УДК 004.94
Бортник К.Я. к.т.н., Прокопюк М.І.
Луцький національний технічний університет

РОБОТА З COM-ПОРТОМ У WINDOWS ТА ANDROID

Бортник К.Я., Прокопюк М.І. Робота з com-портом у Windows та android. У даній статті висвітлено актуальне на даний час питання роботи з COM-портами у середовищах Windows та Android. Запропоновано найпоширеніші, прості та найефективніші способи для роботи з COM-портами.
Ключові слова: алгоритм, COM-порт, Android, файл.

Бортник К.Я, Прокопюк М.И. Работа с com-портом в Windows и android. В данной статье рассматриваются актуаленый в настоящее время вопрос работы с COM-портами в средах Windows и Android. Предложено распространенные, простые и эффективные способы для работы с COM-портами.
Ключевые слова: алгоритм, COM-порт, Android, файл

Bortnyk K., Prokopyuk M. Working with com-port in Windows and android In this article highlights the actual issue currently working with COM-ports in Windows and Android. A most common, simple and most effective ways to work with COM-ports.
Keywords: algorithm, COM-port, Android, file.

Розробники високорівневих мов програмування, очевидно, вважають прийом/передачу даних по протоколу rs232 через комунікаційний порт екзотичною процедурою: мовляв, пересічному користувачеві не потрібно, а непересічний - розбереться самостійно. Тому ні в turbo pascal, ні в delphi немає штатних засобів обміну даними таким способом. Проте останнім часом, особливо у зв'язку з поширенням мікропроцесорних пристроїв, таке завдання постає в аматорських програмах все частіше - в силу простоти і дешевизни реалізації послідовним портом обладнано багато наукових та інженерних приладів, різноманітні датчики і вимірники.

Простота апаратного виконання (для звичайного двостороннього зв'язку потрібно всього три дроти) асинхронного комунікаційного порта все таки веде до деякого ускладнення програмного забезпечення. Можна, звичайно, спробувати послати байт на пристрій com1 засобами dos, але успіх навряд буде досягнутий - як мінімум, треба спочатку налаштувати швидкість обміну. Тому для dos-програм це робиться засобами bios або прямим програмуванням порту "по залізу". В Windows, на щастя, є відповідні функції api.

З послідовними і паралельними портами в Windows працюють як з файлами, тому для відкриття порта використовується функція CreateFile. Її прототип виглядає ось так:

```
HANDLE CreateFile(
    LPCTSTR               lpFileName,
    DWORD                 dwDesiredAccess,
    DWORD                 dwShareMode,
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    DWORD                 dwCreationDistribution,
    DWORD                 dwFlagsAndAttributes,
    HANDLE                hTemplateFile
);
```

Щоб відкрити COM-порт ви повинні виконати цю функцію в коді своєї програми, з заданими вхідними параметрами. Результатом роботи цієї функції буде 32-бітне число handle, по якому ви зможете звертатися до створеного функцією програмному об'єкту зв'язаного з вибраним COM-портом.

Параметры функции CreateFile:

- lpFileName - ім'я COM-порта. Може приймати значення: "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9", якщо більше однієї цифри, то у форматі "\ \. \COM47".

© Бортник К.Я., Прокопюк М.І.

- dwDesiredAccess - режим доступу до файлу. Це чотирьохбайтове число, яке задає різні режими доступу до файлу. Нас цікавить тільки режим читання і запис, цей режим задається числом: C0000000hex в С можна замість числа записати константу з ім'ям "GENERIC_READ | GENERIC_WRITE".
- dwShareMode - режим спільного доступу. COM-порти ПК не підтримують спільний доступ, тільки одна програма може відкрити порт. Тому цей параметр повинен бути рівний 0 (режим заборонений).
- lpSecurityAttributes - атрибути захисту файлу. Для COM-портів не використовується тому завжди рівні 0 ("NULL").
- dwCreationDistribution - управління режимом автостворення файлу. Це чотирьохбайтове число, яке для COM портів завжди повинно бути 00000003hex ("OPEN_EXISTING").
- dwFlagsAndAttributes - задає атрибути створюваного файлу. Це чотирьохбайтове число, яке для COM портів завжди повинно бути 0 ("NULL").
- hTemplateFile - описатель файлу "шаблону" за яким створювався файл.Для COM-портів не використовується тому завжди дорівнює 0 ("NULL").

Приклад відкриття COM1:

```
Com_Handle = CreateFile("COM1", GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
```

Після відкриття COM-порта можна передавати і принімати дані через цей COM-порт.

Для передавання даних використовується API функція WriteFile з бібліотеки kernel32. Для прийому даних використовується API функція ReadFile з бібліотеки kernel32.

