Windows Standard

Serial Communications

Users Manual

(WSC_USR)

Version 5.4

August 11, 2015

This software is provided as-is.
There are no warranties, expressed or implied.

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815 USA

Email: info@marshallsoft.com
Web: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **Windows Standard Serial Communications Library** (WSC) is a serial communication component DLL library that provides full control over a serial port. WSC uses the standard Windows API (Application Programmer's Interface) to communicate with any device connected to a serial port.

A simple interface allows accessing data from a serial port using RS232 or multi-drop RS422 / RS485 serial ports. WSC also supports virtual ports such as those created by Bluetooth and USB/serial converters. WSC provides the capability to quickly write applications to control serial devices such as barcode scanners, modems, lab instruments, medical devices, USB serial devices, scales, GPS navigation, etc.

The User's Manual applies to the **Windows Standard Serial Communications Library** (WSC) for all supported languages. It discusses serial port processing, language independent programming issues, as well as purchasing and licensing information.

There are separate versions of the **Windows Standard Serial Communications Library** for C/C++ and .NET (WSC4C), Delphi (WSC4D), Visual Basic and VB.NET (WSC4VB), PowerBASIC (WSC4PB), Visual FoxPro (WSC4FP), dBASE (WSC4DB) and Alaska Xbase++ (WSC4XB). WSC also works with Microsoft Visual Studio .NET (C++ .NET, C#, VB.NET, Delphi .NET). Purchase a developer license for one software development language and use it with all others. All versions of WSC use the same DLLs (WSC32.DLL or WSC64.DLL). However, the examples provided for each version are written and tested for the specified computer programming language. Development time is shortened because programmers need only learn one interface.

The **Windows Standard Serial Communications Library (WSC)** will run under Windows XP through Windows 10. WSC supports both 32-bit and 64-bit applications.. Fully functional 30-day evaluation versions for all of our serial comm products can be downloaded from our web site at

http://www.marshallsoft.com/serial-communication-library.htm

We also have declaration files and example programs for a few other languages (such as MATLAB). We also offer a GPS specific library. See

http://www.marshallsoft.com/gps-communication-library.htm

1.1 Technical Support

We want you to be successful in developing your applications using the **Windows Standard Serial Communications Library (WSC)**! We are committed to providing the best library that we can. If you have any suggestions or comments, please let us know.

If you are having a problem using WSC, refer to Section 7.0 "Resolving Problems". Technical support is provided to customers with a current WSC license. License updates are available for a nominal fee. If you cannot resolve your problem, email us at:

info@marshallsoft.com

To avoid having your email deleted by our Spam scanners, begin the subject with "WSC" or "MSC HELP", and zip up any attachments. Provide your Customer ID to expedite our response.

The latest versions of our products are available on our web site at
http://www.marshallsoft.com/products.htm

Registered users can download an update (for a period of one year) to the latest DLL's at
http://www.marshallsoft.com/update.htm

1.2 Documentation Set

The complete set of documentation consists of four manuals. This is the second manual (WSC_USR) in the set.

- WSC_4x Programmer's Manual (WSC_4x.PDF)
- WSC User's Manual (WSC_USR.PDF)
- WSC Reference Manual (WSC_REF.PDF)
- Serial User's Manual (SERIAL.PDF)

The WSC_4x Programmer's Manual is the computer programming language specific manual and provides information needed to compile your programs as well as descriptions of each example program included. The "x" in WSC_4x Programmer's Manual specifies the host language such as C for C/C++, VB for Visual Basic, etc. The language specific manuals are as follows:

```
[NAME]        [DESCRIPTION]
WSC_4C    :   WSC Programmer's Manual for C/C++ and .NET
WSC_4D    :   WSC Programmer's Manual for Delphi
WSC_4VB   :   WSC Programmer's Manual for Visual Basic and VB.NET
WSC_4PB   :   WSC Programmer's Manual for PowerBASIC
WSC_4FP   :   WSC Programmer's Manual for Visual FoxPro
WSC_4DB   :   WSC Programmer's Manual for dBASE
WSC_4XB   :   WSC Programmer's Manual for Xbase++
```

The WSC User's Manual (WSC_USR) discusses language independent serial communications programming issues including modem control. Purchasing and licensing information is also provided in the manual. Read this manual after reading the WSC_4* Programmer's Manual.

