Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №1

«Последовательный порт»

Проверил: Выполнил:

к. т. н., доцент студент гр. 150501

Одинец Д. Н. Климович А. Н.

Минск, 2023

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 (write, 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. КОД ПРОГРАММЫ

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

3.1. Код программы, взаимодействующей с портами ввода-вывода.

#include <iostream> //подключение библиотеки потока ввода-вывода

#include <windows.h>

#include "Exception.h"

using namespace std; //использование пространства имён std

LPCTSTR PORT\_NAME\_1 = L"COM1";

LPCTSTR PORT\_NAME\_2 = L"COM2";

#define MAX\_STRING\_LEN 80

class Exception

{

protected:

int code; //код ошибки

char error\_msg[40]; //сообщение об ошибке

public:

Exception(int, const char\*); //конструктор

int getErrorCode(); //получить код ошибки

~Exception(); //деструктор

virtual void what(); //метод вывода ошибки с ее кодом и примечанием

};

Exception::Exception(int newCode = -1, const char\* newMessage = "") //конструктор

{

this->code = newCode;

strcpy\_s(this->error\_msg, strlen(newMessage) + 1, newMessage);

}

Exception::~Exception() //деструктор

{}

int Exception::getErrorCode() //получение кода ошибки

{

return this->code;

}

void Exception::what() //метод вывода ошибки с ее кодом

{

setColor(12);

cout << this->error\_msg << endl;

setColor(6);

cout << "КОД ОШИБКИ: " << this->code << endl;

setColor(7);

}

int main()

{

setlocale(LC\_ALL, "Rus");

HANDLE COM\_Port\_1 = CreateFile(PORT\_NAME\_1, GENERIC\_WRITE, 0, 0, OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, 0); //открытие портов

HANDLE COM\_Port\_2 = CreateFile(PORT\_NAME\_2, GENERIC\_READ, 0, 0, OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, 0);

char inBuff[MAX\_STRING\_LEN + 1] = ""; //буфер отсылаемых данных

char outBuff[MAX\_STRING\_LEN + 1] = ""; //буфер полученных данных

try

{

if (COM\_Port\_1 == INVALID\_HANDLE\_VALUE || COM\_Port\_2 == INVALID\_HANDLE\_VALUE)

throw Exception(-1, "Ошибка открытия порта!");

DCB Parametres = { 0 };

Parametres.DCBlength = sizeof(Parametres);

if (!GetCommState(COM\_Port\_1, &Parametres))

throw Exception(-1, "Ошибка инициализации порта!");

Parametres.BaudRate = CBR\_115200;

Parametres.ByteSize = 8;

Parametres.StopBits = ONESTOPBIT;

Parametres.Parity = NOPARITY;

if (!SetCommState(COM\_Port\_2, &Parametres))

throw Exception(-1, "Ошибка инициализации порта!");

cout << "Введите сообщение (до " << MAX\_STRING\_LEN << " символов):\n";

cin.getline(inBuff, MAX\_STRING\_LEN + 1);

DWORD willWrittenBytes = strlen(inBuff); //сколько будет записано байт информации

DWORD writtenBytes; //сколько удалось записать байт

DWORD readBytes; //число прочитанных байт

if (!WriteFile(COM\_Port\_1, &inBuff, willWrittenBytes, &writtenBytes, NULL))

throw Exception(-3, "Ошибка записи! Не удалось отправить данные.");

cout << "Передано байт: " << writtenBytes << endl;

if (!ReadFile(COM\_Port\_2, &outBuff, strlen(inBuff), &readBytes, 0))

throw Exception(-4, "\nОшибка чтения! Не удалось получить данные.");

cout << "Получено байт: " << readBytes << endl;

cout << "Полученные данные:" << endl << outBuff << endl;

CloseHandle(COM\_Port\_1);

CloseHandle(COM\_Port\_2);

}

catch (Exception& err)

{

err.what();

return err.getErrorCode();

}

return 0;

}

3.2. Код программы, использующей BIOS прерывание 14h.

