федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

|  |  |
| --- | --- |
| / | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Информатики и систем управления |
|  |  |
| КАФЕДРА | Проектирования и технологии производства ЭА |

**Отчет по лабораторной работе:**

по курсу          Проектирование микропроцессорных систем

на тему        Пять режимов работы светодиода с помощью таймеров и прерываний

   и информационный обмен с ПК по шине USB с помощью микросхемы USB-UART

|  |  |  |
| --- | --- | --- |
| Студент |  | Ж.О. Исроилов |
|  | (Подпись, дата) | (И.О.Фамилия) |
| Консультант |  | В.В. Леонидов |
|  | (Подпись, дата) | (И.О.Фамилия) |
|  |  |  |
|  |  |  |
| Отметки о сдачи лабораторной работы №2: |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Москва, 2021

**Исходные элементы.**

Отладочный комплект, светодиод, две кнопки.

**Задание.**

Необходимо запрограммировать микроконтроллер таким образом, чтобы при нажатии на одну кнопку светодиод горел ярче, а при нажатии на другую тусклее, при этом возможно только пять яркостей свечения светодиода. При последующие нажатиях схема никак не должна реагировать. Яркость регулируется с помощью таймеров, кнопки обрабатываются с помощью прерываний. Реализовать информационный обмен с ПК по шине USB с помощью микросхемы USB-UART. Реализовать данную задачу на языках программирования Си.

**Реализация программы на Cи**

Файл main.c

**#include** "main.h"

**char** RxBuffer[RX\_BUFF\_SIZE]; //Буфер приёма USART

**char** TxBuffer[TX\_BUFF\_SIZE]; //Буфер передачи USART

bool ComReceived; //Флаг приёма строки данных

**int** w=2;

bool led\_flag=false;

bool blink = false;

**void** **USART2\_IRQHandler**(**void**){

**if** ((USART2->SR & USART\_SR\_RXNE)!=0){ //Прерывание по приёму данных?

uint8\_t pos = **strlen**(RxBuffer); //Вычисляем позицию свободной ячейки

RxBuffer[pos] = USART2->DR; //Считываем содержимое регистра данных

**if** ((RxBuffer[pos]== 0x0A)) //Если это символ конца строки

{

ComReceived = true; //- выставляем флаг приёма строки

**return**; //- и выходим

}}}

**void** **TIM2\_IRQHandler**(**void**){

TIM2->SR &= ~TIM\_SR\_UIF; //Сброс флага переполнения

**if**((w>0)&&(w<10)){

**if**(led\_flag){

LED\_SWAP();

TIM2->ARR = (10-w+1)-1;

led\_flag = false;}

**else{**

LED\_SWAP();

TIM2->ARR = (w+1)-1;

led\_flag = true;}}

**else** **if**(w==10){

LED\_ON();

led\_flag = true;}

**else{**

LED\_OFF();

led\_flag = false;}}

**void** **TIM3\_IRQHandler**(**void**){

TIM3->SR &= ~TIM\_SR\_UIF; //Сброс флага переполнения

TIM3->CNT=0;

**if**(blink){

blink = false;

TIM2->CNT = 0;

TIM2->CR1 |= TIM\_CR1\_CEN;}

**else{**

blink = true;

TIM2->CR1 &= ~TIM\_CR1\_CEN;

TIM2->CNT = 0;

LED\_OFF();

led\_flag = false;}}

**void** **EXTI0\_IRQHandler**(**void**){

**if** (EXTI->PR & EXTI\_PR\_PR0){ // нас интересует EXTI0

EXTI->PR |= EXTI\_PR\_PR0;

delay(10000); //Задержка для защиты от дребезга контактов

w+=2;

**if**(w>10){

w-=2;}

TIM2->CNT=0;}

**void** **EXTI1\_IRQHandler**(**void**){

**if** (EXTI->PR & EXTI\_PR\_PR1) { // нас интересует EXTI13

EXTI->PR |= EXTI\_PR\_PR1;

delay(10000); //Задержка для защиты от дребезга контактов

w-=2;