The WSC Reference Manual (WSC_REF) contains details on each individual WSC function.

The Serial Communications Manual (SERIAL) contains background information on serial port hardware.

Documentation can be accessed online at http://www.marshallsoft.com/support.htm

1.3 How to Purchase

A developer license for the professional version of the **Windows Serial Communications Library (WSC)** may be purchased for $115 (USD) for electronic (email) delivery or for $195 (USD) with source code (ANSI C) for the DLLs. Licenses are sold on a per developer basis.

The fastest and easiest way to order is on our web site at

http://www.marshallsoft.com/order.htm

You can also order by completing INVOICE.TXT (pro forma invoice contained within the WSC zip file) and emailing (info [at] marshallsoft.com).

Multiple copy discounts (3 or more) and site licenses are available. Please call for details.

We accept American Express, VISA, MasterCard, Discover, checks in US dollars drawn on a US bank, and International Postal Money Orders.

The registered package includes:

- WSC32 and WSC64 Library without the evaluation version popup window.
- Free downloadable updates to the registered DLLs for one year.
- Technical support (telephone and email) for one year.

Note that the registered WSC DLLs never expire.

1.3.1 Source Code

Source code is available for the purpose of re-compiling WSC32.DLL and WSC64.DLL. Source code for the DLL library is standard ANSI C. The source code is copyrighted by MarshallSoft Computing and may not be released in whole or in part.

There are two ways to order Source Code for the **Windows Serial Communication Library SDK.**

(1) Source Code can be ordered at the same time as the Developer's License for $195 (for both).

(2) Source Code can be ordered within one year of purchasing a Developer's License for $100. After one year, a Developer's License update ($30, $55 or $75) must be purchased prior to purchasing the source code.

1.3.2 Academic Discount

We offer an "academic price" of 40% off the normal price for prepaid email orders to faculty and full-time students currently enrolled in any accredited high school, college, or university.

To qualify for the discount, your school must have a web site and you must have an email address at your school. When placing an order online, ask for the "academic discount" or enter "student at" (or "faculty at") along with your schools web site address (URL) in the comments field of the order. Your order will be sent to your email address at your school.

This offer is not retroactive and cannot be used with any other discount. Products bought with academic pricing cannot be used for any commercial purpose nor can the WSC DLLs be distributed. The academic discount does not apply to source code.

1.4 Updates

When a developer license is purchased for the **Windows Standard Serial Communications SDK,** the developer will receive a set of registered DLLs plus a license file (WSCxxxx.LIC) that can be used to update the registered DLL's (does not include source code) for a period of one year from purchase. Updates can be downloaded from

http://www.marshallsoft.com/update.htm

After one year, the developer license must be updated to be able to download updates and receive technical support. The license can be updated for:

- $30 if the update is ordered within one year of the original purchase (or previous update).
- $55 if the update is ordered between one and three years of the original purchase (or previous update).
- $75 if the update is ordered after three years of the original purchase (or previous update). .

A license update includes an additional year of technical support and downloadable updates.

Source code previously purchased may be updated for $40 in addition to the cost of the update ($30, $55 or $75).

Note that the registered WSC DLL **does not** expire.

Also see the file UPDATES.TXT.

1.5 Customer ID

The Customer ID is 5 digits following the product name (WSC) in your license file. For example, customer 12345 would receive license file **WSC12345.LIC**. Provide the Customer ID when contacting us for technical support (WSC4C 12345).

1.6 License File

A license file named WSCxxxxx.LIC, where "xxxxx" is the 5 digit Customer ID, is provided with each developer license. The license file is an encrypted binary file used for downloading WSC updates as explained in Section 1.4 "Updates". The license file is required in order to update your registered DLL's.

