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

|  |  |  |
| --- | --- | --- |
| Студент |  | Ж.О. Исроилов |
|  | (Подпись, дата) | (И.О.Фамилия) |
| Консультант |  | В.В. Леонидов |
|  | (Подпись, дата) | (И.О.Фамилия) |
|  |  |  |
|  |  |  |
| Отметки о сдачи лабораторной работы №1: |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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. Реализовать данную задачу на языках программирования Си и Ассемблер.

**Реализация программы на Cи**

Файл main.c

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

{

uint32\_t D;

uint8\_t n=1;

uint32\_t W;

RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN; //включить тактирование GPIOA

//очистка полей

GPIOA->CRL &= ~(GPIO\_CRL\_CNF5 | GPIO\_CRL\_MODE5);

//и конфигурация

GPIOA->CRL |= GPIO\_CRL\_MODE5\_1;

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

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

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

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

//Бесконечный цикл

**while**(1) {

**if**((GPIOA->IDR & GPIO\_IDR\_IDR0)==0x00)

{

**if**(n<5)

{

delay(push\_delay);

n++;

}

}

**if**((GPIOA->IDR & GPIO\_IDR\_IDR1)==0x00)

{

**if**(n>1)

{

delay(push\_delay);

n--;

}

}

W=2000\*n;

D=T-W;

LED\_ON();

delay(W);

LED\_OFF();

delay(D);

