Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №1

«Последовательный порт»

Выполнил: Проверил:

Студент группы 250504 к.т.н., доцент

Казаченко П.Е. Одинец Д.Н.

Минск 2024

1. Постановка задачи

Разработать программный модуль реализации процедуры передачи (приёма) байта информации через последовательный интерфейс.

Программа должна демонстрировать программное взаимодействие с последовательным интерфейсом с использованием следующих механизмов:

1. Прямое взаимодействие с портами ввода-вывода (wirte, read)
2. Использование BIOS прерывания 14h
3. Работа с COM-портом через регистры как с устройствами ввода-вывода.
4. Алгоритм

Программа состоит из нескольких подпрограмм (частей программы), представляющих собой некоторые функции. К ним относятся функции:

* Инициализация порта
* Запись байта информации в порт
* Чтение байта информации из порта
* Вывод результата на экран

1. Листинг программы

Далее приведены листинги программ, реализующие различные механизмы передачи (приёма) информации через последовательный интерфейс.

3.1. Листинг программы, прямо взаимодействующей с портами ввода-вывода (write, read).

#include <iostream>

#include <windows.h>

enum Mode {

WRITE,

READ

};

class Port {

private:

HANDLE portHandle;

LPCTSTR portName;

public:

Port(LPCTSTR port, Mode mode) : portName(port) {

if (mode == WRITE) {

portHandle = CreateFile(portName, GENERIC\_WRITE, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

}

else if (mode == READ) {

portHandle = CreateFile(portName, GENERIC\_READ, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

}

else

if (portHandle == INVALID\_HANDLE\_VALUE) {

std::wcout << "Failed to open the port " << portName << std::endl;

exit(-1);

}

}

~Port() {

if (portHandle != INVALID\_HANDLE\_VALUE)

CloseHandle(portHandle);

}

HANDLE getPortHandle() {

return portHandle;

}

LPCTSTR getportName() {

return portName;

}

void setPortParams() {

DCB portParams = { 0 };

portParams.DCBlength = sizeof(portParams);

if (!GetCommState(portHandle, &portParams)) {

std::wcout << "Failed to get state for port " << portName << std::endl;

exit(-1);

}

portParams.BaudRate = CBR\_9600; //скорость передачи в бит\с

portParams.ByteSize = 8; //размер байта

portParams.StopBits = ONESTOPBIT; //количество стоповых битов

portParams.Parity = NOPARITY; //использование паритета

if (!SetCommState(portHandle, &portParams)) {

std::wcout << "Failed to set state for port " << portName << std::endl;

exit(-1);

}

}

void readCOM() {

DWORD size = 0;

char recievedChar;

while (true) {

if (ReadFile(portHandle, &recievedChar, 1, &size, nullptr)) {

if (size > 0) {

std::cout << "Recieved char is: " << recievedChar << std::endl;

break;

}

else {

std::cout << "Nothing to read" << std::endl;

}

}

else {

std::cout << "Failed to read port" << std::endl;

}

}

}

};

int main() {

Port port1(L"COM1", WRITE);

Port port2(L"COM2", READ);

port1.setPortParams();

port2.setPortParams();

char data = 'A';

DWORD size = sizeof(data);

DWORD bytesWritten;

BOOL Ret = WriteFile(port1.getPortHandle(), &data, size, &bytesWritten, NULL);

std::cout << size << " Bytes in char. " << bytesWritten << " Bytes sended. " << std::endl;

port2.readCOM();

return 0;

}

* 1. Листинг программы, использующей BIOS прерывание 14h.

.model small

.stack 100h

.data

result db "Char recieved: $"

error\_write db "Writing error!$"

error\_read db "Reading error!$"