**if**(w<0){

w+=2;}

TIM2->CNT=0;}}

**void** **initPorts**(**void**){

RCC->APB2ENR |= RCC\_APB2ENR\_IOPBEN; //включить тактирование GPIOB

//очистка полей

GPIOB->CRL &= ~(GPIO\_CRL\_CNF0 | GPIO\_CRL\_MODE0);

//и конфигурация

GPIOB->CRL |= GPIO\_CRL\_MODE0\_1; } //PA5, выход 2МГц

**void** **initButton**(**void**){

//Включить тактирование порта GPIOC и альтернативных функций

RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN | RCC\_APB2ENR\_AFIOEN;

//PA0

GPIOA->CRL &= ~(GPIO\_CRL\_MODE0 | GPIO\_CRL\_CNF0);

GPIOA->CRL |= GPIO\_CRL\_CNF0\_1;

GPIOA->BSRR |= GPIO\_BSRR\_BS0;

//PA1

GPIOA->CRL &= ~(GPIO\_CRL\_MODE1 | GPIO\_CRL\_CNF1);

GPIOA->CRL |= GPIO\_CRL\_CNF1\_1;

GPIOA->BSRR |= GPIO\_BSRR\_BS1;

//AFIO

AFIO->EXTICR[0] |= (AFIO\_EXTICR1\_EXTI0\_PA | AFIO\_EXTICR1\_EXTI1\_PA);

EXTI->FTSR |= (EXTI\_FTSR\_TR0 | EXTI\_FTSR\_TR1);

EXTI->IMR |= (EXTI\_IMR\_MR0 | EXTI\_IMR\_MR1);

//NVIC PA0

NVIC\_EnableIRQ(*EXTI0\_IRQn*);

NVIC\_SetPriority(*EXTI0\_IRQn*, 0);

//NVIC PA1

NVIC\_EnableIRQ(*EXTI1\_IRQn*);

NVIC\_SetPriority(*EXTI1\_IRQn*, 0);}

**void** **initTIM2**(**void**){

RCC->APB1ENR |= RCC\_APB1ENR\_TIM2EN; //Включить тактирование TIM6

//Частота APB1 для таймеров = APB1Clk \* 2 = 32МГц \* 2 = 64МГц

TIM2->PSC = 64000-1; //Предделитель частоты (64МГц/64000 = 1кГц)

TIM2->ARR = (500)-1; //Модуль счёта таймера (1кГц/1000 = 1с)

TIM2->DIER |= TIM\_DIER\_UIE; //Разрешить прерывание по переполнению таймера

TIM2->CR1 |= TIM\_CR1\_CEN; //Включить таймер

**if**((w>0)&&(w<10)){

TIM2->CNT=0;

TIM2->ARR = (w+1)-1; //Модуль счёта таймера (1кГц/1000 = 1с)

LED\_ON();

**else** **if**(w==10){

LED\_ON();

led\_flag=true;}

**else{**

LED\_OFF();

led\_flag=false;}

NVIC\_EnableIRQ(*TIM2\_IRQn*); //Рарзрешить прерывание от TIM2

NVIC\_SetPriority(*TIM2\_IRQn*, 2); } //Выставляем приоритет

**void** **initTIM3**(**void**){

RCC->APB1ENR |= RCC\_APB1ENR\_TIM3EN; //Включить тактирование TIM6

//Частота APB1 для таймеров = APB1Clk \* 2 = 32МГц \* 2 = 64МГц

TIM3->PSC = 64000-1; //Предделитель частоты (64МГц/64000 = 1кГц)

TIM3->ARR = (1000)-1; //Модуль счёта таймера (1кГц/1000 = 1с)

TIM3->DIER |= TIM\_DIER\_UIE; //Разрешить прерывание по переполнению таймера

NVIC\_EnableIRQ(*TIM3\_IRQn*); //Рарзрешить прерывание от TIM2

NVIC\_SetPriority(*TIM3\_IRQn*, 1); } //Выставляем приоритет

**void** **initClk**(**void**){

// Enable HSI

RCC->CR |= RCC\_CR\_HSION;

