Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №1

“ Последовательный порт”

Проверил: Выполнил:

к.т.н., доцент студент гр.250502

Одинец Дмитрий Николаевич Бригадир А. С.

Минск 2024

**Задача**

Целью данной работы является разработать программный модуль реализации процедуры передачи (приема) 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 устройством ввода-вывода.

**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

**Листинг программы**

**1.**

#include <stdio.h>

#include <windows.h>

void setup\_serial\_port(HANDLE serial\_port)

{

DCB dcb\_serial\_params = {0}; // Создание структуры DCB и инициализация нулевыми значениями

dcb\_serial\_params.DCBlength = sizeof(dcb\_serial\_params); // Установка размера структуры

if (!GetCommState(serial\_port, &dcb\_serial\_params)) // Получение текущих параметров порта и сохранение их в структуре

{

printf("ERROR GET STATE.\n");

}

dcb\_serial\_params.BaudRate = CBR\_9600; // Установка скорости передачи

dcb\_serial\_params.ByteSize = 8; // Установка количества байт

dcb\_serial\_params.StopBits = ONESTOPBIT; // Установка стопового бита

dcb\_serial\_params.Parity = NOPARITY; // Установка проверки четности

if (!SetCommState(serial\_port, &dcb\_serial\_params)) // Инициализация порта с новыми параметрами

{

printf("ERROR SETTING STATE.\n");

}

if (serial\_port == INVALID\_HANDLE\_VALUE) // Проверка успешного открытия порта

{

printf("OPEN ERROR.\n");

ExitProcess(1);

}

}

int main()

{

unsigned char data\_buffer[] = "HELLO "; // Буфер данных для отправки

unsigned char received\_data\_buffer[8] = {0}; // Буфер для принятых данных (размер 8 байт, обнулен)

unsigned long buffer\_size = sizeof(data\_buffer); // Размер данных для отправки

unsigned long received\_data\_size = sizeof(received\_data\_buffer); // Размер принятых данных

HANDLE serial\_port\_second = CreateFile("\\\\.\\COM2", GENERIC\_READ, 0, NULL, OPEN\_EXISTING, 0, NULL); // Открытие COM-порта COM2 для чтения

setup\_serial\_port(serial\_port\_second); // Инициализация параметров порта

HANDLE serial\_port\_first = CreateFile("\\\\.\\COM1", GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, NULL); // Открытие COM-порта COM1 для записи

setup\_serial\_port(serial\_port\_first); // Инициализация параметров порта

printf("COM1 OPENED\n");

if (WriteFile(serial\_port\_first, data\_buffer, buffer\_size, &buffer\_size, 0)) // Запись данных в COM1

printf("DATA SENT\n"); // Вывод сообщения об успешной отправке данных

printf("\nCOM2 OPENED\n");

ReadFile(serial\_port\_second, received\_data\_buffer, received\_data\_size, &received\_data\_size, 0); // Чтение данных из COM2

printf("RECEIVED DATA: %s\n", received\_data\_buffer); // Вывод информации о принятых данных

getchar(); // Ожидание ввода от пользователя

CloseHandle(serial\_port\_first); // Закрытие портов

CloseHandle(serial\_port\_second);

return 0;

}

**2.**

.model small

.stack 100h

.data

Endl db 0Dh,0Ah,"$"

Error\_Write db "Write error!",0Dh,0Ah,'$'

Error\_Read db "Read error!",0Dh,0Ah,'$'

mas db 0

.code

jmp start

Input proc

mov ah, 01h

int 21h

mov mas, al

mov ah, 09h

lea dx, Endl

int 21h

ret

Input endp

Init\_COM1 proc

xor ax,ax

mov al,A3h

mov dx,0h

int 14h

ret

Init\_COM1 endp

IsWrite\_COM1 proc

mov al, mas

mov ah,1h

mov dx,0h

int 14h

test al,80h

jnz NoWRite

ret

IsWrite\_COM1 endp

NoWRite proc

mov ah,9h

lea dx, Error\_Write

add dx,2h

int 21h

ret

NoWRite endp

IsRead\_COM2 proc

mov ah,2h

mov dx,1h

int 14h

test al,80h

jnz NoRead

ret

IsRead\_COM2 endp

NoRead proc

mov ah,9h

lea dx, Error\_Read

add dx,2h

int 21h

ret

NoRead endp

Output proc

mov ah,02h

mov dl,al

int 21h

ret

Output endp

Exit proc

mov ax,4C00h

int 21h

ret

Exit endp

error:

call Exit

start:

mov ax, @data

mov ds, ax

call Input

call Init\_COM1

call IsWrite\_COM1

call IsRead\_COM2

call Output

call Exit

end start

**3.**

#include <windows.h>

#include <conio.h>

#include <stdio.h>

#include <dos.h>

#include <stdlib.h>

#include <math.h>

#include <malloc.h>

int Com\_Init(int port, unsigned long baud);

void com\_outchar(char chr, int base\_port);

char com\_inchar(int base\_port);

int check\_snd(int base\_port);

int Com\_Init(int port, unsigned long baud)

