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 Грибовская А. А.

Минск 2024

**Задача**

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

Работа с COM-портом через регистры как с устройствами 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, взаимодействующей с портами ввода-вывода

#include <windows.h> // Подключение заголовочного файла windows.h для использования функций работы с COM-портами в Windows.

#include <iostream> // Подключение заголовочного файла iostream для использования стандартных потоков ввода-вывода.

using namespace std; // Использование пространства имен std.

HANDLE COM\_Port\_1; // Объявление переменной типа HANDLE для первого COM-порта.

LPCWSTR Port\_Name\_1 = L"COM3"; // Объявление переменной типа LPCWSTR (указатель на строку Unicode) и присвоение значения имени первого COM-порта.

HANDLE COM\_Port\_2; // Объявление переменной типа HANDLE для второго COM-порта.

LPCWSTR Port\_Name\_2 = L"COM4"; // Объявление переменной типа LPCWSTR и присвоение значения имени второго COM-порта.

void Read\_from\_COM() // Объявление функции Read\_from\_COM(), которая считывает данные с COM-порта.

{

DWORD Size;

char Received\_Char;

ReadFile(COM\_Port\_2, &Received\_Char, 1, &Size, 0); // Чтение 1 байта данных с COM-порта 4.

if (Size > 0) // Если количество считанных байт больше нуля.

{

cout << Received\_Char; // Вывод прочитанного символа на консоль.

}

}

int main() // Объявление главной функции main().

{

COM\_Port\_1 = ::CreateFileW(Port\_Name\_1, GENERIC\_WRITE, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0); // Открытие COM-порта 1 для записи.

COM\_Port\_2 = ::CreateFileW(Port\_Name\_2, GENERIC\_READ, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0); // Открытие COM-порта 2 для чтения.

if (COM\_Port\_1 == INVALID\_HANDLE\_VALUE||COM\_Port\_2 == INVALID\_HANDLE\_VALUE) // Если один из COM-портов не удалось открыть.

{

if (GetLastError() == ERROR\_FILE\_NOT\_FOUND) // Если последняя ошибка – файл не найден.

{

cout << "COM-port does not exist!\n"; // Вывод сообщения о том, что COM-порт не существует.

}

cout << "Some other error.\n"; // Вывод сообщения о другой ошибке.

return 1; // Завершение программы с кодом ошибки 1.

}

DCB Serial\_Params = { 0 }; // Объявление структуры DCB для установки параметров COM-порта.

Serial\_Params.DCBlength = sizeof(Serial\_Params); // Установка размера структуры.

if (!GetCommState(COM\_Port\_1, &Serial\_Params) || !GetCommState(COM\_Port\_2, &Serial\_Params)) // Получение параметров COM-портов.

{

cout << "Getting state error.\n"; // Вывод сообщения об ошибке получения состояния COM-портов.

return 1; // Завершение программы с кодом ошибки 1.

}

Serial\_Params.BaudRate = CBR\_9600; // Установка скорости передачи данных.

Serial\_Params.ByteSize = 8; // Установка размера данных в байтах.

Serial\_Params.StopBits = ONESTOPBIT; // Установка количества стоп-битов.

Serial\_Params.Parity = NOPARITY; // Установка типа проверки четности.

if (!SetCommState(COM\_Port\_1, &Serial\_Params) || !SetCommState(COM\_Port\_2, &Serial\_Params)) // Установка параметров COM-портов.

{

cout << "Error setting serial port state.\n"; // Вывод сообщения об ошибке установки состояния COM-портов.

return 1; // Завершение программы с кодом ошибки 1.

}

char data; // Объявление переменной типа char для хранения отправляемого символа.

cout << "Enter a character to send: "; // Вывод запроса на ввод символа.

cin >> data; // Считывание введенного символа.

DWORD Size = sizeof(data); // Определение размера отправляемых данных.

DWORD Bytes\_Written; // Объявление переменной для хранения количества записанных байт.

BOOL Ret = WriteFile(COM\_Port\_1, &data, Size, &Bytes\_Written, NULL); // Запись данных в COM-порт 1.

cout << Size << " Bytes in string. " << Bytes\_Written << " Bytes sended. " << endl; // Вывод сообщения о количестве отправленных байт.

Read\_from\_COM(); // Вызов функции для чтения данных с COM-порта 2.

CloseHandle(COM\_Port\_1); // Закрытие COM-порта 1.

CloseHandle(COM\_Port\_2); // Закрытие COM-порта 2.

return 0; // Завершение программы без ошибок.

