в микроконтроллерах.