}

}

Файл main.h

**#ifndef** MAIN\_H\_

**#define** MAIN\_H\_

**#include** "stm32f1xx.h"

// Величина задержки между вкл/выкл светодиодов

**#define** T 10000

**#define** push\_delay 100000

/\* Управление светодиодами \*/

**#define** LED\_ON() GPIOA->BSRR = GPIO\_BSRR\_BS5

**#define** LED\_OFF() GPIOA->BSRR = GPIO\_BSRR\_BR5

**#define** PB\_UP() GPIOA->BSRR = GPIO\_BSRR\_BS0

**#define** PB\_DOWN() GPIOA->BSRR = GPIO\_BSRR\_BS1

/\* Прототипы функций \*/

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

**#endif** /\* MAIN\_H\_ \*/

**Графики**

Полученные графики соответствуют каждому из пяти режимов яркости светодиода

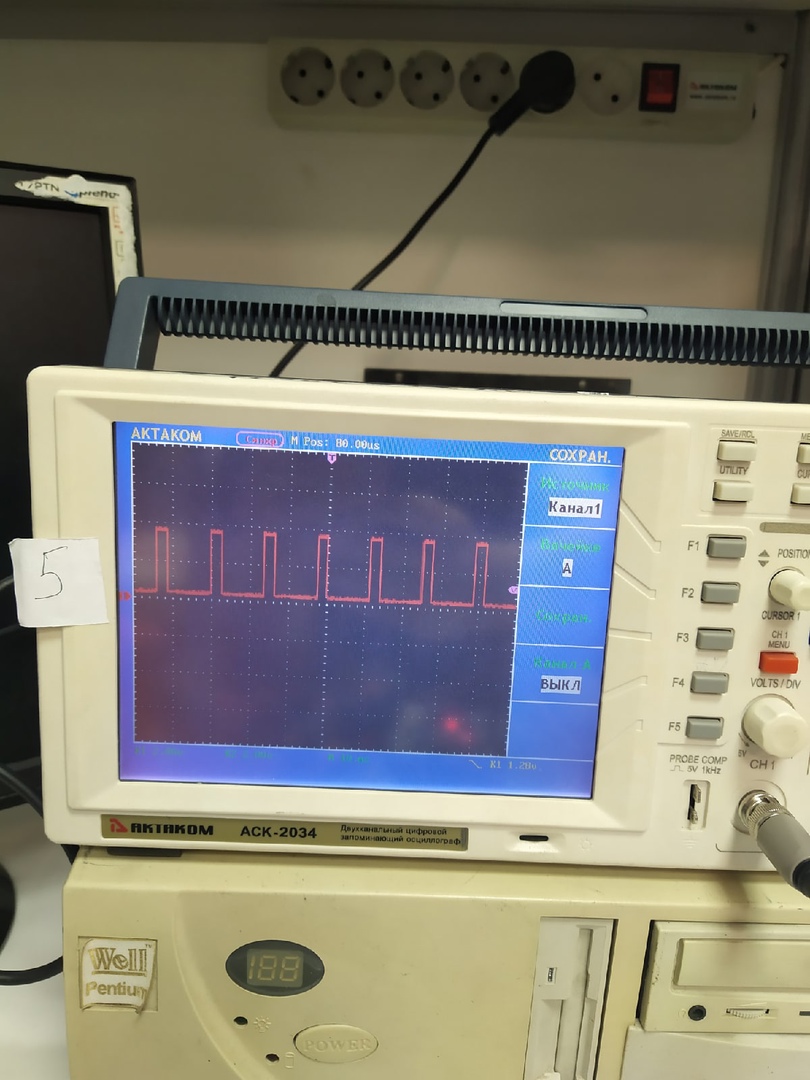


Рисунок 1 – Сигнал, полученный при 1 режиме яркости светодиода

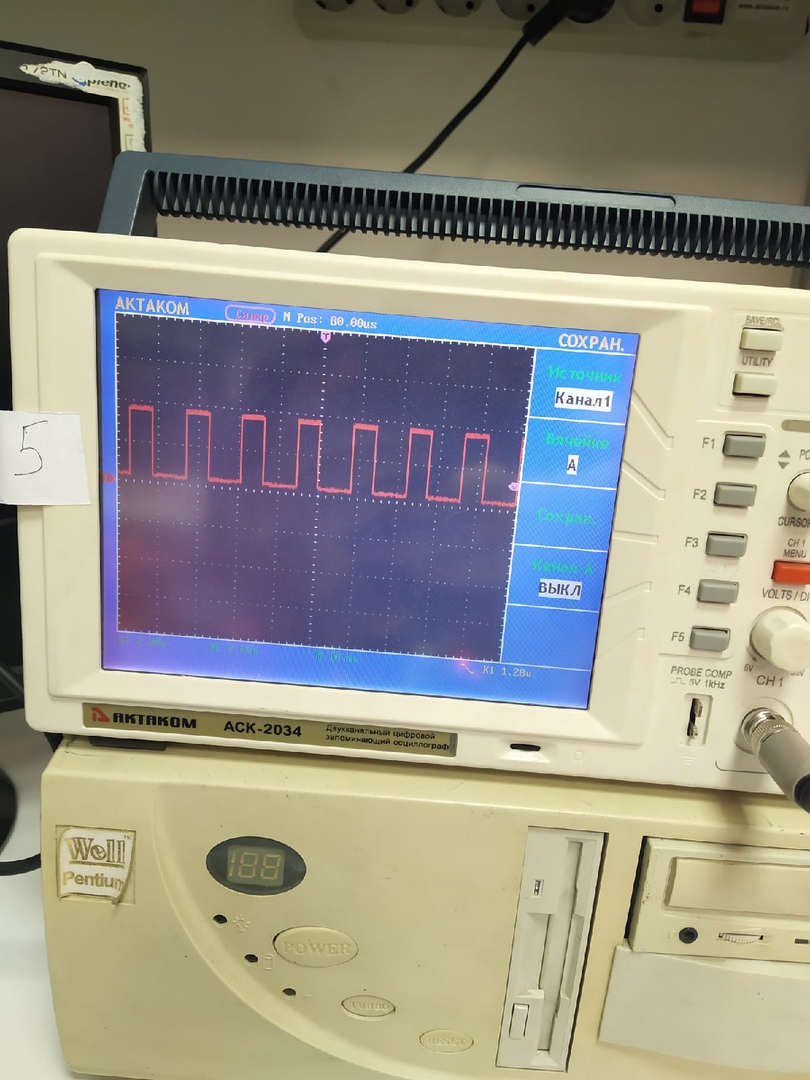


Рисунок 2 – Сигнал, полученный при 2 режиме яркости светодиода

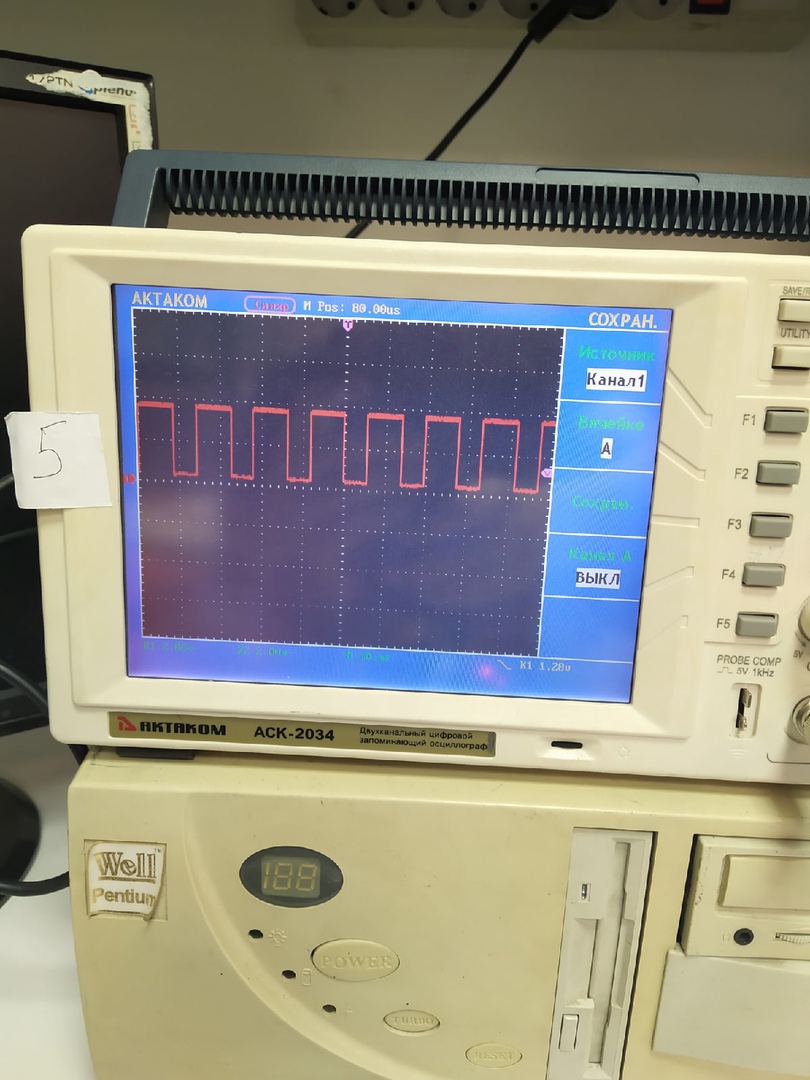


Рисунок 3 – Сигнал, полученный при 3 режиме яркости светодиода

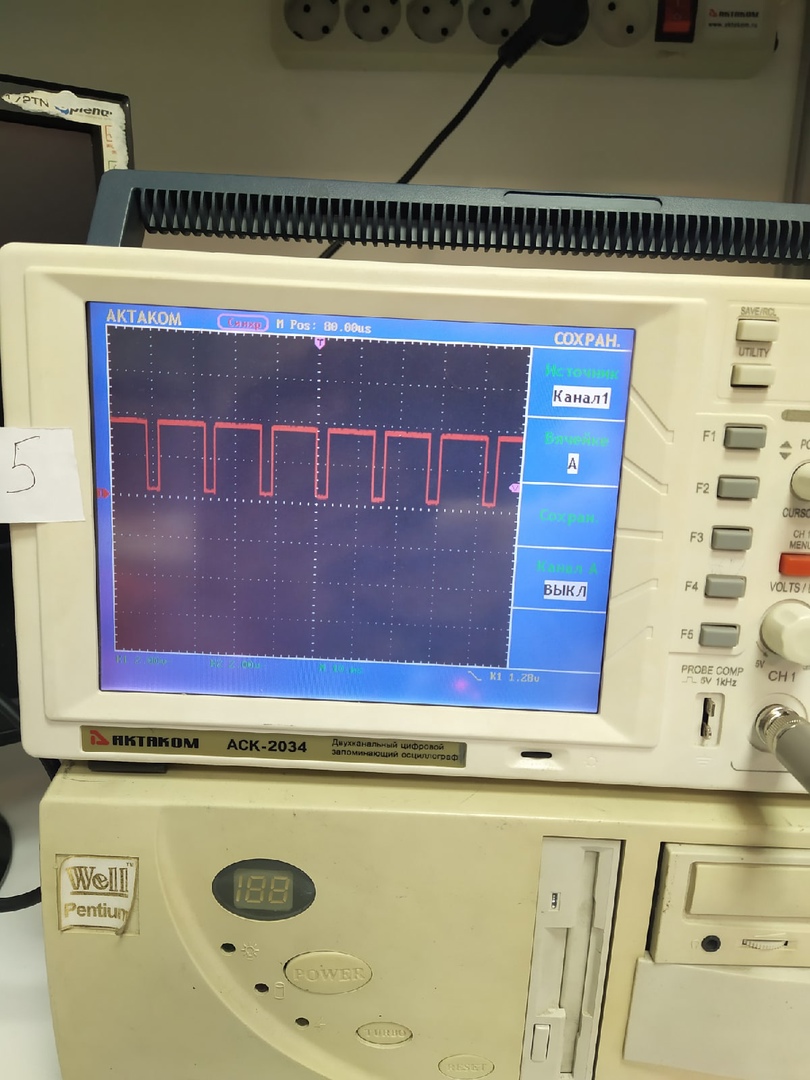


Рисунок 4 – Сигнал, полученный при 4 режиме яркости светодиода

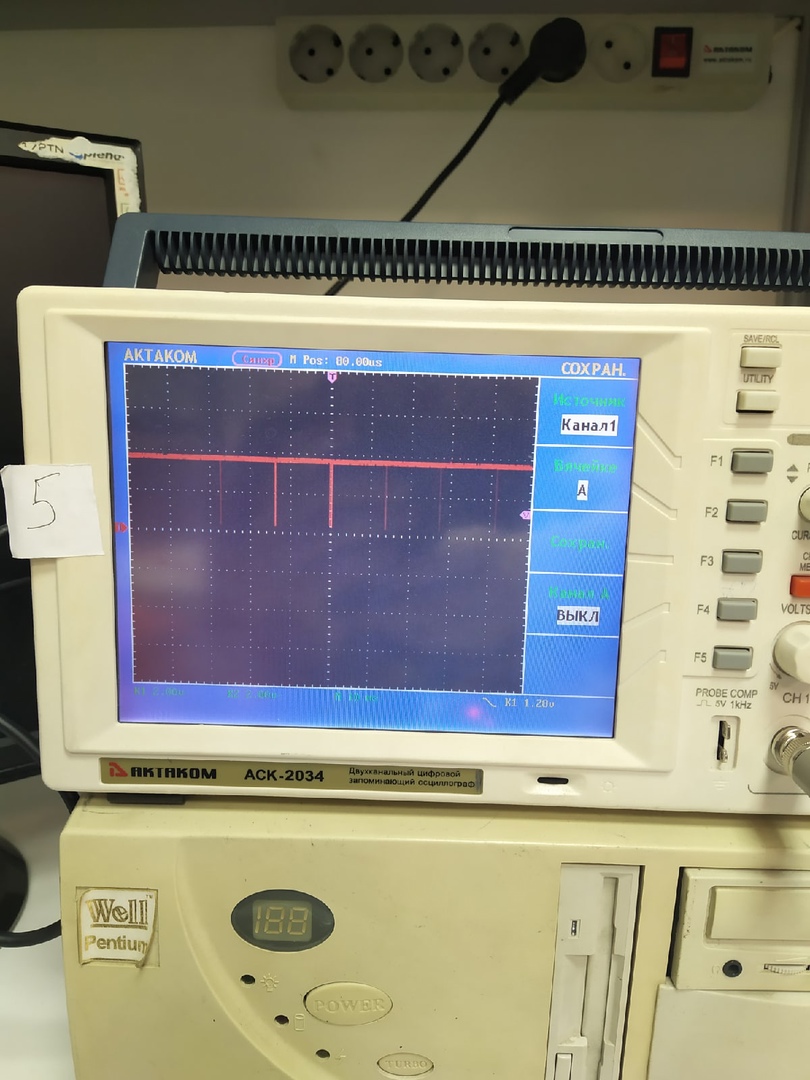


Рисунок 5 – Сигнал, полученный при 5 режиме яркости светодиода

**Реализация программы на Ассемблере**

**Reset\_Handler:**

ldr r0, =\_estack

mov sp, r0 /\* set stack pointer \*/

mov32 r0, PERIPH\_BB\_BASE+ (RCC\_APB2ENR-PERIPH\_BASE)\*32 + 2\*4

@ вычисляем адрес для BitBanding 5-го бита регистра RCC\_APB2ENR

@ BitAddress = BitBandBase + (RegAddr \* 32) + BitNumber \* 4

//Настройка вывода PA5

mov r1, #1

@ включаем тактирование порта A (в 5-й бит RCC\_APB2ENR пишем '1`)

str r1, [r0] @ загружаем это значение

mov32 r0, GPIOA\_CRL @ адрес порта

mov r1, #0x03 @ 4-битная маска настроек для Output mode 50mHz, Push-Pull ("0011")