Приклад опису функцій WriteFile і ReadFile:

```
BOOL ReadFile(
  HANDLE hFile, // дескриптор COM порта
  LPVOID lpBuffer, // Вказівник на буфер, який приймає прочитані дані
з порта
  DWORD nNumberOfBytesToRead,// Число байтів, які читаються з порта
  LPDWORD          lpNumberOfBytesRead, // Вказівник на змінну, яка
отримує число прочитаних байтів
  LPOVERLAPPED lpOverlapped    // Вказівник на структуру OVERLAPPED.
     );
BOOL WriteFile(
  HANDLE hFile,// дескриптор COM порта
  LPCVOID       lpBuffer,// Вказівник на буфер, який містить дані, які
будуть записані у файл.
  DWORD nNumberOfBytesToWrite,// Число байтів, які будуть записані в
файл.
  LPDWORD lpNumberOfBytesWritten,//  Вказівник на змінну, яка отримує
число записаних байтів
  LPOVERLAPPED lpOverlapped // Вказівник на структуру OVERLAPPED
     );
```

Основні параметри послідовного порта описуються структурою DCB. Тимчасові параметри – структурою COMMTIMEOUTS. Існує ще декілька інформаційних і керуючих структур, але вони використовуються рідше. Налаштування порту полягає в заповненні керуючих структур і наступному виклику функцій настройки:

```
typedef struct _DCB {
      DWORD DCBlength;           // довжина структури (DCB)
      DWORD BaudRate;            // швидкість в біт/сек
      DWORD fBinary:1;           // бинарний режим
      DWORD fParity:1;           // розширення контролю пароності
      DWORD fOutxCtsFlow:1;      // слідкування за CTS
```

© Бортник К.Я., Прокопюк М.І.

```
      DWORD fOutxDsrFlow:1;       // слідкування за DSR
      DWORD fDtrControl:2;        // режим роботи сигналу DTR
      DWORD fDsrSensitivity:1;    // чутливість до DSR
      DWORD fTXContinueOnXoff:1;  // продовження передавання при XOFF
      DWORD fOutX:1;               // програмне управління потоком при
передаванні (XON/XOFF)
      DWORD fInX:1;                // програмне управління потоком при
прийомі (XON/XOFF)
      DWORD fErrorChar:1;         // заміна помилкових символів
      DWORD fNull:1;              // дії при прийомі нульового символа
      DWORD fRtsControl:2;         // Задає режим управління потоком
для сигналу RTS
      DWORD fAbortOnError:1;       // ігнорування запису/читання при
помилці
      DWORD fDummy2:17;           // зарезервовано
      WORD  wReserved;            // не використовується, рівне 0
      WORD   XonLim;                    // мін. кількість символів для
посилання XON
      WORD   XoffLim;                    // макс. кіл-ть символів для
посилання XOFF
      BYTE  ByteSize;             // кількість біт в символе
      BYTE    Parity;                        // режим паритету 0-
4=no,odd,even,mark,space
      BYTE  StopBits;             // довжина стопового біта 0,1,2 = 1,
1.5, 2
      char  XonChar;              // символ для XON
      char  XoffChar;             // символ для XOFF
      char  ErrorChar;            // символ для заміни помилок
      char  EofChar;              // символ кінця даних
      char  EvtChar;              // символ події
      WORD  wReserved1;           // резервний
  } DCB;
```

Для роботи з DCB структурою використовують API функції з бібліотеки kernel32:

BuildCommDCB – заповнює зазначену структуру DCB значеннями, заданими в рядку управління пристроєм. Рядок управління пристроєм використовує синтаксис команди mode MS-DOS.

SetCommState – конфігурує комунікаційний пристрій згідно з даними вказаними в структурі DCB. Функція повторно ініціалізує всі апаратні і керуючі настройки, але не спорожняєте черги виводу або введення даних.

GetCommState – читає DCB структуру.

Після закінчення роботи з портом, його потрібно закрити. Закриття порта здійснюється функцією CloseHandle з бібліотеки kernel32.

Приклад опису функції CloseHandle:

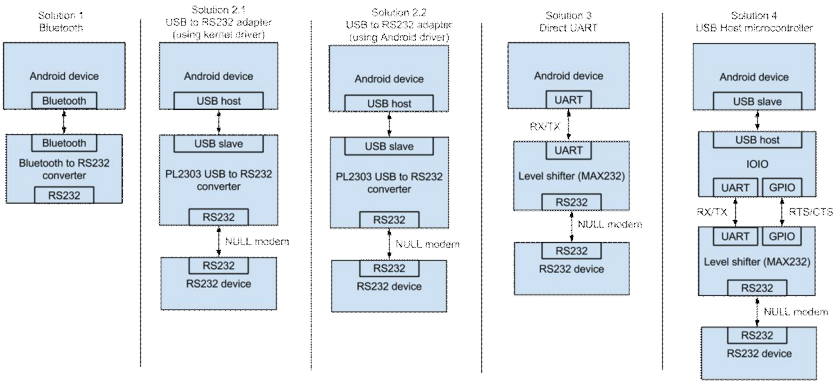
```
BOOL CloseHandle(HANDLE hObject);
```

Для роботи з COM-портом на Android, на даний час існують різноманітні способи. Одним із способів є використання Virtual Serial Port разом з TCP_SerPort, яка створює віртуальний COM порт в системі і перенаправляє дані по мережі на Android. Також можна працювати через емулятор терміналу такий як Irssi ConnectBot з підтримкою USB Host. І варіант на якому я зупинився, це використання бібліотеки android-serialport-api.