1.7 Distribution

In order to run your application (that calls WSC functions) on another computer, the file WSC64.DLL (for 64-bit applications) or WSC32.DLL (for 32-bit applications) must be copied to the Windows directory of the other computer. The Windows directory is normally \WINDOWS. Do not attempt to "register" the DLLs.

2 WSC Application Notes

2.1 Keycode

When developer license for WSC is purchased, a new Keycode and a new set of the WSC DLLs will be provided. Pass the new keycode as the argument to **SioKeyCode**. The keycode will be found in the file named "KEYCODE". The keycode for the evaluation (shareware) version is 0. The keycode for the registered version will be a unique 9 or 10 digit number. Note: Your keycode is NOT your Customer ID/Registration number.

2.2 Dynamic Link Libraries

WSC includes Win64 [WSC64.DLL] and Win32 [WSC32.DLL] dynamic link libraries (DLLs). A DLL is characterized by the fact that it need not be loaded until required by an application program and that only one copy of the DLL is necessary regardless of the number of application programs that use it. Contrast this to a static library that is bound at link time to each and every application that uses it. Note that registered users

The WSC DLL's make calls **only** to the core Windows DLL's and do **not** depend on support DLL's. The DLLs can be called from **any** language capable of making calls to the Windows API, and will work with all versions of your compiler.

2.3 GUI and Console Mode

WSC functions can be called from WIN32/Win64 console mode programs as well as GUI programs. A "console mode" program is a WIN32/Win64 command line program running in a command window. Although console mode programs look like DOS programs, they are WIN32/WIN64 programs which have access to the entire Windows address space.

2.4 Using the WSC Library

The first **Windows Serial Comm Library (WSC)** function that should be called is **SioKeyCode**, which initializes the WSC library and allocates necessary resources. **SioKeyCode** is typically called in the initialization section of your application.

After **SioKeyCode** is called, you are ready to call the other WSC functions.

Before exiting your application, **SioDone** should be called to terminate further serial processing allowing other applications to use the port.

The best way to get familiar with WSC is to try out one of the example programs. The example programs are described in the WSC Programmer's Manual for each development environment.

```
[NAME]        [DESCRIPTION]
WSC_4C     :  WSC Programmer's Manual for C/C++
WSC_4D     :  WSC Programmer's Manual for Delphi
WSC_4VB    :  WSC Programmer's Manual for Visual Basic
WSC_4PB    :  WSC Programmer's Manual for PowerBASIC
WSC_4CB    :  WSC Programmer's Manual for COBOL
WSC_4FP    :  WSC Programmer's Manual for Visual FoxPro
WSC_4DB    :  WSC Programmer's Manual for Visual dBase
WSC_4XB    :  WSC Programmer's Manual for Xbase++
```

2.5 Development Languages Supported

We have versions of the **Windows Serial Comm Library (WSC)** component library for C/C++ (WSC4C), Borland Delphi (WSC4D), Visual Basic and VB.NET (WSC4VB), PowerBASIC (WSC4PB), Visual FoxPro (WSC4FP), dBASE (WSC4DB), and Alaska Xbase++ (WSC4XB). All versions of WSC use the same DLLs (WSC64.DLL and WSC32.DLL). Evaluation versions for these may be downloaded from our website at

http://www.marshallsoft.com/serial-communication-library.htm

The **Windows Serial Comm Library (WSC)** DLL's can also be used with any application written in any language capable of calling the Windows (95/98/Me, NT/2000/2003/2012/XP/Vista/Win7/Win8/Win10) API. Win32 applications use WSC32.DLL while Win64 (x64) applications use WSC64.DLL.

2.5.1 Using WSC with Supported Languages.

Once a developer license is purchased for one programming language version of the **Windows Serial Comm Library SDK (WSC),** the same developer can use it with all other supported programming languages. Supported languages are C/C++ (.NET), Visual Basic (VB.NET), PowerBASIC, Delphi, Visual FoxPro, Visual dBase, and Xbase++.

For example, assume that the developer has previously downloaded and installed the registered version of WSC4C and now ants to also call WSC functions from Visual Basic.

