Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №5

«Программирование клавиатуры»

Вариант 6

Выполнил: Проверил:

Студент группы 150504 Преподаватель

Горбачевский К.В. Одинец Д.Н.

Минск, 2023

1. Постановка задачи

Программируя клавиатуру помигать ее индикаторами. Алгоритм мигания произвольный. Условия реализации программы, необходимые для выполнения лабораторной работы:

1. Запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считывание и анализом регистра состояния контроллера клавиатуры.
2. Для каждого байта команды необходимо считывать и анализировать код возврата. В случае считывания кода возврата, требующего повторить передачу байта, необходимо повторно, при необходимости – несколько раз, выполнить передачу байта. При этом повторная передача данных не исключает выполнения всех оставшихся условий.
3. Для определения момента получения кода возврата необходимо использовать аппаратное прерывания от клавиатуры.
4. Все коды возврата должны быть выведены на экран в шестнадцатеричной форме.
5. Алгоритм

Переопределить обработчик прерывания клавиатуры. Добавить вывод состояний.

Записывать новые значения состояний для светодиодов в порт 60h с проверкой заполнения буфера обмена.

1. Листинг программы

Далее приведен листинг программы, реализующей все поставленные задачи.

#include <stdio.h>

#include <stdlib.h>

#include <dos.h>

#include <conio.h>

typedef enum { false, true } bool;

#define KEYBOARD\_INTERRUPT 0x09

#define KEYBOARD\_LIGHTS\_CODE 0xED // Для управления индикаторами через 60h отправляется код EDh.

#define NONE 0x00

#define SCROLL\_LOCK 0x01

#define NUM\_LOCK 0x02

#define CAPS\_LOCK 0x04

typedef unsigned char byte;

int command\_succeeded = 0;

void interrupt (\*old\_handler)(void);

void interrupt new\_handler(void) {

byte scan\_code = inp(0x60);

// FA – байт обработан успешно,

// FE – произошла ошибка.

printf("%X\n", scan\_code);

command\_succeeded = (scan\_code == 0xFA);

old\_handler();

}

void set\_mask(byte mask) {

int i = 0;

while (!command\_succeeded) {

// 60 - регистр данных, 64 - регистр состояния

// Ждем пока буфер освободится

while ((inp(0x64) & 0x02));

outp(0x60, KEYBOARD\_LIGHTS\_CODE); // посылаем в регистр данных код, для управления клавиатурой

// Ждем пока буфер освободится

while ((inp(0x64) & 0x02));

outp(0x60, mask); // посылаем в регистр данных код клавиши

// В случае ошибки передачу байта нужно повторить. Пересылка выполняется до 3 раз,

// если ошибка не исчезла, нужно вывести сообщение и выйти из программы.

if (++i == 3) {

fputs("Failed to set mask 3 times in a row\n", stderr);

setvect(KEYBOARD\_INTERRUPT, old\_handler);

exit(EXIT\_FAILURE);

}

}

command\_succeeded = 0;

}

int main(void) {

old\_handler = getvect(KEYBOARD\_INTERRUPT); // получаем старый обработчик

setvect(KEYBOARD\_INTERRUPT, new\_handler); // устанавливаем по адресу новый обработчик

puts("\nSTART\n\n");

set\_mask(NUM\_LOCK); // зажигаем нум лок

delay(1500);

set\_mask(CAPS\_LOCK); // зажигаем капс лок

delay(1500);

set\_mask(SCROLL\_LOCK); // зажигаем скролл лок

delay(1500);

set\_mask(NONE); // тушим всё

delay(1500);

set\_mask(SCROLL\_LOCK | NUM\_LOCK | CAPS\_LOCK); // зажигаем все

delay(1500);

set\_mask(NONE);

setvect(KEYBOARD\_INTERRUPT, old\_handler); // возвращаем на место старый обработчик

return EXIT\_SUCCESS;

}

1. Тестирование программы

Рисунок 4.1 — Вывод программы



1. Заключение

В данной лабораторной работе были выполнены все поставленные задачи: написана программа, которая управляет диодами клавиш NUMLOCK, CAPSLOCK, SCROLLLOCK. Были выведены статусы изменения состояния этих светодиодов

Программа компилировалась в Turbo C++ и запускалась в DOS, который эмулировался с помощью VirtualBox.