**while**(!(RCC->CR & RCC\_CR\_HSIRDY)){};

// Enable Prefetch Buffer

FLASH->ACR |= FLASH\_ACR\_PRFTBE;

// Flash 2 wait state

FLASH->ACR &= ~FLASH\_ACR\_LATENCY;

FLASH->ACR |= FLASH\_ACR\_LATENCY\_2;

// HCLK = SYSCLK

RCC->CFGR |= RCC\_CFGR\_HPRE\_DIV1;

// PCLK2 = HCLK

RCC->CFGR |= RCC\_CFGR\_PPRE2\_DIV1;

// PCLK1 = HCLK/2

RCC->CFGR |= RCC\_CFGR\_PPRE1\_DIV2;

// PLL configuration: PLLCLK = HSI/2 \* 16 = 64 MHz

RCC->CFGR &= ~RCC\_CFGR\_PLLSRC;

RCC->CFGR |= RCC\_CFGR\_PLLMULL16;

// Enable PLL

RCC->CR |= RCC\_CR\_PLLON;

// Wait till PLL is ready

**while**((RCC->CR & RCC\_CR\_PLLRDY) == 0) {};

// Select PLL as system clock source

RCC->CFGR &= ~RCC\_CFGR\_SW;

RCC->CFGR |= RCC\_CFGR\_SW\_PLL;

// Wait till PLL is used as system clock source

**while** ((RCC->CFGR & RCC\_CFGR\_SWS) != RCC\_CFGR\_SWS\_PLL){};}

**void** **initUSART2**(**void**){

RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN;

RCC->APB2ENR |= RCC\_APB2ENR\_AFIOEN; //включить тактирование альтернативных ф-ций портов

RCC->APB1ENR |= RCC\_APB1ENR\_USART2EN; //включить тактирование UART2

GPIOA->CRL &= ~(GPIO\_CRL\_MODE2 | GPIO\_CRL\_CNF2); //PA2 на выход

GPIOA->CRL |= (GPIO\_CRL\_MODE2\_1 | GPIO\_CRL\_CNF2\_1);

GPIOA->CRL &= ~(GPIO\_CRL\_MODE3 | GPIO\_CRL\_CNF3); //PA3 - вход

GPIOA->CRL |= GPIO\_CRL\_CNF3\_0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Скорость передачи данных - 115200

Частота шины APB1 - 32МГц

1. USARTDIV = 32'000'000/(16\*115200) = 17.4

2. 17 = 0x11

3. 16\*0.4 = 6

4. Итого 0x116

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USART2->BRR = 0xD05;

USART2->CR1 |= USART\_CR1\_RE | USART\_CR1\_TE | USART\_CR1\_UE;

USART2->CR1 |= USART\_CR1\_RXNEIE; //разрешить прерывание по приему байта данных

NVIC\_EnableIRQ(*USART2\_IRQn*);}

**void** **txStr**(**char** \*str, bool crlf)

{

uint16\_t i;

**if** (crlf) //если просят,

**strcat**(str,"\r\n"); //добавляем символ конца строки

**for** (i = 0; i < **strlen**(str); i++)

{

USART2->DR = str[i]; //передаём байт данных

**while** ((USART2->SR & USART\_SR\_TC)==0) {}; //ждём окончания передачи

}

}

**void** **ExecuteCommand**(**void**){

**memset**(TxBuffer,0,**sizeof**(TxBuffer)); //Очистка буфера передачи

/\* Обработчик команд \*/

**if** (**strncmp**(RxBuffer,"\*IDN?",5) == 0) //Это команда "\*IDN?"

{

**strcpy**(TxBuffer,"Isroilov IU4-73B"); //Она самая, возвращаем строку идентификации

}

**else** **if** (**strncmp**(RxBuffer,"BLINK ON",8) == 0) //Команда запуска таймера?

{

blink = true;

TIM3->CR1 |= TIM\_CR1\_CEN;