1. Make a backup copy of the WSC DLLs (WSC64.DLL and WSC32.DLL) found in the Windows directory \WINDOWS.
2. Download and install the evaluation version of WSC4VB (http://www.marshallsoft.com/wsc4vb.htm) into a separate directory from WSC4C.
3. Compile and run the Visual Basic WSCVER example program found in the APPS directory created in step 2 above. It should display the pop-up evaluation screen.
4. Restore WSC64.DLL and WSC32.DLL to your Windows directory as saved in step 1 above.
5. Paste the key code value found in (the registered version of WSC4C) KEYCODE.H into KEYCODE.BAS.
6. Run the Visual Basic WSCVER example program again. It should no longer display the pop-up screen.

A quicker and easier way would be to request multiple programming versions of WSC when a developer license is purchased. There is no additional charge.

2.5.2 Using WSC with Unsupported Languages

The **Windows Serial Comm Library DLLs** can be called from any program capable of using the Windows API. In addition to declaration files for the development environments listed in Section 2.5.1, we also have declaration files for the following languages:

[LANGUAGE]	[FILE]
Fujitsu COBOL	WSC32.CBI
ABSOFT FORTRAN	WSC32.INC
LabView	WSC32.TXT

Email us a call if you need a declaration not listed above.

If you have interfaced WSC to an unusual programming language, email us the declaration file!

2.6 UARTs and Serial Ports

Bytes are moved from the UARTs FIFO buffer to the receive memory buffer by the Windows Interrupt Service Routine (ISR), asynchronous to your code. Similarly, the Windows ISR moves bytes from your transmit queue to the UARTs transmit FIFO buffer.

Refer to the SERIAL User's Manual (`SERIAL.PDF`) for more information regarding UARTs and serial ports.

`http://www.marshallsoft.com/serial.pdf`

2.7 PCMCIA Modems

The **Windows Serial Comm Library (WSC)** will work with PCMCIA modems (card modems for laptops). The PCMCIA drivers that come with the modem card must be properly installed.

The PCMCIA modem must appear as a normal serial port modem to Windows. The PCMCIA modem should come with several drivers, including a serial port driver and a modem driver.

2.8 Threads

The **Windows Serial Comm Library SDK (WSC)** is thread safe. Note that calling **SioEvent**, **SioEventChar**, or **SioEventWait** will block the thread. The blocking can be stopped by calling (from another thread)

```
SioSetInteger(Port, 'S', 1)        [C/C++, Delphi]
SioSetInteger(Port, ASC("S"), 1)   [BASIC, FoxPro, dBASE, Xbase++]
```

Refer to the WSC Reference Manual (WSC_REF) for more information on the **SioEvent**, **SioEventChar,** and **SioEventWait** functions.

2.9 SioPuts Notes

The WSC function to transmit a string or byte buffer, **SioPuts,** can operate in two ways: (1) "wait for completion" and (2) "immediate return".

In the first (default) way ("wait for completion"), **SioPuts** will not return to the caller until the entire contents of the buffer (passed to **SioPuts**) has been accepted by the Windows serial port driver. This means that the entire contents of the buffer will be transmitted, except perhaps for the last 16 bytes (for a 16550) in the UART's transmit FIFO, when **SioPuts** returns.

In the second way ("immediate return"), the contents of the buffer (passed to **SioPuts**) are queued in the transmit (TX) buffer within Windows, after which **SioPuts** returns. This means that **SioPuts** will return immediately, before any data is passed to the UART for transmission. This allows the calling program code to continue processing at the same time that the serial data is being transmitted.

In all cases, the value returned by **SioPuts** must be checked by the caller to determine the number of bytes actually accepted.

To enable the second method of operation (immediate return), before calling **SioPuts,** call the following:

```
SioSetInteger(Port, 'W', 1)        [C/C++, Delphi]
SioSetInteger(Port, ASC("W"), 1)   [VB, PowerBASIC, FoxPro, dBASE, Xbase++]
```

2.10 Communicating with Serial Devices

WSC can be used to communicate with any RS232 or multi-drop RS422 / RS485 serial port device. Most serial devices expect the computer to transmit commands to which the serial device replies. These commands are normally found in the technical documentation that accompanies the device.

