# Creating a Serial communication on Win32

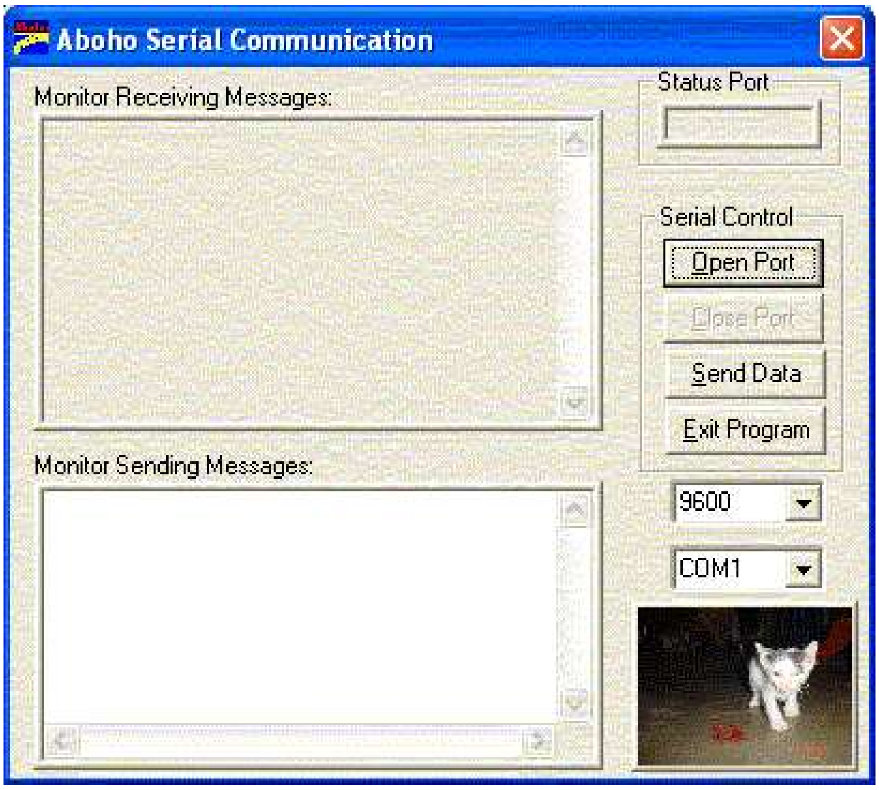
**konchat**, 21 Oct 2002

4.55 (64 votes) Rate: Vote!

The purpose of this article is to describe how to interface to serial port on Win32.

**Download demo project - 12.4 KB**

**Download source - 33.4 KB**


Aboho Serial Communication
Monitor Receiving Messages:
Status Port
Serial Control
Open Port
Close Port
Send Data
Exit Program
Monitor Sending Messages:
9600
COM1

# Introduction

The purpose of this article is to describe how to interface to serial port on Win32. The serial port can be implemented by several techniques such as ActiveX, access I/O and file operation. This article explains the use of serial port on Win32 platform by file operation technique. The programmer can use *kernel32.lib* library that is provided with the Microsoft Visual C++ Version 6.0. In Microsoft Windows (2000, Me, XP and 95/98), serial port can be treated as a file. Therefore it's possible to open a serial port by using Windows file-creating function.

This article explains not only about serial port communication but also how to implement multi-tasking that can apply with our project "serial port" application. The reason why the software (serial communication) will be implemented with multi-tasking method is that the serial communication application has to handle work with more than one task at the same time. For example data-reading task, data-sending task, GUI task etc.

These topics describe the basic operation of interfacing a serial port on Win32:

# Initial/Open serial port communication.

- Creating a port handle
- Restoring a configuration (DCB)
- Modifying a configuration
- Storing a configuration
- Setting a Time-Out communication