**strcpy**(TxBuffer, "OK");

}

**else** **if** (**strncmp**(RxBuffer,"BLINK OFF",9) == 0) //Команда остановки таймера?

{

TIM3->CR1 &= ~TIM\_CR1\_CEN;

TIM3->CNT = 0;

TIM2->CNT = 0;

TIM2->CR1 |= TIM\_CR1\_CEN;

**strcpy**(TxBuffer, "OK");

}

**else** **if** (**strncmp**(RxBuffer,"BRIGHTNESS",10) == 0) //Команда изменения периода таймера?

{

uint16\_t tim\_value;

**sscanf**(RxBuffer,"%\*s %hu", &tim\_value); //преобразуем строку в целое число

**if** ((0 <= tim\_value) && (tim\_value <= 100))//параметр должен быть в заданных пределах!

{

w=tim\_value/10;

TIM2->CNT = 0;

**if**((w>0)&&(w<10))

{

TIM2->CNT=0;

TIM2->ARR = (w+1)-1; //Модуль счёта таймера (1кГц/1000 = 1с)

}

**else** **if**(w==10)

{

LED\_ON();

led\_flag=true;

}

**else**

{

LED\_OFF();

led\_flag=false;

}

**strcpy**(TxBuffer, "OK");

}

**else**

**strcpy**(TxBuffer, "Parameter is out of range"); //ругаемся

}

**else**

**strcpy**(TxBuffer,"Invalid Command");//Если мы не знаем, чего от нас хотят, ругаемся в ответ

txStr(TxBuffer,true); //Отправляем буефер передачи с символами конца строки

**memset**(RxBuffer,0,RX\_BUFF\_SIZE); //Очистка буфера приёма

ComReceived = false; //Сбрасываем флаг приёма строки

}

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

{

initClk();

initPorts();

initButton();

initTIM2();

initTIM3();

initUSART2();