For all serial devices, the required baud bate, number of data bits, parity, and number of stop bits being used by the serial device's port must be known and matched in the program that wants to communicate with the device.

2.11 RS485 and RS422 Ports

WSC can be used with multi-drop serial lines such as RS422 and RS485 serial ports. However, most RS422/485 serial ports require that RTS be raised before transmitting, and dropped before anyone else on the multidrop line can transmit. This usually means that one wants to determine when the last bit of the last byte has been transmitted.

Since Windows NT/2000/XP/Vista/Win7/Win8 does **not** allow direct access to hardware, one must wait long enough for the last bit to be sent. Once the TX queue becomes empty [**SioTxQue**(Port) = 0], up to 16 bytes may remain in the TX FIFO in the UART (for a 16550 UART with 16-byte FIFO's). Thus, one must wait about one millisecond per 1000 characters at 9600 baud, or 16 milliseconds for 16 bytes.

2.12 Hardware Flow Control & Blocking

If hardware flow control is set, and CTS is 0, your application will block until CTS is raised by the serial device (modem, etc.). To prevent blocking since there is no modem connected, do the following:

Set DTR. If DSR is 0, then you are NOT connected to a modem. If CTS is 0, then the modem has NOT enabled hardware flow control.

You may also set transmitter timeouts using the SioSetTimeouts function. For example, to set TX timeout = 1mSec/char + 2 seconds, call:

SioSetTimeouts(Port,(DWORD)-1,(DWORD)0,(DWORD)0,(DWORD)1,(DWORD)2000)

2.13 Virtual Serial Ports

A “virtual” serial port is a COM port that appears to be a real RS232 serial port to the Windows API (and thus to WSC), but is in reality a COM port emulator.

WSC works with all virtual port drivers. The most common virtual ports are those created for USB/RS232 serial port converters, Bluetooth, and virtual serial port drivers for particular devices.

Also see the following section "Overlapped I/O"

2.13.1 USB Ports

WSC cannot communicate directly with USB devices.

2.13.1.1 Communicating with USB Devices

Some USB devices come with a USB to RS232 interface so that the user can communicate with the USB device through a virtual COM port that the USB device driver creates. WSC will work with these virtual COM ports.

2.13.1.2 Communicating with RS232 Devices via USB

Very few new computers come with RS232 ports anymore. However, one can communicate with RS232 devices by using a "USB to RS232 Serial Port" converter. There are quite a few USB to RS232 serial port converters on the market, such as the Belkin “USB-to-Serial Portable Adaptor”.

A "USB to RS232 Serial Port" converter consists of two parts; (1) the USB/RS232 cable and (2) the USB/RS232 device driver. The USB/RS232 cable has a USB connector on one end that connects to the USB port on the computer and a RS232 connector (usually a DB9) on the other end that connects to the RS232 serial device. The Windows USB/RS232 driver converts data between RS232 and USB data.

The drivers for some USB to RS232 Serial Port converters do not implement the RESETDEV Windows API command. For this reason **SioReset** will **not** call the RESETDEV command unless it is specifically requested by calling **SioDebug** before calling **SioReset**:

```
SioDebug('R')      -- C/C++ and Delphi.
SioDebug(ASC("R")) -- BASIC, FoxPro, dBase, Xbase++, etc.
```

2.13.2 Bluetooth

Bluetooth is a short range radio technology used to communicate between devices located within a small area (typically less than 30 feet). Bluetooth is available for desktop/laptop Windows computers. Most Bluetooth adaptors use a USB port and are available in USB 1.1 and USB 2.0 configurations. Bluetooth adaptors are made by Kensington, IOGEAR, Belkin, LinkSys, and D-Link among others.

2.13.2.1 Bluetooth Serial Ports

Most Bluetooth adaptors include serial port emulation. The Bluetooth adaptor comes with a Windows device driver that emulates a serial port; that is, it creates a virtual serial port that looks like a real RS232 serial port to the Windows API, and thus to WSC.

After installing the Bluetooth adaptor, a new virtual serial port will become available. To verify the virtual COM port created, start the Windows Device Manager and look under "Ports".