.code

output\_string proc

push bp

mov bp, sp ;[bp + 4] string

push ax

push dx

mov ah, 09h

mov dx, offset [bp + 4]

int 21h

pop dx

pop ax

pop bp

ret 2

output\_string endp

init\_com proc

push bp

mov bp, sp ; [bp + 4] com port number

push ax

push dx

xor ax, ax

mov al, 11100011b

mov dx, [bp + 4]

int 14h

pop dx

pop ax

pop bp

ret 2

init\_com endp

write\_COM1 proc

mov ah, 01h

mov dx, 00h

mov al, 'B'

int 14h

test al, 10000000b

jz success\_write

lea dx, error\_write

push dx

call output\_string

jmp exit

success\_write:

ret

write\_COM1 endp

read\_COM2 proc

mov ah, 02h

mov dx, 01h

int 14h

ret

read\_COM2 endp

start:

mov ax, @data

mov ds, ax

mov dx, 0

push dx

call init\_com

mov dx, 1

push dx

call init\_com

call write\_COM1

xor ax, ax

call read\_COM2

lea dx, result

push dx

call output\_string

mov ah, 02h

mov dl, al

int 21h

exit:

mov ax,4C00h

int 21h

end start

3.3. Листинг программы, работающей с COM-портами через регистры как с устройствами ввода-вывода.

#include <stdio.h>

#include <dos.h>

void init\_COM1() {

outp(0x3FB, 0x80);

outp(0x3F8, 0x0C);

outp(0x3F9, 0x00);

outp(0x3FB, 0x03);

}

void init\_COM2() {

outp(0x2FB, 0x80);

outp(0x2F8, 0x0C);

outp(0x2F9, 0x00);

outp(0x2FB, 0x03);

}

void write\_byte(char data) {

while ((inp(0x3FD) & 0x20) == 0);

outp(0x3F8, data);

}

char read\_byte() {

while ((inp(0x2FD) & 0x01) == 0);

return inp(0x2F8);

}

int main() {

init\_COM1();

init\_COM2();

write\_byte('C');

char chr = read\_byte();

printf("Byte recieved: %c", chr);

return 0;