Дане рішення представляє собою Java обгортку в якій через JNI здійснюються виклики до USB пристрою.

Розробники бібліотеки пропонують 4 варіанти підключення Android пристрою до COM-порта через USB.

© Бортник К.Я., Прокопюк М.І.



Перевагами такого рішення є:

- USB-RS232 перетворювач можна легко знайти на будь-якому радіо ринку;
- Ніяких "збочень" при роботі з Android пристроями;
- Можлива висока швидкість передавання інформації.

Недоліками є:

- Знадобиться USB хост коннектор;
- Більш за все, пристрій прийдеться отримати рут-права.

Для початку нам знадобиться завантажити Android NDK – середовище для роботи з нативним кодом в Java. Також необхідно завантажити вихідний код проекту з snv для робот из Serial port RS232.

Далі в Android проекті нам необхідно створити папку /jni і скопіювати в неї вміст папки /jni проекту, викачаного з svn (або просто скопіювати всю папку /jni) в проект Android. Потім необхідно додати наступні файли з викачаного проекту:

- SerialPort.java;
- Application.java;
- SerialPortActivity.java;
- SerialPortFinder.java.

Дані файли дозволяють працювати з COM портом, але Вам потрібно буде їх відредагувати під потреби свого проекту. Як відомо, всі пристрої в *nix подібних системах знаходяться по шляху /dev. Для того щоб встановити шлях до пристрою і baud rate, необхідно у файлі Application.java встановити відповідні значення полів.

```
String path = -path to device-;
int baudrate = -baud rate-;
```

Клас SerialPortActivity.java є розширенням класу Activity і містить абстрактний метод protected abstract void onDataReceived (final byte [] buffer, final int size). Ви можете успадковувати від цього класу свої Activity, в яких буде робота з Serial Port, і, перевизначивши метод onDataReceived, обробити отримання даних з порту.

```
@Override
protected void onDataReceived(final byte[] buffer, final int size) {
runOnUiThread(new Runnable() {
public void run() {
//TO DO your logic
}
});
}
```

Можна винести логіку роботи з портом в свої власні класи або класи і не використовувати успадкування від SerialPortActivity.java.

За допомогою класу SerialPortFinder.java з його методами getAllDevices() і getAllDevicesPath() ви можете отримати списки всіх пристроїв і їх шляхів відповідно.

Запис у порт здійснюється за допомогою простого запису в OutputStream, створеного за допомогою класу SerialPort.java, COM порту.

© Бортник К.Я., Прокопюк М.І.

```
mOutputStream.write(new String(“text”).getBytes());
mOutputStream.write('\n');
```

Завантаження нативного коду в Android додаток міститься в класі SerialPort.java за допомогою виклику System.loadLibrary ("serial_port"):

```
private native static FileDescriptor open(String path, int baudrate, int flags);
public native void close();
static {
System.loadLibrary("serial_port");
}
```

Параметр serial_port - це модуль, отриманий внаслідок компіляції коду C за допомогою NDK. Він зазначений в make файлі /jni/Android.mk. Файл SerialPort.c в папці / jni містить виклик нативних функцій системи для роботи з Serial port. Такі параметри COM порту як Data bits, Parity, Stop bits та інші можна змінити в цьому файлі допомогою структури termios, наприклад, так:

```
cfg.c_cflag |= ~PARENB;
cfg.c_cflag &= ~CSTOPB;
cfg.c_cflag &= ~CSIZE;
cfg.c_cflag |= CS8;
(Data bits=8, Parity=none, Stop bits=1)
```

Після зміни файлу SerialPort.c необхідно скомпілювати необхідні бібліотеки наступним чином:

1. Відкрити командний рядок
2. Зайти в папку з NDK.
3. Встановити шлях до проекту - set NDK_PROJECT_PATH = <path to your android project>
4. Набрати - ndk-build.

Бібліотеки будуть скомпільовані і додані в папку /libs вашого проекту. Після цього для установки Android програми на пристрій можна виконати наступну команду:

```
adb install <path to you .apk file>
```

Для коректної роботи serialPortFinder необхідно, щоб перехідник USB-RS232 був підключений, визначений в /dev.

Таким чином, у нас є всі інструменти для здійснення спілкування з великою кількістю пристроїв через через COM-порт у Windows та Android. Використання цих інструментів для відправки і отримання інформації допомагає управляти будь якими пристроями і реалізовувати най неочікуваніші бізнес ідеї.

1. А. А. Мячев, В. Н. Степанов, В. К. Щербо; Интерфейсы систем обработки данных. - М.: Радио и связь, 1989. - 416 с.
2. https://ru.wikibooks.org/w/index.php?title=COM-порт_в_Windows_(программирование)
3. http://www.lammertbies.nl/comm/info/RS-232_specs.html

© Бортник К.Я., Прокопюк М.І.