}

Листинг программы, использующей BIOS прерывание 14h

.model small

.stack 100h

.data

buffer db ? ; Объявление переменной buffer типа байт (db) для хранения символа.

message db 10, 13, "Enter a character: $" ; Объявление строки message для вывода запроса ввода символа.

received\_message db 10, 13, "Received character: $" ; Объявление строки received\_message для вывода сообщения о полученном символе.

.code

main:

mov ax, @data ; Перемещение адреса сегмента данных в регистр ax.

mov ds, ax ; Перемещение адреса сегмента данных в регистр ds.

mov ah, 9 ; Загрузка функции для вывода строки с прерыванием 21h.

lea dx, message ; Загрузка адреса строки message в регистр dx.

int 21h ; Вызов прерывания для вывода сообщения.

mov ah, 1 ; Загрузка функции для чтения символа с прерыванием 21h.

int 21h ; Вызов прерывания для чтения символа в регистр al.

mov buffer, al ; Сохранение прочитанного символа в переменной buffer.

mov ah, 0 ; Загрузка нулевой функции для передачи символа с прерыванием 14h.

mov dx, 0 ; Загрузка номера COM-порта (COM1) в регистр dx.

mov al, buffer ; Загрузка символа из buffer в регистр al.

int 14h ; Вызов прерывания для передачи символа в COM1. (Прочитать управляющий порт модема. Вход: DX = номер порта (0-3). Выход: BL - Значение в управляющем порту модема)

mov ah, 9 ; Загрузка функции для вывода строки с прерыванием 21h.

lea dx, received\_message ; Загрузка адреса строки received\_message в регистр dx.

int 21h ; Вызов прерывания для вывода сообщения о полученном символе.

mov ah, 0 ; Загрузка нулевой функции для передачи символа с прерыванием 14h.

mov dx, 1 ; Загрузка номера COM-порта (COM2) в регистр dx.

int 14h ; Вызов прерывания для передачи символа в COM2.

(Прочитать управляющий порт модема. Вход: DX = номер порта (0-3). Выход: BL - Значение в управляющем порту модема)

xor ah,ah

mov dl, bl ; Загрузка символа из bl(COM2) в регистр dl.

mov ah, 2 ; Загрузка функции для вывода символа на экран с прерыванием 21h.

int 21h ; Вызов прерывания для вывода символа на экран.

mov ah, 4ch ; Загрузка функции для завершения программы с прерыванием 21h.

int 21h ; Вызов прерывания для завершения программы.

end main ; Конец программы.

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

1. На языке Си:

#include <dos.h> // Подключение заголовочного файла dos.h для работы с операциями ввода-вывода.

#include <stdio.h> // Подключение заголовочного файла stdio.h для стандартных операций ввода-вывода.

#define COM1\_PORT 0x3F8 // Определение порта для COM1.

#define COM2\_PORT 0x2F8 // Определение порта для COM2.

void init\_serial(int port)

{

outp(port + 1, 0x00); // Отключение всех прерываний. Это устанавливает регистр IER (Interrupt Enable Register) на 0x00, что отключает все прерывания.

outp(port + 3, 0x80); // Включение DLAB (установить делитель скорости передачи). Это устанавливает бит DLAB (Divisor Latch Access Bit) в регистре LCR (Line Control Register) на 1, что позволяет устанавливать делитель скорости.

outp(port + 0, 0x03); // Установить делитель на 3 (младший байт) для скорости 38400 бод. Это устанавливает младший байт (LSB) и старший байт (MSB) делителя скорости (базовая частота / (делитель \* 16) = скорость передачи).

outp(port + 1, 0x00); // (старший байт).

outp(port + 3, 0x03); // 8 бит, без контроля четности, один стоп-бит.

outp(port + 2, 0xC7); // Включить FIFO, очистить их, установить порог в 14 байт. Это устанавливает регистр FCR (FIFO Control Register) таким образом, что FIFO (First In, First Out) включены, они очищены и порог установлен на 14 байт.

outp(port + 4, 0x0B); // Разрешить прерывания, установить RTS/DSR. Это устанавливает биты для разрешения прерываний и управления сигналами RTS (Request To Send) и DSR (Data Set Ready).

}

int is\_transmit\_empty(int port)

{

return inp(port + 5) & 0x20;

}

void write\_serial(int port, char a) {

while (is\_transmit\_empty(port) == 0); // Ждать, пока буфер отправки не станет пустым.

outp(port, a); // Отправить символ.

}

int is\_data\_ready(int port) {

return inp(port + 5) & 1;

}

char read\_serial(int port) {

while (is\_data\_ready(port) == 0); // Ждать, пока данные не будут доступны.

return inp(port); // Прочитать символ из порта.