}

1. Тестирование программ

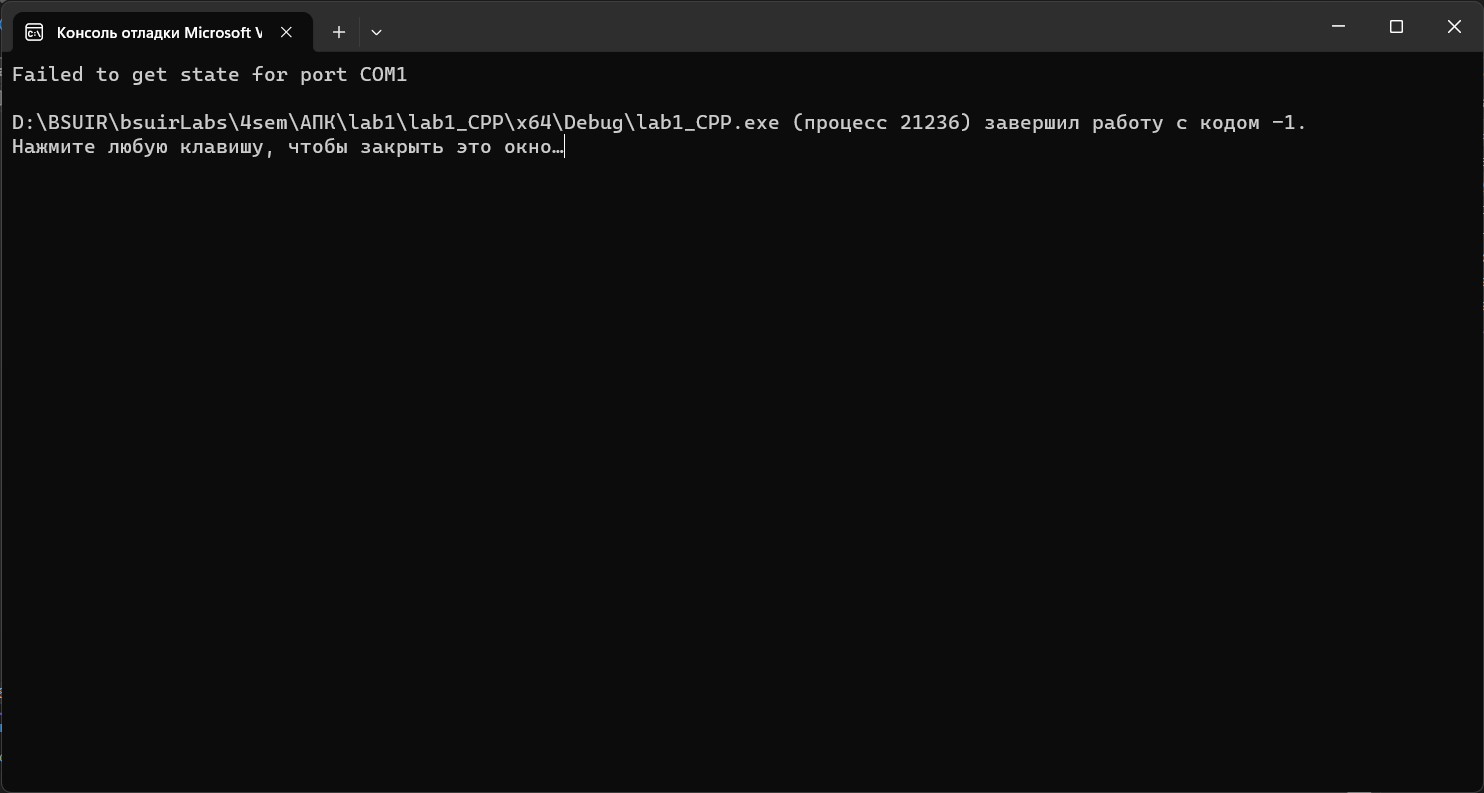


Рисунок 4.1 – Результат работы программы, взаимодействующей с портами ввода-вывода, при выключенной эмуляции COM-портов.