**while**(true){

**if** (ComReceived) //Ждём приема строки

ExecuteCommand();}}

Файл main.h

**#ifndef** \_\_MAIN\_H

**#define** \_\_MAIN\_H

**#include** "stm32f1xx.h"

**#include** "stdbool.h"

**#include** "string.h"

**#include** "stdio.h"

**#define** RX\_BUFF\_SIZE 256

**#define** TX\_BUFF\_SIZE 256

**#define** DELAY\_VAL 1000000

**#define** LED\_ON() GPIOB->BSRR = GPIO\_BSRR\_BS0

**#define** LED\_OFF() GPIOB->BSRR = GPIO\_BSRR\_BR0

**#define** LED\_SWAP() GPIOB->ODR ^= GPIO\_ODR\_ODR0

**void** **initUSART2**(**void**);

**void** **txStr**(**char** \*str, bool crlf);

**void** **ExecuteCommand**(**void**);

**void** **initClk**(**void**);

**void** **initPorts**(**void**);

**void** **initButton**(**void**);

**void** **initTIM2**(**void**);

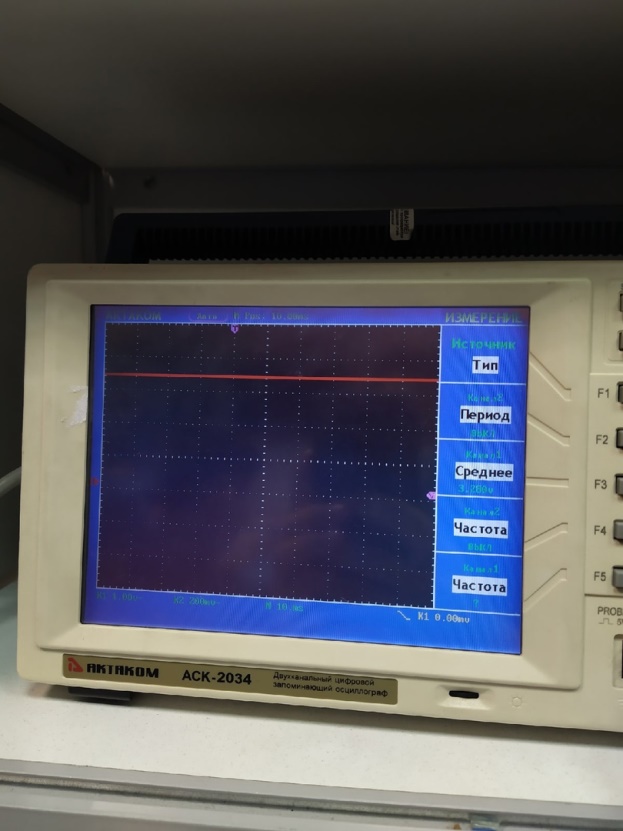
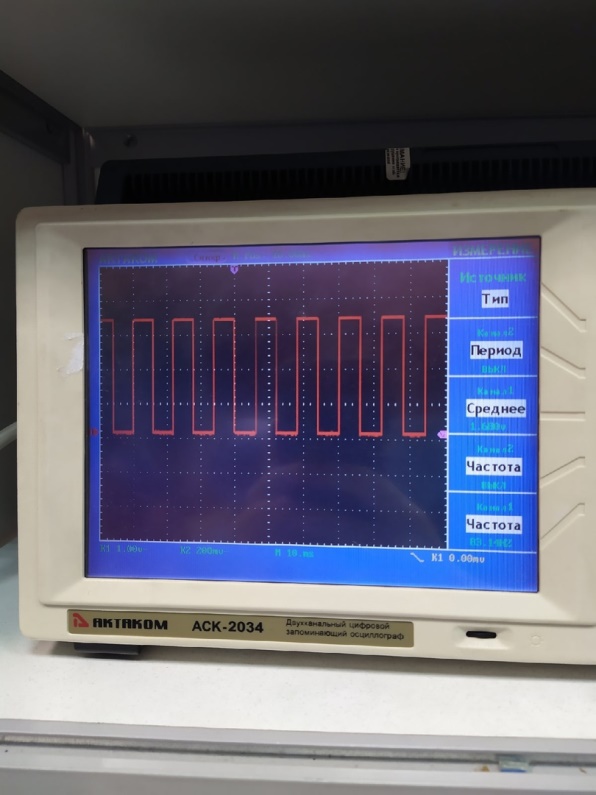
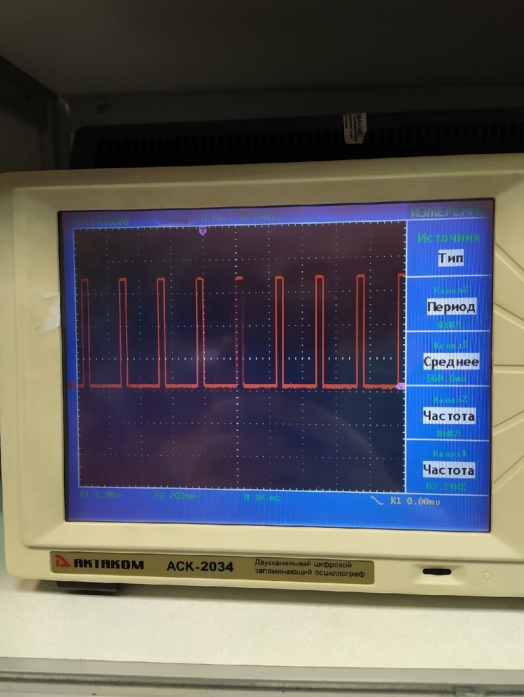
**void** **delay**(uint32\_t takts);