.model small

.stack 100h

.data

buff\_1 db ? ; buffer for writting

buff\_2 db ? ; buffer for reading

invite\_msg db "Enter a symbol:", 0dh, 0ah, '$'

received\_symbol\_msg db 0dh, 0ah, "Received symbol: $"

error\_write\_msg db 0dh, 0ah, "Write error!",0Dh,0Ah,'$'

error\_read\_msg db 0dh, 0ah, "Read error!",0Dh,0Ah,'$'

.code

jmp start

;=======================================

;MACROS:

;---------------------------------------

PUSH\_ALL MACRO

push ax

push dx

ENDM

POP\_ALL MACRO

pop dx

pop ax

ENDM

PUTS\_STRING MACRO str

PUSH\_ALL

mov dx, offset str

mov ah, 09h

int 21h

POP\_ALL

ENDM

PUTCHAR MACRO symbol

PUSH\_ALL

mov ah, 02h

mov dl, symbol

int 21h

POP\_ALL

ENDM

GETCHAR MACRO symbol

PUSH\_ALL

mov ah, 01h

int 21h

mov symbol, al

POP\_ALL

ENDM

;---------------------------------------

; Functions for ports

;=======================================

INIT\_COM\_PORT proc ;set options

PUSH\_ALL

xor ax,ax ; ah = 0 - function of opening port

mov al, 11100010b ; "111" - 9600 speed,

; "00" - NoParitet,

; "0" - length of stopbit = 1,

; "10" - amount of bits in symbol = 7

mov dx, 0 ; n-1, where n - port number

int 14h

POP\_ALL

endm

;---------------------------------------

; Write/read information

;=======================================

WRITE\_TO\_COM1 PROC

mov al,buff\_1 ; Initialize symbol

mov ah,01h ; ah = 1 - function of writing symbol to the port

mov dx,1 ; Intialize port name(number = 1)

int 14h

test ah, 00001110b ; check on errors

jnz ERROR\_write

ret

ENDP

READ\_FROM\_COM2 proc

PUSH\_ALL

mov ah,02h ; ah = 2 - function of reading symbol from the port

mov dx,2 ; Initialize port name(number = 2)

int 14h

test ah, 00001110b ; check on errors

mov buff\_2, al ; al - symbol was read

POP\_ALL

ret

endp

ERROR\_write proc

PUTS\_STRING error\_read\_msg

call EXIT

ret

endp

ERROR\_read proc

PUTS\_STRING error\_read\_msg

call EXIT

ret

endp

;=======================================

EXIT proc

mov ax,4C00h

int 21h

ret

endp

;---------------------------------------

; MAIN FUNCTION

;=======================================

start:

mov ax, @data

mov ds, ax

PUTS\_STRING invite\_msg

GETCHAR buff\_1

call INIT\_COM\_PORT

call WRITE\_TO\_COM1

call READ\_FROM\_COM2

PUTS\_STRING received\_symbol\_msg

PUTCHAR buff\_2

call EXIT

end start

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

.model small

.stack 100h

.data

buff\_1 db ? ; buffer for writting

buff\_2 db ? ; buffer for reading

invite\_msg db "Enter a symbol:", 0dh, 0ah, '$'

received\_symbol\_msg db 0dh, 0ah, "Received symbol: $"

error\_write\_msg db 0dh, 0ah, "Write error!",0Dh,0Ah,'$'

error\_read\_msg db 0dh, 0ah, "Read error!",0Dh,0Ah,'$'

.code

;=======================================

;MACROS:

;---------------------------------------

PUSH\_ALL MACRO

push ax

push dx

ENDM

POP\_ALL MACRO

pop dx

pop ax

ENDM

PUTS\_STRING MACRO str

PUSH\_ALL

mov dx, offset str

mov ah, 09h

int 21h

POP\_ALL

ENDM

PUTCHAR MACRO symbol

PUSH\_ALL

mov ah, 02h

mov dl, symbol

int 21h

POP\_ALL

ENDM

GETCHAR MACRO symbol

PUSH\_ALL

mov ah, 01h

int 21h

mov symbol, al

POP\_ALL

ENDM

;---------------------------------------

; Functions for ports

;=======================================

INIT\_COM\_PORT proc

PUSH\_ALL

mov al,10000000b ; '10' - length of word = 7 bits

; '0' - amount of stopbits = 1

; '00' - NoParitet

; '0' - fraquency

; '0' - BREAK

; '0' - usual using(for writing)

