Министерство образования Республики Беларусь
Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №1
“ Последовательный порт”

Проверил:
к.т.н., доцент
Одинец Дмитрий Николаевич

Выполнил:
студент гр.150501
Кипятков В. И.

Минск 2023

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. 1 прямое взаимодействие с портами ввода-вывода (write, read)
2. 2 использование BIOS прерывания 14h,
3. 3 работа с COM-портом через регистры как с устройством ввода-вывода.

Алгоритм

Каждая программа имеет одинаковый алгоритм :

- инициализация порта
- передача данных
- приём данных
- анализ состояния порта.

Таким образом при прямом взаимодействии с портами используются системные функции `inp()` и `outp()` из библиотеки `<conio.h>`. Работая с портами как с устройствами ввода вывода, используются системные функции работы с файлами `WriteFile()` и `ReadFile()` из библиотеки `<windows.h>`. И третий способ использует BIOS прерывание 14h.

Листинг программы

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>
//#include <string>

//using namespace std;
int Com_Init(int port, unsigned long baud);
void com_outchar(char chr, int base_port);
char com_inchar(int base_port);
int check_rcv(int base_port);
int check_snd(int base_port);
int com_RTS(int rts, int base_port);


int Com_Init(int port, unsigned long baud)
{
    /* Определяем значения базового порта ввода-вывода для
    соответствующего интерфейса*/
    //char base_port = 0x3f8 - 0x100 * port;
    char base_port = port;
    /* Определяем значения константы делителя частоты
    (см. формулу 1.1) */
```

```
    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;
    //устанавливаем бит DLAB регистра LCR
    regst = inp(base_port + 0x03);
    outp(base_port + 0x03, regst | 0x80);
    //записываем значение делителя частоты
    outp(base_port + 0x01, (div >> 8) & 0x00ff); // DLM
    outp(base_port, div & 0x00ff); // DLL
    //сбрасываем бит DLAB регистра LCR
    outp(base_port + 0x03, regst & 0x7f);
    // отключаем прерывания
    outp(base_port + 0x01, 0x00);
    // настраиваем линию
    // проверка паритета на четность
    // 1 стоп-бит
    // размер байта - 8 бит
    outp(base_port + 0x03, 29);
    // настраиваем регистр управления модемом
    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); // читаем регистр управления
    outp(base_port + 0x04, regst | 0x02); //устанавливаем RTS
    outp(base_port, chr); // запись байта
    delay(100);
    while (!check_snd(base_port)); //ждем готовности передатчика
    outp(base_port + 0x04, regst & 0xfd); //сбрасываем RTS
}

char com_inchar(int base_port)
{
    return inp(base_port);
}

int check_rcv(int base_port)
{
    unsigned char regst;
    regst = inp(base_port + 0x05);
    return (regst & 0x01 == 0x01);
}

int check_snd(int base_port)
```

```
{
    unsigned char regst;
    regst = inp(base_port + 0x05);
    return ((regst & 0x20) >> 5) == 0x01;
}

int com_RTS(int rts, int base_port)
{
    unsigned char regst;
    regst = inp(base_port + 0x04);
    if (rts == 0)
      outp(base_port + 0x04, regst & 0xfd); //сброс
    else if (rts == 1)
      outp(base_port + 0x04, regst | 0x2); //установка
    else return 0;
    return 1;
}

int main()
{
    int port1 = 0x3f8;
    int port2 = 0x2f8;
    char inf;

    if (!Com_Init(port1, 9600)) puts("(1)some ERROR");
    if (!Com_Init(port2, 9600)) puts("(1)some ERROR");

    printf("pre_inf(1) : %c\n", com_inchar(port2));

    if (check_snd(port1) == 1) puts("ready");
      else puts("not ready");

    puts("Enter some char to snd:");
    rewind(stdin);
    inf = getchar();