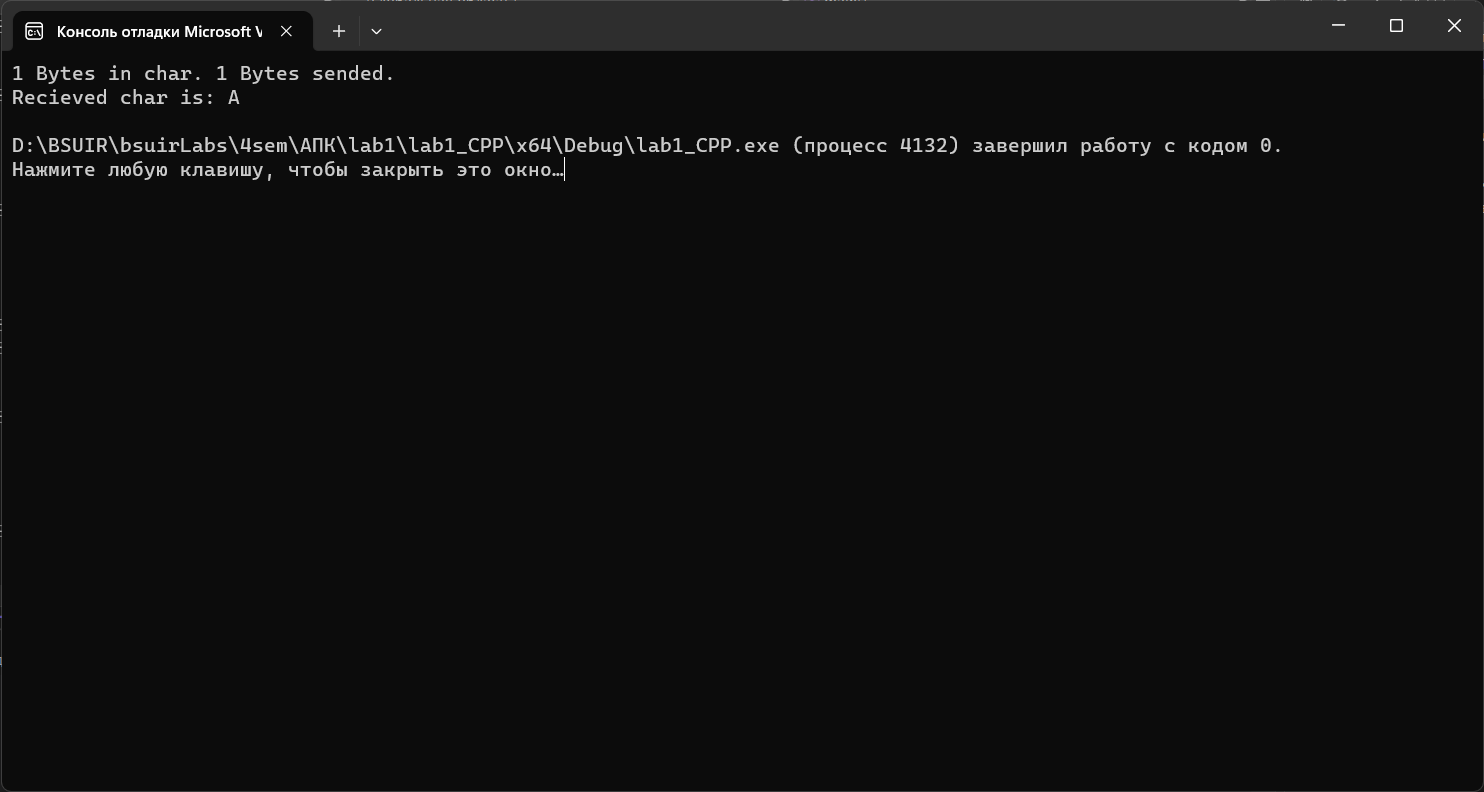


Рисунок 4.2 – Результат работы программы, взаимодействующей с портами ввода-вывода, при включенной эмуляции COM-портов.

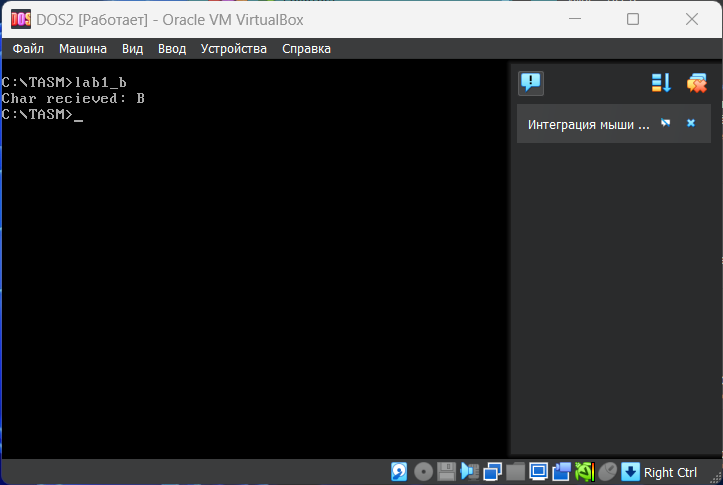


Рисунок 4.3 – Результат работы программы, использующей BIOS прерывание 14h.

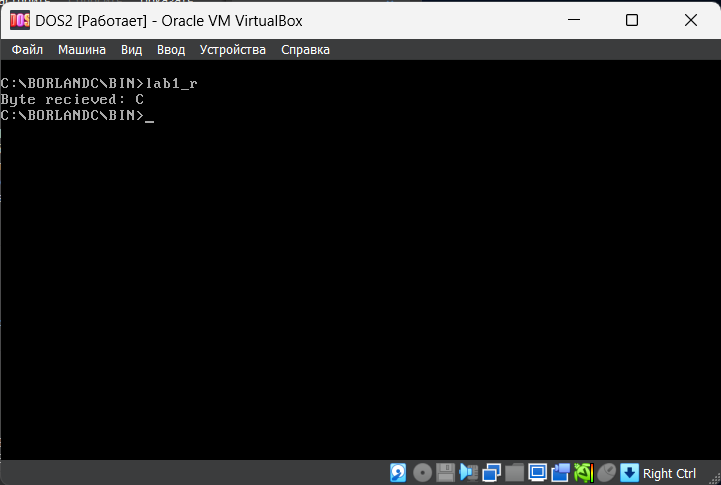


Рисунок 4.4 – Результат работы программы, работающей с COM-портами через регистры как с устройствами ввода-вывода.

1. Заключение

В ходе лабораторной удалось передать 1 байт информации через последовательный порт с использованием различных механизмов.

Для эмуляции COM портов использовался com0com Null-modem emulator, для эмуляции DOS используется Oracle Virtual Box на хосте 64-ех разрядной Windows 11.