# Receive/Send data

- Sending data
- Receiving data
- Closing a serial port

# Design approach

## Initial/Open serial port

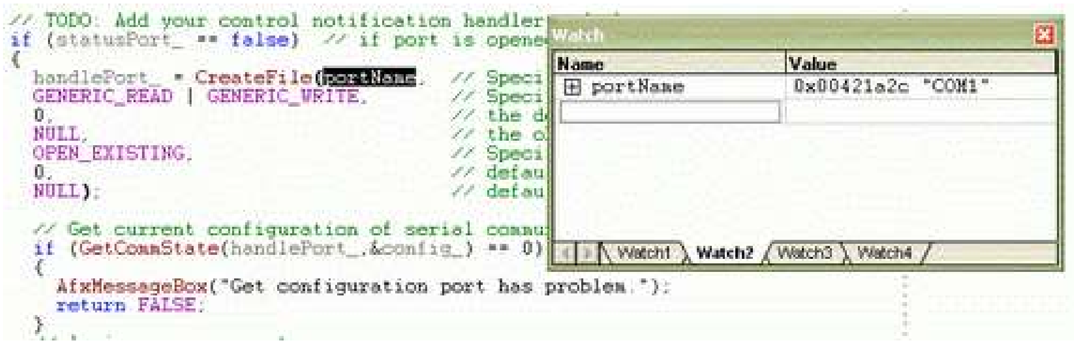
The first step in opening a serial port is initiation or setting a serial port's configuration. The purpose of this is to create the serial port agent. All throughout the article we are going to use a file handle as serial port agent.

### **Creating a port handle**

The serial port's handle is a handle that can be used to access the object of serial port. The function that is used to create the serial port handle is the `CreateFile` function. The following code shows the function that is used to create a handle:

```
handlePort_ = CreateFile(portName,  // Specify port device: default "COM1"
GENERIC_READ | GENERIC_WRITE,       // Specify mode that open device.
0,                                  // the devide isn't shared.
NULL,                               // the object gets a default security.
OPEN_EXISTING,                      // Specify which action to take on file.
0,                                  // default.
NULL);                              // default.
```

As figure 2 shows, `portName` = "COM1": the `portName` is a variable that is declared by `const char*`. It is used to specify port name that wants to create a serial port handle.



```
// TODO: Add your control notification handler
if (statusPort_ == false)  // if port is opene
{
  handlePort_ = CreateFile(portName,          // Speci
  GENERIC_READ | GENERIC_WRITE,               // Speci
  0,                                          // the d
  NULL,                                       // the c
  OPEN_EXISTING,                              // Speci
  0,                                          // defau
  NULL);                                      // defau

  // Get current configuration of serial commu
  if (GetCommState(handlePort_,&config_) == 0)
  {
    AfxMessageBox("Get configuration port has problem.");
    return FALSE;
  }
```

| Name | Value |
|---|---|
| ⊞ portName | 0x00421a2c "COM1" |

***Figure 2:* `CreateFile` *function***

## Restoring a configuration

The restoration of serial port configuration is getting current configuration at control device. The configuration of serial port includes parameters that are used for setting a serial communications device.

The `GetCommState` function is used to get the current device-control and then fills to a device-control block (a DBC structure) with the current control settings for a specified communications device. The following code shows the function that is used to get the current control device:

```
// Get current configuration of serial communication port.
if (GetCommState(handlePort_,&config_) == 0)
{
   AfxMessageBox("Get configuration port has problem.");
   return FALSE;
}
```

## Modifying a configuration

When you already have serial port configuration in the DBC format, you have to modify parameters a bit. Following code shows the parameters modified:

```
// Assign user parameter.
config_.BaudRate = dcb.BaudRate;  // Specify buad rate of communicaiton.
config_.StopBits = dcb.StopBits;  // Specify stopbit of communication.
config_.Parity = dcb.Parity;      // Specify parity of communication.
config_.ByteSize = dcb.ByteSize;  // Specify  byte of size of communication.
```

- `DWORD BaudRate`:

  Current baud rate (default = 9600)

- `BYTE StopBits`:

  0,1,2 = 1, 1.5, 2 (default = 0)

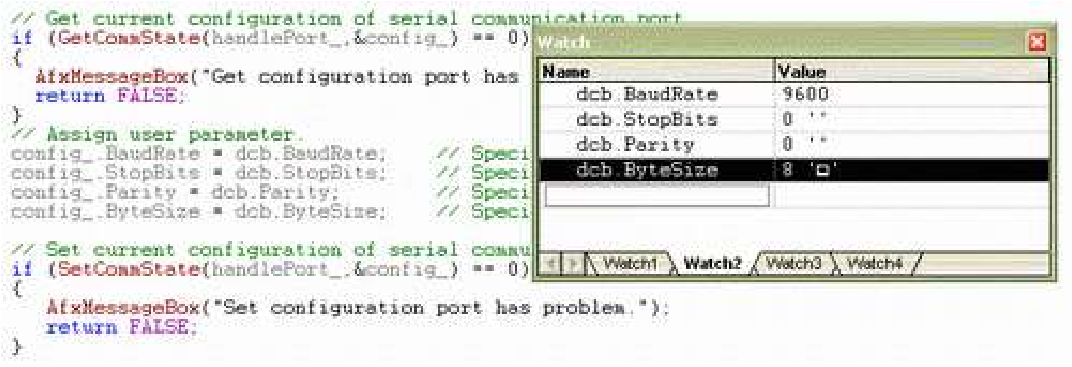
- `BYTE Parity`:

  0-4= no, odd, even, mark, space (default = 0)

- `BYTE ByteSize`:

  Number of bits/byte, 4-8 (default = 8)

Note: Recommend that programmers use default value for typical communication. As shown in figure 3, Watch Dialog Box shows the default values that are used for typical communication.



```
// Get current configuration of serial communication port
if (GetCommState(handlePort_,&config_) == 0)
{
  AfxMessageBox("Get configuration port has
  return FALSE;
}
// Assign user parameter.
config_.BaudRate = dcb.BaudRate;    // Speci
config_.StopBits = dcb.StopBits;    // Speci
config_.Parity = dcb.Parity;        // Speci
config_.ByteSize = dcb.ByteSize;    // Speci

// Set current configuration of serial commu
if (SetCommState(handlePort_,&config_) == 0)
{
   AfxMessageBox("Set configuration port has problem.");
   return FALSE;
}
```

```
Watch
Name            Value
  dcb.BaudRate  9600
  dcb.StopBits  0 ''
  dcb.Parity    0 ''
  dcb.ByteSize  8 '□'
```

***Figure 3: Serial port configuration***

## Storing a configuration

The next step is the storage of new configuration that is modified already into device control. Call `SetCommState` API function to store the configuration. The `SetCommState` function configures a communications device according to the specifications in a device-control block (a DBC structure). The function reinitializes all hardware and control settings, but it does not empty output or input queues. Following code shows storage of a new configuration:

```
if (SetCommState(handlePort_,&config_) == 0)
{
  AfxMessageBox("Set configuration port has problem.");
  return FALSE;
}
```

## Setting a Time-Out communication

The final step in serial port opening is setting communication Time-out by using the `COMMTIMEOUTS` data-structure and calling `SetCommTimeouts` function. The code below shows setting time-out of communication:

```
// instance an object of COMMTIMEOUTS.
COMMTIMEOUTS comTimeOut;
// Specify time-out between charactor for receiving.
comTimeOut.ReadIntervalTimeout = 3;
// Specify value that is multiplied
// by the requested number of bytes to be read.
comTimeOut.ReadTotalTimeoutMultiplier = 3;
// Specify value is added to the product of the
// ReadTotalTimeoutMultiplier member
comTimeOut.ReadTotalTimeoutConstant = 2;
// Specify value that is multiplied
// by the requested number of bytes to be sent.
comTimeOut.WriteTotalTimeoutMultiplier = 3;
// Specify value is added to the product of the
// WriteTotalTimeoutMultiplier member
comTimeOut.WriteTotalTimeoutConstant = 2;
// set the time-out parameter into device control.
SetCommTimeouts(handlePort_,&comTimeOut);
```

### ReadIntervalTimeout

Specifies the maximum time, in milliseconds, allowed to elapse between the arrival of two characters on the communications line. During a `ReadFile` operation, the time period begins when the first character is received. If the interval between the arrival of any two characters exceeds this amount, the `ReadFile` operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used.

A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the characters that have already been received, even if no characters have been received.

### ReadTotalTimeoutMultiplier

Specifies the multiplier, in milliseconds, used to calculate the total time-out period for read operations. For each read operation, this value is multiplied by the requested number of bytes to be read.

### ReadTotalTimeoutConstant

Specifies the constant, in milliseconds, used to calculate the total time-out period for read operations. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes.

A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.

### WriteTotalTimeoutMultiplier

Specifies the multiplier, in milliseconds, used to calculate the total time-out period for write operations. For each write operation, this value is multiplied by the number of bytes to be written.

### WriteTotalTimeoutConstant

Specifies the constant, in milliseconds, used to calculate the total time-out period for write operations. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written.

A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.

Note: After the user has set the time-out of communication without any error, the serial port has opened already.