Some Bluetooth serial port implementations (such as iPAQ PDA's) use an "inbound" and an "outbound" serial port. The Bluetooth device that initiates the serial connection between devices uses the outbound port (for serial transfer in both directions) and the other device uses its inbound port (also for both directions). Note that an application will never use both inbound and outbound ports at the same time.

2.13.2.2 Bluetooth GPS

A Bluetooth enabled computer can communicate with a Bluetooth enabled GPS receiver, such as the Teletype GPS and many others. The GPS receiver transmits NMEA 183 sentences over its Bluetooth serial port which can then be received by a Bluetooth enabled computer.

For more information on Bluetooth, see http://www.bluetooth.com

2.13.3 Other Virtual COM Ports

Some data devices come with a USB port used to connect to the USB port on your computer. In order to access the device data from an application program, a USB/serial driver is provided by the manufacturer of the data device that creates a virtual COM port.

2.14 Overlapped I/O

Overlapped I/O enables Windows to perform asynchronous I/O operations which can result in faster I/O times. Overlapped I/O was introduced in Windows NT & Windows 98.

However, some virtual serial port drivers (which implement the virtual port) do not work particularly well with overlapped I/O, so it is best to disable it in those cases. To disable overlapped I/O, call (just before calling SioReset)

```
SioSetInteger(Port, 'O', 0)
or SioSetInteger(Port, 79, 0)
or SioSetInteger(Port, WSC_OVERLAPPED, 0)
```

3 Modem I/O (MIO)

3.1 MIO Introduction

The Modem I/O functions ease communicating with modems using AT commands.

The MIO functions are broken down into parts called states, and control is returned to Windows between executing each state. This allows other code to run while waiting for MIO functions to complete.

For example (C/C++ Example):

(1) Send the string to the MIO driver by executing:

```
Code = mioSendTo(Port,100,"!ATDT1,800,555,1212")
```

The '!' characters is converted to a carriage return. The text string is copied into the driver's data area.

(2) Call **mioDriver** (typically based on a timer) until MIO_IDLE is returned. Each time **mioDriver** is called, it will send another character to the modem provided the required delay (since the previous character was sent) has passed. If the delay has not passed, the driver simply returns MIO_RUNNING, but without actually sending a character to the modem. Once all characters have been sent, **mioDriver** will return MIO_IDLE, indicating it is done and is ready to accept another function.

mioDriver will return MIO_RUNNING if it is still processing. Any other return value indicates that it is still processing and the returned value is a character from the modem that can be displayed if wanted.

(3) Once **mioDriver** returns MIO_IDLE, call **mioResult** to get the result of the **mioSendTo** call.

3.2 MIO Function Summary

- mioDriver : Drives execution of **mioSendTo**, **mioWaitFor**, or **mioQuiet** once started.
- mioBreak : Forces the MIO driver to IDLE state.
- mioSendTo : Sends a string (including control chars) to the modem.
- mioWaitFor : Waits for a particular string from the modem, passing all else through.
- mioQuiet : Waits for continuous quiet of a specified duration.
- mioBreak : Breaks further modem I/O activity.

Refer to the WSC Reference Manual (WSC_REF) for more information on individual MIO functions.

4 XMODEM and YMODEM

4.1 XYM Introduction

The XMODEM and YMODEM functions are implemented as DLL's (Dynamic Link Libraries), in XYM16.DLL, XYM32.DLL and XYM64.DLL, and are state driven like the Modem I/O functions described in the previous section.

Files can be sent and received using XMODEM, XMODEM/CRC, XMODEM/1K, and YMODEM. The basic procedure used to run XMODEM and YMODEM is as follows:

1. Call xyAcquire() immediately after calling SioReset().
2. To receive a file, call xyStartRx(), then call xyDriver() repeatedly until XY_IDLE is returned.
3. To send a file, call xyStartTx(), then call xyDriver() repeatedly until XY_IDLE is returned.
4. Call xyRelease() immediately before calling SioDone().