**#endif**

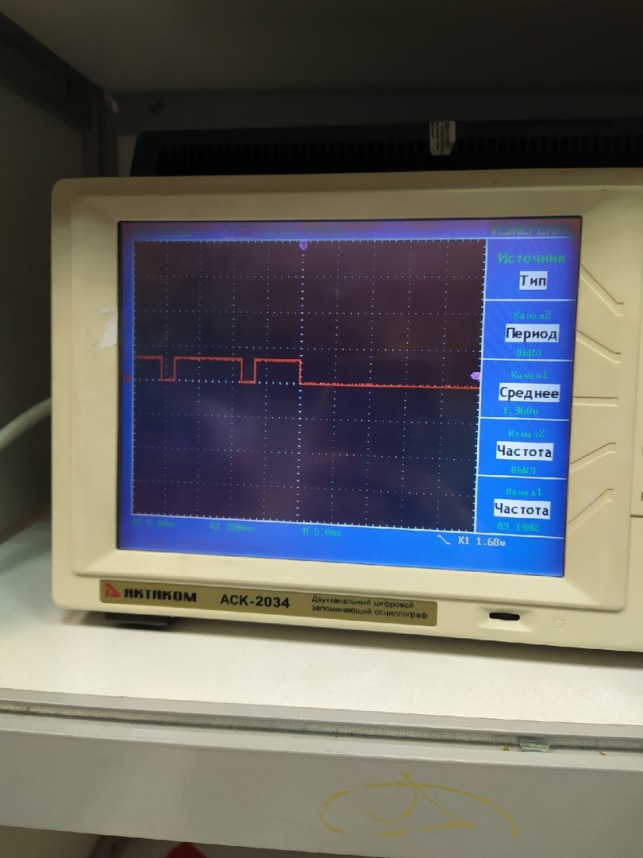
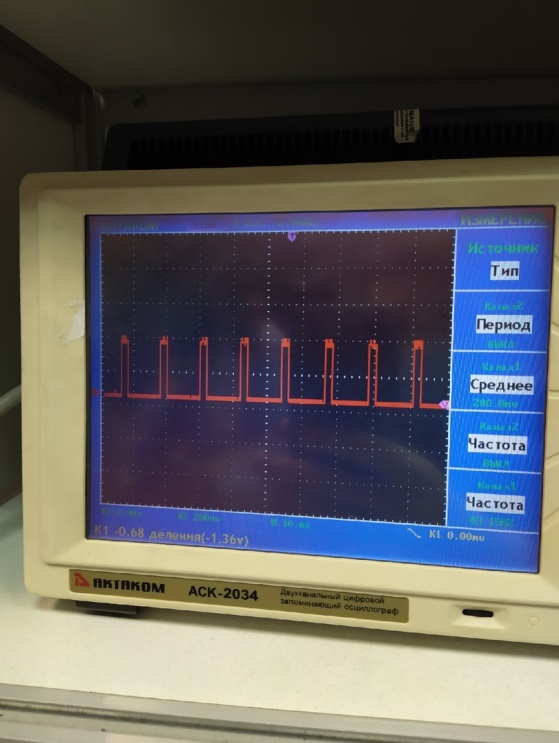
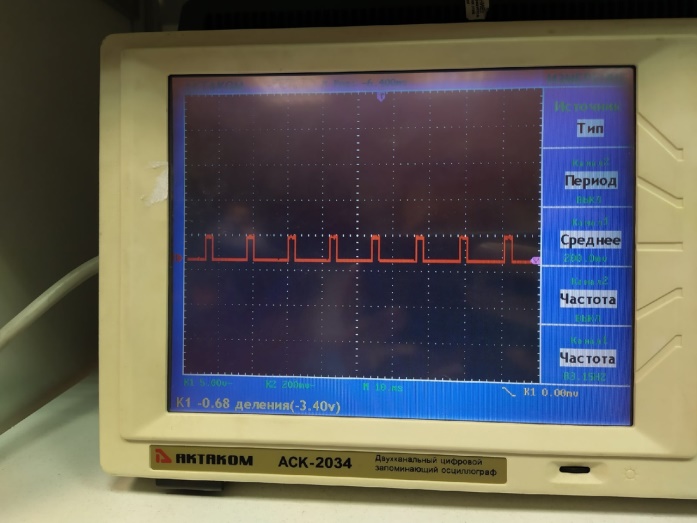
**Графики**

Полученные графики соответствуют каждому из пяти режимов яркости светодиода, также соответствуют обработке команд по USB-UART.

Brightness 10,50,100



Blink On



Команда \*ISDN?