ldr r2, [r0] @ считать порт

bfi r2, r1, #20, #4 @ скопировать биты маски в позицию PIN5

str r2, [r0] @ загрузить результат в регистр настройки порта

//Настройка вывода PA0

mov r1, #0x08 @Настройка MODE00 и CNF10

bfi r2, r1, #0, #4 @Запись в регистр CRL 0x08(1000) в 0ю позицию по ширине 4бита

str r2, [r0] @Запись найстройки CRL в память GPIO\_CRL

//Настройка вывода PA1

mov r1, #0x08 @Настройка MODE00 и CNF10

bfi r2, r1, #4, #4 @Запись в регистр CRL 0x08(1000) в 4ю позицию по ширине 4бита

str r2, [r0] @Запись найстройки CRL в память GPIO\_CRL

mov32 r0, GPIOA\_BSRR @ адрес порта выходных сигналов

mov32 r1, PERIPH\_BB\_BASE+ (GPIOA\_IDR-PERIPH\_BASE)\*32 + 0\*4 @адрес пина GPIOA\_IDR\_IDR0 ввода/вывода только для чтения

mov32 r2, PERIPH\_BB\_BASE+ (GPIOA\_IDR-PERIPH\_BASE)\*32 + 1\*4 @адрес пина GPIOA\_IDR\_IDR1 ввода/вывода только для чтения

mov32 r12, #12000 //Период T

mov32 r11, #2000 //Ширина импульса W

mov32 r10, #1 //Счетчик n

**loop:** @ Бесконечный цикл

mov32 r3, #GPIO\_BSRR\_BS5 //включаем светодиод

mov32 r4, #GPIO\_BSRR\_BR5 //выключаем светодиод

//PA0 = 1

ldr r5, [r1] @ Загрузка данных из вывода GPIOA\_IDR->GPIO\_IDR0

cmp r5, #0 @ Сравниваем r5 с логической 0 (Проверка на нажатие кнопки pa0)

it ne @ Если не равны

bne pa0 @ Переходим по метке pa0 (скипаем строки 82-86)

add r10, #1 @ r10=r10+1

cmp r10, #6 @ Сравниваем r10 и 6

it cs @ Если r10 больше либо равно 6

subcs r10, #1 @ r10=r10-1

bl push\_delay @ Переходим по метке push\_delay (задержа при нажатии кнопки)

**pa0:**

//PA1 = 1

ldr r5, [r2] @Загрузка данных из вывода GPIOA\_IDR->GPIO\_IDR1

cmp r5, #0 @ Сравниваем r5 с логической 0 (Проверка на нажатие кнопки pa1)

it ne @ Если не равны

bne pa1 @ Переходим по метке pa0 (скипаем строки 94-98)

sub r10, #1 @ r10=r10+1

cmp r10, #1 @ Сравниваем r10 и 6

it cc @ r10 меньше 1

addcc r10, #1 @ Прибавляем 1 к r10

bl push\_delay @ Переходим по метке push\_delay (Задержка после нажатия)

**pa1:**

str r3, [r0] //логическая 1/ на PA5 записываем 1 в PA5 по GPIO\_BSRR

bl delay @переходим к метке delay (задержка равна ширине импульса)

str r4, [r0] //логический 0 на PA5/ записываем 0 на PA5 по GPIO\_BSRR

bl delay //Переходим к метке delay (задержка равна D=T-W\*n)

b loop @ возвращаемся к началу цикла

**delay:** @ Подпрограмма задержки

push {r0} @ Загружаем в стек R0, т.к. его значение будем менять

mov32 r6, PERIPH\_BB\_BASE+(GPIOA\_IDR-PERIPH\_BASE)\*32 + 5\*4

@ в r6 записываем адрес GPIO\_IDR\_IDR5

ldr r5, [r6] @ загружаем значение по адресу r6 в r5

cmp r5, #1 @ сравниваем r5 с 1

ite eq @ они равны

muleq r9, r10, r11 @да - r9=r10\*r11

mlsne r9, r10, r11, r12 @нет - r9=r12-(r10\*r11)

mov r0, r9 @ псевдоинструкция Thumb (загрузить константу в регистр)

**delay\_loop:**

subs r0, #1 @ SUB с установкой флагов результата

it NE @ if(ложь)

bne delay\_loop @ переход, если Z==0 (результат вычитания не равен нулю)

pop {r0} @ Выгружаем из стека R0

bx lr @ выход из подпрограммы (переход к адресу в регистре LR - вершина стека)

**push\_delay:**

push {r0} @загружаем r0 в стек

mov32 r0, #2000000 @записываем в r0 число

**push\_delay\_loop:**

subs r0, #1 @SUB с установкой флагов результата

it NE @if(ложь)

bne push\_delay\_loop

pop {r0}

bx lr