```
#include <stdio.h>
#include <conio.h>
#include <windows.h>


void initializeComm(HANDLE hCom)
{

/*
Настройка режима работы сом порта осуществляется с помощью структур данных, которые
представляют из себя набор переменных разного типа. Структуры загружаются и читаются с
помощью API функций.
Рассмотрим основные структуры для настройки режимов работы сом порта:

DCB структура

Структура DCB определяет основные настройки COM порта.
В ней содержиться реальная информация из регистров UART.
  typedef struct _DCB {
      DWORD DCBlength;            // длина структуры (DCB)
      DWORD BaudRate;             // скорость в бит/сек
      DWORD fBinary:1;            // бинарный режим
      DWORD fParity:1;            // разрешение контроля четности
      DWORD fOutxCtsFlow:1;       // слежение за CTS
      DWORD fOutxDsrFlow:1;       // слежение за DSR
      DWORD fDtrControl:2;        // режим работы сигнала DTR
      DWORD fDsrSensitivity:1;    // чувствительность к DSR
      DWORD fTXContinueOnXoff:1;  // продолжение передачи при XOFF
      DWORD fOutX:1;              // программное управление потоком при передачи
(XON/XOFF)
      DWORD fInX:1;               // программное управление потоком при приеме
(XON/XOFF)
      DWORD fErrorChar:1;         // замена ошибочных символов
      DWORD fNull:1;              // действия при приёме нулевого символа
      DWORD fRtsControl:2;        // Задает режим управления потоком для сигнала RTS
      DWORD fAbortOnError:1;      // игнорирование запись/чтение при ошибке
      DWORD fDummy2:17;           // зарезервировано
      WORD  wReserved;            // не используется, равно 0
      WORD  XonLim;               // мин. количество символов для посылки XON
      WORD  XoffLim;              // макс. кол-во символов для посылки XOFF
      BYTE  ByteSize;             // количество бит в символе
      BYTE  Parity;               // режим паритета 0-4=no,odd,even,mark,space
      BYTE  StopBits;             // длина стопового бита 0,1,2 = 1, 1.5, 2
      char  XonChar;              // символ для XON
      char  XoffChar;             // символ для XOFF
      char  ErrorChar;            // символ для замены ошибок
      char  EofChar;              // символ конца данных
      char  EvtChar;              // символ события
      WORD  wReserved1;           // резервный
  } DCB;

Для работы с DCB структурой используют API функции из библиотеки kernel32. (в файле
заголовка windows.h :

BuildCommDCB- заполняет указанную структуру DCB значениями, заданными в строке
управления устройством. Строка управления устройством использует синтаксис команды mode
MS-DOS.
SetCommState- конфигурирует коммуникационное устройство согласно данным указанным в
структуре DCB. Функция повторно инициализирует все аппаратные и управляющие настройки,
но не опорожняет очереди вывода или ввода данных.
GetCommState- читает DCB структуру.

*/
       DCB dcb;
       ::GetCommState(hCom, &dcb);      // Получение данных о текущих установках COM... и
заполнение структуры dcb

       dcb.BaudRate = CBR_2400;  // Установка новых параметров для COM.., скорость 38400
       dcb.Parity = NOPARITY;           // нет контроля четности
       dcb.ByteSize = 8;         // 8-ми битный обмен
       ::SetCommState(hCom, &dcb);      // Установка новых параметров для COM2
```

```
	::PurgeComm(hCom, PURGE_RXCLEAR); //Clears the input buffer (if the device driver
has one).
	::PurgeComm(hCom, PURGE_TXCLEAR); //Clears the output buffer (if the device driver
has one).

}

DWORD executeSyncCommand(HANDLE hCom, const char* cmd, unsigned int len, unsigned int
responseLen)
{
  printf("executing command: %.*s\n", len, cmd);

	::PurgeComm(hCom, PURGE_RXCLEAR);
	::PurgeComm(hCom, PURGE_TXCLEAR);

	::SetLastError(0); //извлекает значение кода последней ошибки вызывающего потока.
/* Код последней ошибки сохраняется при посредстве базового компонента потока. Многие
потоки не записывают поверх друг друга коды последней ошибки.*/


  DWORD toWrite = (len > 0 && cmd[len - 1] == 0) ? len - 1 : len;
  DWORD written = 0;
  ::WriteFile(hCom, (const void*)(cmd), toWrite, &written, NULL);
/*
Функция производит запись блока данных начиная с текущей позиции в файле
Параметры

$hFile              Дескриптор файла для записи

$pBuffer     Указатель на буфер, содержащий данные для записи

$iToWrite    Количество байт для записи в файл

$iWritten    Количество записанных байт

$pOverlapped [опционально]       Указатель на структуру $tagOVERLAPPED

Возвращаемое значение:

Успех: Возвращает True
Ошибка: Возвращает False
*/

  DWORD dwError = ::GetLastError();
  if (dwError)
    return dwError;

  if (!responseLen)
    return 0;

	::SetLastError(0);

	char resp[60] = { 0 };
	DWORD read = 0;
	::ReadFile(hCom, resp, responseLen, &read, NULL);
/*
Функция читает из файла блок данных, начиная с текущей позиции.