Refer to XMODEM.TXT for more information on the internal operation of the XMODEM protocol, and to YMODEM.TXT for more information on the internal operation of the YMODEM protocol. Also see the TERM example program.

4.2 XYM Function Summary

- xyAbort : Abort driver at any time.
- xyAcquire : Acquire a port.
- xyDebug : Set the debug level.
- xyDriver : Executes the next state or states.
- xyGetMessage : Get the next debug message.
- xyGetParameter : Get a driver parameter.
- xyGetFileName : Get name of file being sent or received.
- xyRelease : Release a port.
- xyStartRx : Start a receive.
- xyStartTx : Start a transmit.
- XySetString: Set parameter string.

Refer to the WSC Reference Manual (WSC_REF) for more information on individual XY Modem functions.

5 ASCII File Transfer

5.1 ASCII Protocol

The "ASCII Protocol" is not a defined protocol, but rather it is a loose set of conventions developed over a period of years by BBS operators. Because XON/OFF flow control is used, only ASCII text can be transferred. The ASCII protocol is implemented as a DLL and is state driven like the MIO and XYM code.

(1) Call **ascInit(**Port,RxQueSize,xFlag) to perform initialization, where

- Port = The connected port (COM1, COM2, etc).
- RxQueSize = The size of the Rx Queue as passed to **SioReset**().
- xFlag = 1 if this module will perform XON/XOFF flow control.
- Set to FALSE if flow control is being performed by the WSC code (you called **SioFlow**()).

(2) To send an ASCII file, call

ascStartTx(FileName,CharPace,TermChar,EchoFlag**),** where

- FileName = The path & name of the file to send.
- CharPace = The delay in milliseconds between characters.
- TermChar = The termination character to send after the file has been sent. If none, use 0x00.
- EchoFlag = 1 if screen echo is desired.

Then call **ascDriver**() until it returns IDLE (1).

(3) To receive an ASCII file, call

ascStartRx(FileName,TermChar,FirstWait,CharWait,EchoFlag), where

- FileName = The path & name of the file to send.
- TermChar = The termination character. If none, use 0x00.
- FirstWait = The maximum number of seconds to wait for the first incoming character.
- CharWait = The maximum number of seconds after which it is assumed that the other side has completed sending.
- If unsure, set this to 3 seconds.
- EchoFlag = 1 if screen echo is desired.

Then call **ascDriver**() until it returns IDLE (1).

5.2 ASCII Function Summary

- ascAbort : Abort driver at any time.
- ascDriver : Executes the next state or states.
- ascGetMessage : Get the next debug message.
- ascGetParameter : Get a driver parameter.
- ascGetFileName : Get name of file being sent or received.
- ascInit : Initialize the driver.
- ascStartRx : Start a receive.
- ascStartTx : Start a transmit.

6 Versions of WSC

The **Windows Standard Serial Communications (WSC)** library is available in three versions. All three versions have identical functionality.

6.1 Evaluation Version

The evaluation version can be differentiated from the other two versions by:

(1) The evaluation reminder screen is displayed at startup.

(2) The evaluation version may not be used for commercial purposes.

(3) The evaluation version can be used for 30 days.

6.2 Academic Version

The academic version can be differentiated from the other two versions by:

(1) There is no evaluation reminder screen.

(2) The academic version may not be used for commercial purposes.

DLLs purchased with the academic discount may not be distributed, and must be used for educational purposes only.

6.3 Professional Version

The professional version can be differentiated from the other two versions by:

(1) There is no evaluation reminder screen.

(2) Source code may be purchased for WSC, MIO, and XYM.

(3) Royalty free.

Your compiled DLLs may be distributed with your compiled applications as specified by the software license. However, neither the Keycode nor the source code (if purchased) to the DLLs can be distributed. The Professional version may be used for commercial purposes.

Licensing information is provided in Section 8.2. For details on ordering, refer to Section 1.3 "How to Purchase" or http://www.marshallsoft.com/order.htm.

7 Resolving Problems

The evaluation version example programs must be run on the same machine as on which SETUP was run.

Before attempting to write your own code, run several of the example programs.