    com_outchar(inf, port1);
    printf("inf(2) : %c\n", com_inchar(port2));
}
```

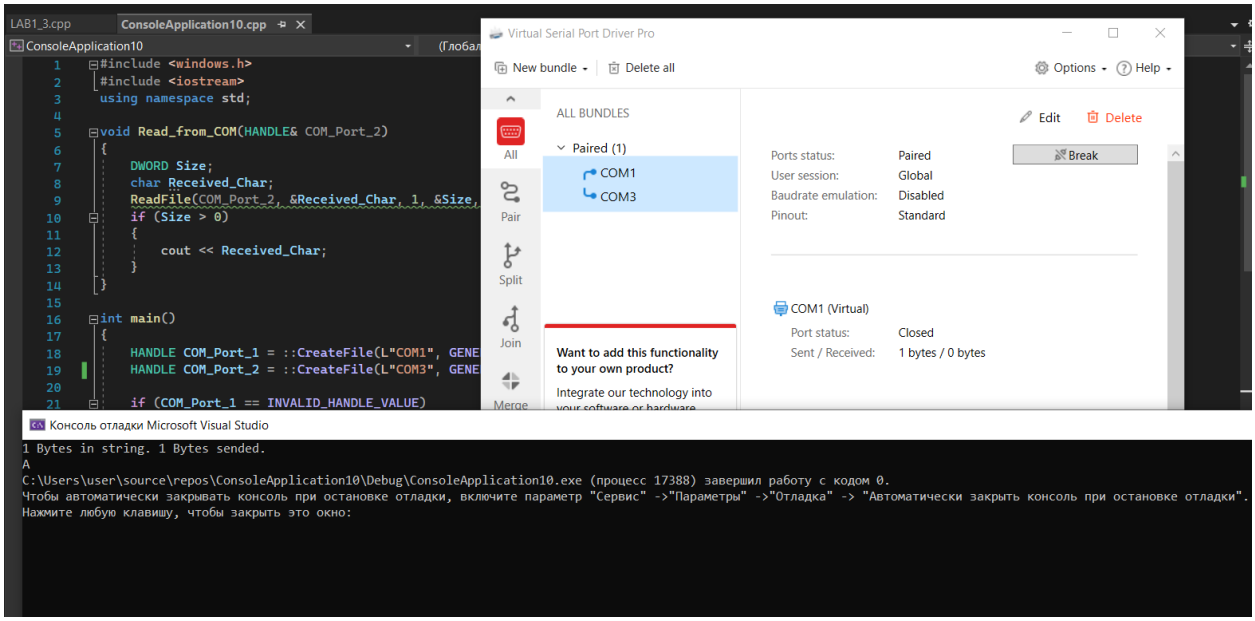
```cpp
#include <windows.h>
#include <iostream>
using namespace std;

void Read_from_COM(HANDLE& COM_Port_2)
{
    DWORD Size;
    char Received_Char;
    ReadFile(COM_Port_2, &Received_Char, 1, &Size, 0);
    if (Size > 0)
    {
        cout << Received_Char;
    }
}

int main()
{
    HANDLE COM_Port_1 = ::CreateFile(L"COM1", GENERIC_WRITE, 0, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
    HANDLE COM_Port_2 = ::CreateFile(L"COM3", GENERIC_READ, 0, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

    if (COM_Port_1 == INVALID_HANDLE_VALUE)
    {
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
        {
            cout << "COM-port does not exist!\n";
        }
        cout << "Some other error.\n";
    }

    DCB Serial_Params = { 0 };
    Serial_Params.DCBlength = sizeof(Serial_Params);
    if (!GetCommState(COM_Port_1, &Serial_Params))
    {
        cout << "Getting state error.\n";
    }
    Serial_Params.BaudRate = CBR_9600;
    Serial_Params.ByteSize = 8;
    Serial_Params.StopBits = ONESTOPBIT;
    Serial_Params.Parity = NOPARITY;
    if (!SetCommState(COM_Port_2, &Serial_Params))
    {
        cout << "Error setting serial port state.\n";
    }

    char data = 'A';
    DWORD Size = sizeof(data);
    DWORD Bytes_Written;

    BOOL Ret = WriteFile(COM_Port_1, &data, Size, &Bytes_Written, NULL);

    cout << Size << " Bytes in string. " << Bytes_Written << " Bytes sended. "
<< endl;

    Read_from_COM(COM_Port_2);