{

// char base\_port = 0x3f8 - 0x100 \* port;

char base\_port = port;

unsigned int div;

switch (baud)

{

case 110:

div = 1040;

break;

case 150:

div = 768;

break;

case 300:

div = 384;

break;

case 600:

div = 192;

break;

case 1200:

div = 96;

break;

case 2400:

div = 48;

break;

case 4800:

div = 24;

break;

case 9600:

div = 12;

break;

case 19200:

div = 6;

break;

case 38400:

div = 3;

break;

case 57600:

div = 2;

break;

case 115200:

div = 1;

break;

default:

return 0;

}

unsigned int regst;

regst = inp(base\_port + 0x03);

outp(base\_port + 0x03, regst | 0x80);

outp(base\_port + 0x01, (div >> 8) & 0x00ff); // DLM

outp(base\_port, div & 0x00ff); // DLL

// DLAB LCR

outp(base\_port + 0x03, regst & 0x7f);

outp(base\_port + 0x01, 0x00);

outp(base\_port + 0x03, 29);

outp(base\_port + 0x04, 0x00); // DTR=0 RTS=0

return 1;

};

void com\_outchar(char chr, int base\_port)

{

char regst = inp(base\_port + 0x04);

outp(base\_port + 0x04, regst | 0x02); // RTS

outp(base\_port, chr);

delay(100);

while (!check\_snd(base\_port))

;

outp(base\_port + 0x04, regst & 0xfd); // RTS

}

char com\_inchar(int base\_port)

{

return inp(base\_port);

}

int check\_snd(int base\_port)

{

unsigned char regst;

regst = inp(base\_port + 0x05);

return ((regst & 0x20) >> 5) == 0x01;

}

int main()

{

int port1 = 0x3f8;

int port2 = 0x2f8;

char inf;

if (!Com\_Init(port1, 9600))

puts("(1) ERROR COM1");

if (!Com\_Init(port2, 9600))

puts("(1) ERROR COM2");

printf("oldInfo(COM1) : %c\n", com\_inchar(port1));

printf("oldInfo(COM2) : %c\n", com\_inchar(port2));

if (check\_snd(port1) == 1)

puts("ready");

else

puts("not ready");

puts("Enter char to sending:");

rewind(stdin);

inf = getchar();

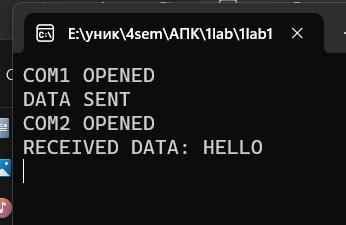
com\_outchar(inf, port1);

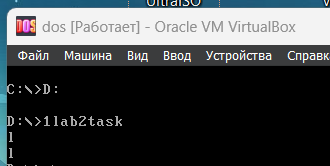
printf("info(COM1) : %c\n", com\_inchar(port1));

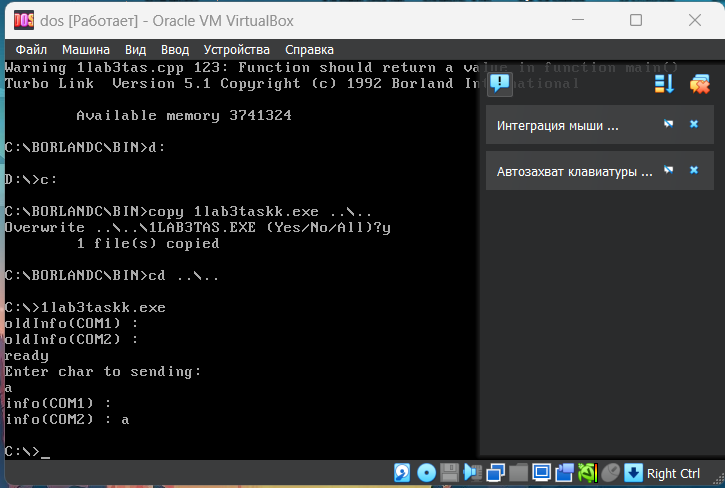
printf("info(COM2) : %c\n", com\_inchar(port2));

}

**Тест**



****

****

**Заключение**

В данной лабораторной работе разработана программа, которая позволяет записывать в данные в один COM-порт и читать их из другого COM-порта (в данном случае это пара COM1 – COM2).