После прочтения блока, позиция переносится в конец прочитанного блока.
Параметры

$hFile              Дескриптор файла для чтения

$pBuffer     Указатель на буфер, который получает данные, считанные из файла

$iToRead     Максимальное количество байт для чтения

$iRead              Количество прочитанных байт

$pOverlapped [опционально] Указатель на структуру $tagOVERLAPPED
```

```
*/

  dwError = ::GetLastError();
  if (dwError)
    return dwError;

  printf("received response length %u: %.*s\n", responseLen, read, resp);

  return 0;
}

int main(int argc, char* argv[])
{
       if (2 != argc)
       {
              printf("please specify the COM port name\n");
              return 0;
  }

       HANDLE hCom = ::CreateFile(argv[1], GENERIC_READ | GENERIC_WRITE,
              NULL, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

/*
Параметры функции CreateFile:

lpFileName- имя COM-порта.
Может принимать значения:
"COM1","COM2","COM3","COM4","COM5","COM6","COM7","COM8","COM9",
если более одной цифры, то в формате "\\.\COM47"

dwDesiredAccess- режим доступа к файлу.
Это четырехбайтовое число, которое задает различные режимы доступа к файлу.
Нас интересует только режим чтение и запись,
этот режим задаётся числом: C0000000hex в СИ можно вместо числа записать константу с
именем "GENERIC_READ|GENERIC_WRITE".

dwShareMode- режим совместного доступа.
COM-порты ПК не поддерживают совместный доступ, только одна программа может открыть
порт.
Поэтому этот параметр должен быть равен 0("NULL")(режим запрещен).

lpSecurityAttributes- атрибуты защиты файла.
Для COM-портов не используется поэтому всегда равны 0 ("NULL").

dwCreationDistribution- управление режимом автосоздания файла.
Это четырехбайтовое число, которое для COM портов всегда должно быть 00000003hex
("OPEN_EXISTING")

dwFlagsAndAttributes- задает атрибуты создаваемого файла.
Это четырехбайтовое число, которое для COM портов всегда должно быть 0 ("NULL")

hTemplateFile- описатель файла "шаблона" по которому создавался файл.
Для COM-портов не используется поэтому всегда равен 0 ("NULL").

Пример открытия COM1 в Си:
Com_Handle = CreateFile("COM1", GENERIC_READ | GENERIC_WRITE, NULL, NULL, OPEN_EXISTING,
NULL, NULL);

*/

  if (hCom == NULL)
  {
    printf("Failed to open the com port %s\n", argv[1]);
    return 1;
  }

       printf("the COM port initialization\n");

       initializeComm(hCom);

  printf("put the device under remote control\n");
```

```
  const char cmdStatusF[] = "F\r";
  if (DWORD res = executeSyncCommand(hCom, cmdStatusF, sizeof(cmdStatusF), 7))
  {
    printf("got error %u", res);
    ::CloseHandle(hCom);
        return 1;
  }

  ::Sleep(1000);

  printf("getting status\n");
  const char cmdStatus[] = "L\r";
  if (DWORD res = executeSyncCommand(hCom, cmdStatus, sizeof(cmdStatus), 37))
  {
    printf("got error %u", res);
    ::CloseHandle(hCom);
        return 1;
  }

  printf("reset voltage to 0\n");
  const char cmdResetVoltageZero[] = "SV 00.00\r";
  if (DWORD res = executeSyncCommand(hCom, cmdResetVoltageZero,
sizeof(cmdResetVoltageZero), 0))
  {
    printf("got error %u", res);
    ::CloseHandle(hCom);
        return 1;
  }

  ::Sleep(1000);

  printf("increasing present voltage on one unit per each second\n");
  const char cmdIncreaseV[] = "SV+\r";
  for(unsigned int i = 0; i < 10; ++i)
  {
    printf("getting current status\n");
    if (DWORD res = executeSyncCommand(hCom, cmdStatus, sizeof(cmdStatus), 37))
    {
      printf("got error %u", res);
      ::CloseHandle(hCom);
          return 1;
    }

    if (DWORD res = executeSyncCommand(hCom, cmdIncreaseV, sizeof(cmdIncreaseV), 0))
    {
      printf("got error %u", res);
      ::CloseHandle(hCom);
          return 1;
    }

    ::Sleep(1000);
  }

  printf("getting resulting status\n");
  if (DWORD res = executeSyncCommand(hCom, cmdStatus, sizeof(cmdStatus), 37))
  {
    printf("got error %u", res);
    ::CloseHandle(hCom);
        return 1;
  }

  ::CloseHandle(hCom);
      return 0;
}
```