}

int main() {

char data;

init\_serial(COM1\_PORT); // Инициализация порта COM1.

init\_serial(COM2\_PORT); // Инициализация порта COM2.

printf("Enter a character: "); // Вывод запроса на ввод символа.

scanf("%c", &data); // Получение символа из ввода пользователя.

write\_serial(COM1\_PORT, data); // Отправка символа через COM1.

printf("Sent character: %c\n", data); // Вывод отправленного символа.

printf("Received character: %c\n", read\_serial(COM2\_PORT)); // Вывод принятого символа через COM2.

return 0; // Завершение программы.

}

1. На языке Ассемблер:

.model small ; Определение модели памяти small.

.stack 100h ; Определение размера стека.

.data ; Начало секции данных.

Error\_Write db "Write error!",0Dh,0Ah,'$' ; Объявление строки для сообщения об ошибке записи.

Error\_Read db "Read error!",0Dh,0Ah,'$' ; Объявление строки для сообщения об ошибке чтения.

Information db 0Dh,0Ah,"Byte sent: $" ; Объявление строки для информации о переданном байте.

Data\_Byte db ? ; Объявление переменной для хранения передаваемого байта.

Data\_Byte2 db ? ; Объявление переменной для хранения принятого байта.

.code ; Начало секции кода.

Init\_COM1 proc

mov al,80h ; Подготовка регистра AL для установки регистра IER.

mov dx,3FBh ; Порт для установки регистра IER.

out dx,al ; Установка регистра IER.

mov dx,3F8h ; Порт COM1.

mov al,00h ; Нулевой байт (будем отправлять данные).

out dx,al ; Отправка нулевого байта.

mov al,0Ch ; Значение для регистра LCR. Параметр чётности используется для обнаружения ошибок при передаче данных по последовательному порту. Приемник может проверить бит чётности и определить, были ли ошибки во время передачи данных.

mov dx,3F9h ; Порт для регистра LCR.

out dx,al ; Установка регистра LCR.

mov dx,3FCh ; Порт COM1 для FCR.

mov al,00001011b ; Значение для FCR.

out dx,al ; Установка FCR.

mov dx,3F9h ; Порт COM1 для MCR.

mov al,0 ; Нулевой байт для MCR.

out dx,al ; Установка MCR.

ret ; Возврат из процедуры.

Init\_COM1 endp

// Последующие процедуры IsWrite\_COM1, NoWrite, IsRead\_COM2, NoRead следует проанализировать и описать по аналогии с вышеприведенным комментарием.

start:

mov ax,@data ; Загрузка базового адреса данных в регистр AX.

mov ds,ax ; Установка сегмента данных равного базовому адресу данных.

call Init\_COM1 ; Вызов процедуры инициализации порта COM1.

call IsWrite\_COM1 ; Проверка, можно ли писать в COM1.

call Send\_Byte ; Отправка байта через COM1.

mov al,2 ; Загрузка значения 2 в AL.

call IsRead\_COM2 ; Проверка, можно ли читать из COM2.

call Read\_Byte ; Чтение байта из COM2.

mov dx,offset Information ; Загрузка адреса Information в DX.

mov ah,9 ; Загрузка 9 в AH (прерывание для вывода строки).

int 21h ; Вызов прерывания для вывода строки.

mov ah,02h ; Загрузка 2 в AH (прерывание для вывода символа).

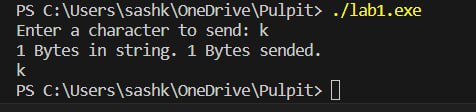
mov dl,Data\_Byte2 ; Загрузка прочитанного байта в DL.

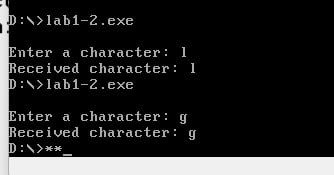
int 21h ; Вызов прерывания для вывода символа.

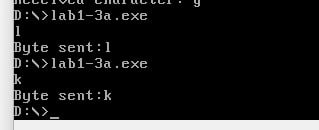
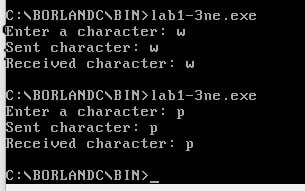
call Exit ; Выход из программы.

end start ; Конец программы.

Тест





Заключение

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

Для эмуляции COM портов использовался Virtual Serial Port Driver 9, для эмуляции DOS используется MS-DOS на хосте 64-ех разрядной Windows 10.