    return 0;
}
```

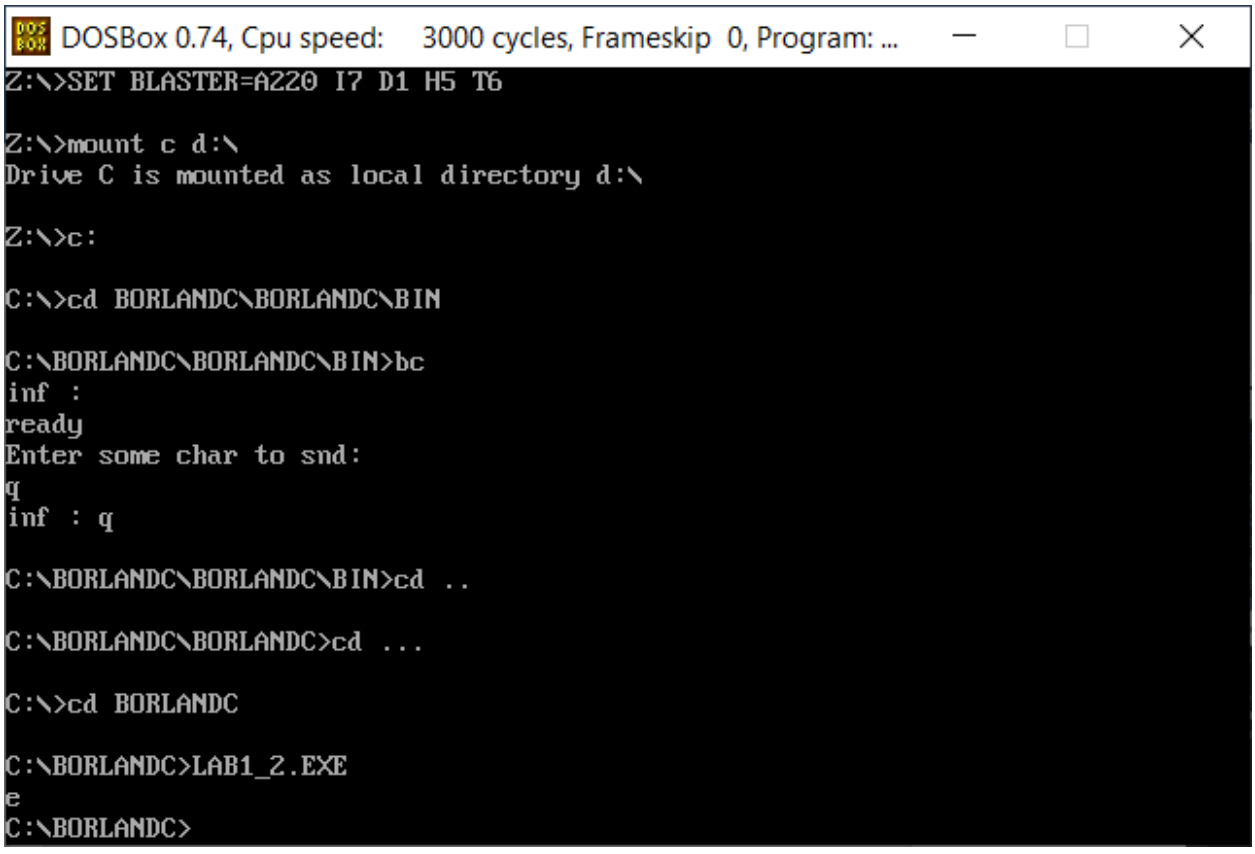
```
.model small
.stack 100h

.data

Error_Write db "Write error!",0Dh,0Ah,'$'
Error_Read db "Read error!",0Dh,0Ah,'$'
Information db "Byte sent: $"

.code

jmp start

;################################################################################
; WORK WITH TRANSMITTED PORT
;################################################################################
Init_COM1 proc
   xor ax,ax                          ; Clear ax register
   mov al,10100011b                   ; Set transfer frequency
   mov dx,0                                 ; Initialize port name
   int 14h
   ret
Init_COM1 endp

IsWrite_COM1 proc
   mov al,'A'                         ; Initialize symbol
   mov ah,1                                 ; Write symbol to the port
   mov dx,0                                 ; Initialize port name
   int 14h
   test al,80h                              ; Test DSR
   jnz NoWRite                              ; If we cant write ... @099
   ret
IsWrite_COM1 endp

; Support function
NoWRite proc
   mov ah,9
   mov dx,offset Error_Write     ;@099 ... - show error message
   add dx,2
   int 21h
   ret
NoWRite endp

;################################################################################
; WORK WITH RECIVED PORT
;################################################################################
IsRead_COM2 proc                      ; Read in the second port
    mov ah,2                          ; Read symbol
    mov dx,1                          ; Initialize port name
    int 14h
    test al,80h                              ; Test RTS
    jnz NoRead                               ; If we cant write ... @099
    ret
IsRead_COM2 endp

NoRead proc
   mov ah,9
   mov dx,offset Error_Read           ;@099 ... - also show error message!
   add dx,2
   int 21h
   ret
NoRead endp
```

```
;################################################################################
; OUTPUT BYTE
;################################################################################
Output proc
   mov ah,02h                          ; Read byte from secong port
   mov dl,al                           ; And show
   int 21h
   ret
Output endp

;################################################################################
; EXIT FUNCTION
;################################################################################
Exit proc
    mov ax,4C00h
    int 21h
    ret
Exit endp

;################################################################################
; MAIN FUNCTION
;################################################################################
start:
   call Init_COM1
   call IsWrite_COM1
   mov al,'e'
   call IsRead_COM2
   ;push ax

   ;mov ah,9
   ;mov dx,offset Information
   ;add dx,2
   ;int 21h

   ;pop ax
   call Output
   call Exit

end start
```

Тест

```
pre_inf(1) :
ready
Enter some char to snd:
D
inf(2) : D
```





Заключение

В данной лабораторной работе разработан программный модуль, который реализует процедуры передачи (приёма) байта информации через последовательный интерфейс.