Using the Serial Ports in Visual C++

The serial communications ports are COM1–COM8, but most computers only have COM1 and COM2 installed. Some have a single communication port (COM1). Here, the Windows API is discussed and used to operate the COM ports.

The serial ports are accessed through any version of Windows and Visual C++ by using a few system application interface (API) functions. An example of a short C++ function that accesses the serial ports is listed in Example 1. The function is called WriteComPort and it contains two parameters. The first parameter is the port as in COM1, COM2, etc and the second parameter is the character to be sent through the port. A return true indicates that the character was sent and a return false indicates that a problem exists. To use the function to send the letter A through the COM1 port call it with a WriteComPort("COM1", "A"). This function is written to send only a single byte through the serial COM port, but could be modified to send strings. To send a 00H (no other number can be sent this way) through COM2 use WriteComPort("COM2", 0x00). Notice that the COM port is set to 9600 Baud, but this is easily changed by changing the CBR_9600 to another acceptable value. See Table 1 for the allowed Baud rates.

EXAMPLE 1

```
bool WriteComPort(CString PortSpecifier, CString data)
{
	DCB dcb;
	DWORD byteswritten;

	HANDLE hPort = CreateFile(

		PortSpecifier,
		GENERIC_WRITE,
		0,
		NULL,
		OPEN_EXISTING,
		0,
		NULL
	);

	if (!GetCommState(hPort,&dcb))
		return false;

	dcb.BaudRate = CBR_9600;		//9600 Baud
	dcb.ByteSize = 8;			//8 data bits
	dcb.Parity = NOPARITY;			//no parity
	dcb.StopBits = ONESTOPBIT;		//1 stop

	if (!SetCommState(hPort,&dcb))
		return false;

	bool retVal = WriteFile(hPort,data,1,&byteswritten,NULL);
	CloseHandle(hPort);			//close the handle
	return retVal;
}
```

The CreateFile structure creates a handle to the COM ports that can be used to write data to the port. After getting and changing the state of the port to meet the Baud rate requirements, the WriteFile function sends data to the port. The parameters used with the WriteFile function are the file handle (hPort) the data to be written as a string, the number of bytes to write(1 in this example), and a place to store the number of bytes actually written to the port.

Table 1 Allowable Baud rates for the COM ports.

Keyword	Speed in bits per second
CBR_110	110
CBR_300	300
CBR_600	600
CBR_1200	1200
CBR_2400	2400
CBR_4800	4800
CBR_9600	9600
CBR_14400	14400
CBR_19200	19200
CBR_38400	38400
CBR_56000	56000
CBR_57600	57600
CBR_115200	115200
CBR_128000	128000
CBR_256000	256000

Receiving data through the COM port is a little more challenging because errors occur more frequently than with transmission. There are also many types of errors that can be detected that often should be reported to the user. Example 2 illustrates a C++ function that is used to read a character from the serial port called ReadByte. The ReadByte function either returns the character read from the port or an error code of 0x100 is the port could not be opened or 0x101 is the receiver detected an error. If data are not received, this function will hang because no timeouts were set.

EXAMPLE 2

```
int ReadByte(CString PortSpecifier)
{
	DCB dcb;
	int retVal;
	BYTE Byte;
	DWORD dwBytesTransferred;
	DWORD dwCommModemStatus;

	HANDLE hPort = CreateFile(

		PortSpecifier,
		GENERIC_READ,
		0,
		NULL,
		OPEN_EXISTING,
		0,
		NULL
	);

	if (!GetCommState(hPort,&dcb))
		return 0x100;

	dcb.BaudRate = CBR_9600;		//9600 Baud
	dcb.ByteSize = 8;			//8 data bits
	dcb.Parity = NOPARITY;			//no parity
	dcb.StopBits = ONESTOPBIT;		//1 stop

	if (!SetCommState(hPort,&dcb))
		return 0x100;

	SetCommMask (hPort, EV_RXCHAR | EV_ERR);	//receive character event
	WaitCommEvent (hPort, &dwCommModemStatus, 0);	//wait for character

	if (dwCommModemStatus & EV_RXCHAR)
		ReadFile (hPort, &Byte, 1, &dwBytesTransferred, 0);	//read 1
	else if (dwCommModemStatus & EV_ERR)
		retVal = 0x101;
	retVal = Byte;
	CloseHandle(hPort);
	return retVal;
}
```