mov dx,3FBh

out dx,al ; Set options

mov dx,3F8h ; Port is used for writing

mov al,00000000b

out dx,al

mov al,11100000b ; Permission for interrupts for ready giving data

mov dx,3F9h

out dx,al

POP\_ALL

ret

endp

WRITE\_TO\_PORT proc

PUSH\_ALL

xor al,al

mov dx,3FDh ; port is ready for write

in al,dx

test al,00100000b ; register is empty, u can write some information to

jnz ERROR\_write

mov dx,3F8h ; writing

mov al,buff\_1

out dx,al ; load data in COM1

POP\_ALL

ret

endp

READ\_FROM\_PORT proc

PUSH\_ALL

xor al,al ; zero for reading

mov dx,3FDh

in al,dx

test al,00001110b ; check on errors

jnz ERROR\_read

mov dx,3F8h ; read

in al,dx

mov buff\_2,al ; load data from COM1

POP\_ALL

ret

endp

ERROR\_write proc

PUTS\_STRING error\_read\_msg

call EXIT

ret

endp

ERROR\_read proc

PUTS\_STRING error\_read\_msg

call EXIT

ret

endp

;=======================================

EXIT proc

mov ax,4C00h

int 21h

ret

endp

;---------------------------------------

; MAIN FUNCTION

;=======================================

start:

mov ax,@data

mov ds,ax

PUTS\_STRING invite\_msg

GETCHAR buff\_1

call INIT\_COM\_PORT

call WRITE\_TO\_PORT

call READ\_FROM\_PORT

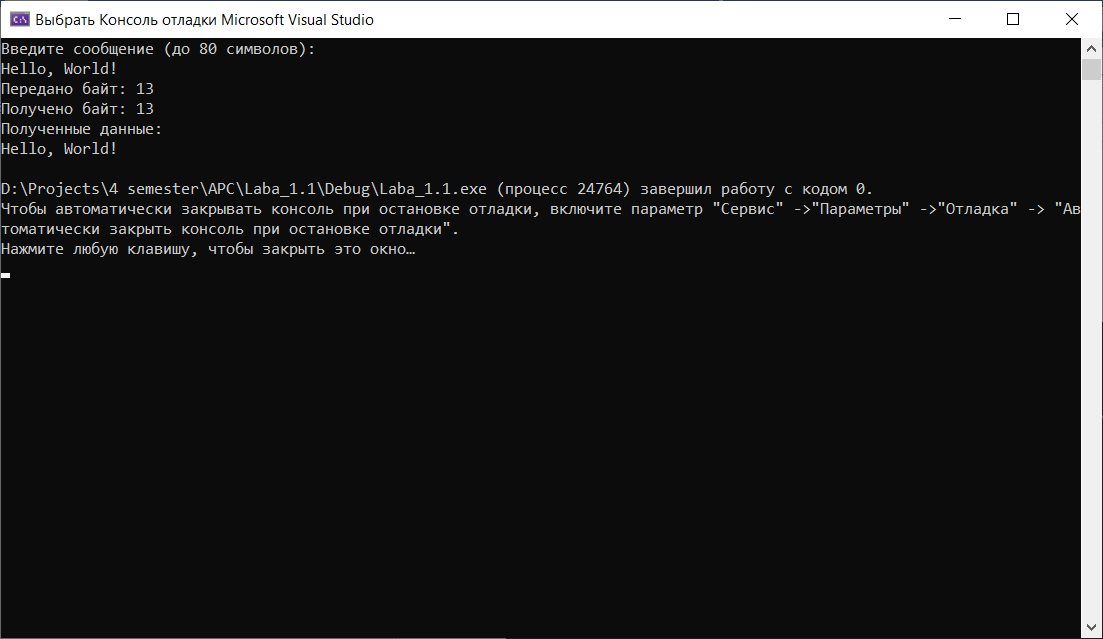
PUTS\_STRING received\_symbol\_msg

PUTCHAR buff\_2

call EXIT

end start

1. ТЕСТИРОВАНИЕ



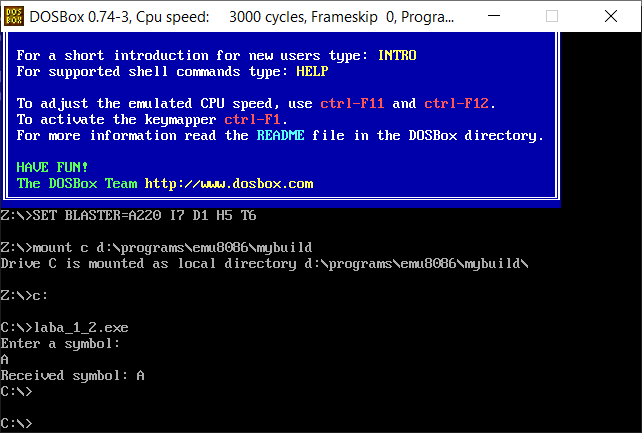
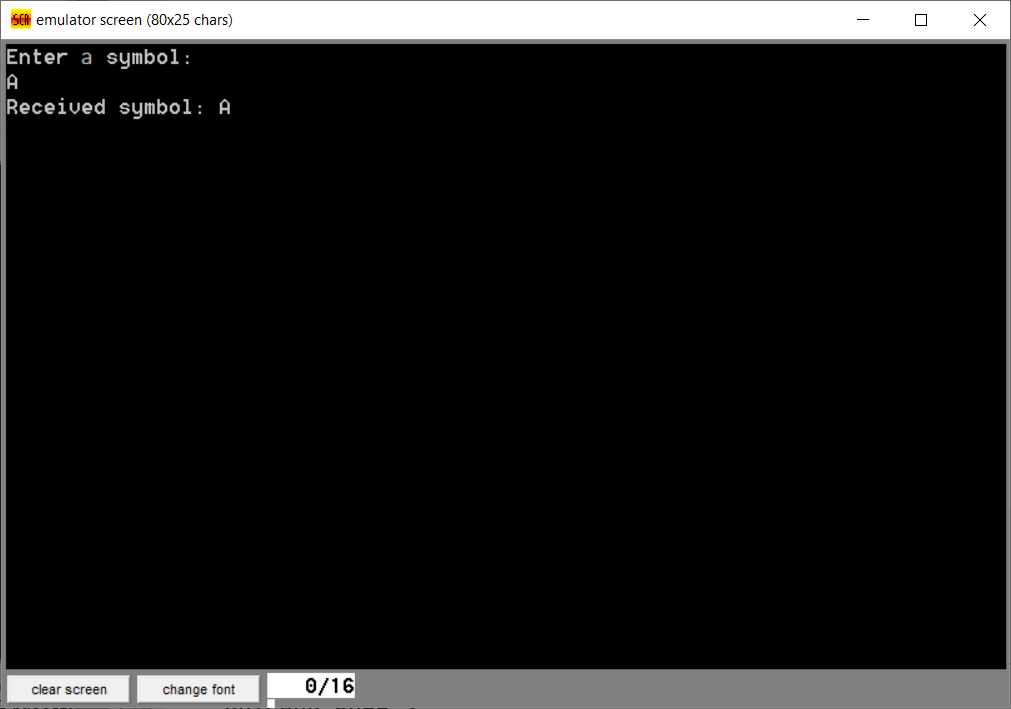
Рисунок 4.1 – – Результат работы программы, взаимодействующей с портами ввода-вывода.

Рисунок 4.2 – Результат работы программы, использующей BIOS прерывание 14h.

Рисунок 4.3 – Результат работы программы, работающей с COM-портами через регистры как с устройствами ввода-вывода.

1. ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был разработан программный модель для передачи(приёма) байта информации через последовательный порт с использованием различных механизмов.

В данной программе передавался один байт информации. В дальнейшем программу можно улучшить, расширив буфер передаваемой информации.

Для эмуляции COM портов использовался Virtual Serial Port Driver, для эмуляции DOS использовался DOSBox 0.74-3 на хосте 64-ех разрядной Windows 10.