## Sending data

Most of data transmission of serial port is done as writing a file. Programmer can apply file operation functions for sending data to serial port. The WriteFile function is a function used to send data in serial port communication.

```
if (WriteFile(handlePort_,   // handle to file to write to
    outputData,              // pointer to data to write to file
    sizeBuffer,              // number of bytes to write
    &length,NULL) == 0)      // pointer to number of bytes written
{
    AfxMessageBox("Reading of serial communication has problem.");
    return FALSE;
}
```

Note: If the function succeeds, the return value is nonzero.

## Receiving data

Most of data reception of serial communication is done as reading a file. Programmer can apply file operation functions for receiving data from serial port. The ReadFile function is the function that handles reading data in serial port communication.

```
if (ReadFile(handlePort_,  // handle of file to read
    inputData,                  // handle of file to read
    sizeBuffer,                 // number of bytes to read
    &length,                    // pointer to number of bytes read
    NULL) == 0)                 // pointer to structure for data
{
    AfxMessageBox("Reading of serial communication has problem.");
    return FALSE;
}
```

Note: If the function succeeds, the return value is nonzero.

## Closing a serial port

The serial port closing calls the `CloseHandle` API function to close handle of device control.

```
if(CloseHandle(handlePort_) == 0)    // Call this function to close port.
{
    AfxMessageBox("Port Closeing isn't successed.");
    return FALSE;
}
```

Note: If the function succeeds, the return value is nonzero.

## License

## Share

TWITTER FACEBOOK LINKEDIN REDDIT GOOGLE+

## About the Author

**konchat**
Web Developer
United States

I am living in Thailand, I am working as a Software Engineer at a transportation Compay, In my spare time I like to play ping pong.

My programming experience includes C/C++, Java, C#, Assembler(Intel/Motolora), MFC, ATL, OpenGL, a bit for HTML.

## You may also be interested in...

A Solution Blueprint for DevOps

SAPrefs - Netscape-like Preferences Dialog

Serial communication for Win32 with modem support

Window Tabs (WndTabs) Add-In for DevStudio

Win32 SDK Serial Comm Made Easy

To Heap or not to Heap; That's the Large Object Question?

## Comments and Discussions

Add a Comment or Question

First Prev Next

**Not working**
**Member 10994587 28-Dec-17 5:35**

**Thank You A Lot!**
**Carlos1907 26-Jun-14 2:44**

**INVALID_HANDLE_VALUE for COM10 or higher fix**
**NZSmartie 23-Mar-14 4:48**

**How to get portnames**
**g8anush 2-Feb-14 23:27**

**Serial port reading is hanging**
**Member 10434756 29-Nov-13 13:20**

**VC++ 2010 Compile issue**
**MGreatwolf 13-Jan-13 20:01**

Re: VC++ 2010 Compile issue
**john_1726** 27-Apr-15 13:09

**My vote of 1**
**samdcvn 20-Sep-12 20:57**

Re: My vote of 1

**Carlos1907** 26-Jun-14 1:39

**My vote of 4**
**w-peuker** **26-Jul-12 6:22**

Re: My vote of 4
**w-peuker** 26-Jul-12 6:26

Re: My vote of 4
**Vozzie2** 2-Oct-12 8:23

**Why does it not send my data to the port before I close the application**
**dgipling** **5-Jul-12 10:42**

Re: Why does it not send my data to the port before I close the application
**Agnius Vasiliauskas** 8-Jan-13 7:27

**Want to display entire packet**
**Member 8262724** **23-Sep-11 15:44**

**My vote of 5**
**Norm Heyder** **25-Feb-11 17:32**

**serial communication**
**arvinder456** **18-Nov-08 16:11**

**Detected memory leaks!**
**Bincker** **16-Sep-08 4:43**

Re: Detected memory leaks!
**wct** 2-Jan-09 9:51

Re: Detected memory leaks!
**Bincker** 9-Jan-09 2:50

Re: Detected memory leaks!
**wct** 10-Jan-09 10:36

Re: Detected memory leaks!
**Bincker** 19-Jan-09 0:47

**Is there a way to know the buffer size before reading it ?**
**joelparker** **19-Aug-08 13:17**

**Way to close a comm port on which the handle has been lost?**
**Member 1929642** **7-Aug-08 12:12**

Re: Way to close a comm port on which the handle has been lost?
**Raven1979933** 5-Oct-08 2:59

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.