(1) First, be sure that the proper Keycode is being passed to **SioKeycode**. See Section 2.1.

(2) If the registration reminder screen is still displayed after registering, the problem is that Windows is finding the evaluation DLL before the registered DLL. The solution is to delete (or zip up) all evaluation versions of the WSC32.DLL and WSC64.DLL .

(3) If "error -74" is received when calling **SioKeyCode**, the problem is that the keycode passed to **SioKeyCode** does not match the keycode in the DLL's. This is caused by using the evaluation keycode (value = 0) with the registered DLL.

(4) If you cannot get your application to run properly, first compile and run the terminal emulator program SIMPLE provided in the WSC distribution zip file. Test SIMPLE by connecting two computers with a null modem cable or by commanding a Hayes AT command set compatible modem.

Once SIMPLE runs successfully, compile and run the SELFTEST program. This program will test your serial ports' functionality.

(5) If your application does not run but SIMPLE and SELFTEST run correctly, then you have most likely made a programming mistake in your application. MarshallSoft Computing cannot debug your application, especially over the telephone! However, consider each of the following when searching for an error in your application.

(6) Are the receive and transmit buffers large enough? Use a buffer size that is twice the size of the largest expected block.

(7) Have you selected too high a baud rate? Windows can multi task many tasks at once. You may have to lower your baud rate (or get 16550 UART's).

(8) Did SioReset return a zero value? If not, then you must call SioReset again.

(9) Did you send the proper initialization string to your modem? Did you set DTR and RTS? (you should).

(10) Are you trying to link a 64-bit DLL to a 32-bit program (or vice versa)? Keep Win32 and Win64 development separate.

(11) If you terminate your program when running inside the Visual Basic environment, and you do not call SioDone first, you will leave VB itself with an open handle to your COM port. When you run again, you will get an "ACCESS DENIED" error and you must re-start VB itself.

We recommend the following steps if you believe that you have discovered a bug in the library:

(1) Create the smallest, simplest test program possible that demonstrates the problem.
(2) Document your exact machine configuration and what error the test program demonstrates.
(3) Email us the example source.

If the problem is an error in the library and can be solved with an easy work-around, we will publish the work-around. If the problem requires a modification to the library, we will make the change and make the modified library available to our customers without charge.

8 Legal Issues

8.1 License

This license agreement (LICENSE) is a legal agreement between you (either an individual or a single entity) and MarshallSoft Computing, Inc. for this software product (SOFTWARE). This agreement also governs any later releases or updates of the SOFTWARE. By installing and using the SOFTWARE, you agree to be bound by the terms of this LICENSE. If you do not agree to the terms of this LICENSE, do not install or use the SOFTWARE

MarshallSoft Computing, Inc. grants a nonexclusive license to use the SOFTWARE to the original purchaser for the purposes of designing, testing or developing software applications. Copies may be made for back-up or archival purposes only. This product is licensed for use by only one developer at a time. All developers working on a project that includes a MarshallSoft Software SDK, even though not working directly with the MarshallSoft SDK, are required to purchase a license for that MarshallSoft product.

The "academic" registered DLL's may not be distributed under any circumstances, nor may they be used for any commercial purpose.

The "professional" registered DLL's may be distributed (royalty free) in object form only, as part of the user's compiled application provided the value of the Keycode is not revealed. The registered DLL's may NOT be distributed as part of any software development system (compiler or interpreter) without our express written permission. The source code if purchased for the library (WSC32.C, WSC64.C, MIO.C, and XYM.C) is copyrighted by MarshallSoft Computing and may not be released in whole or in part.

8.2 Warranty

MARSHALLSOFT COMPUTING, INC. DISCLAIMS ALL WARRANTIES RELATING TO THIS SOFTWARE, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ALL SUCH WARRANTIES ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. NEITHER MARSHALLSOFT COMPUTING, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THIS SOFTWARE SHALL BE LIABLE FOR ANY INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH SOFTWARE EVEN IF MARSHALLSOFT COMPUTING, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR CLAIMS. IN NO EVENT SHALL MARSHALLSOFT COMPUTING, INC.'S LIABILITY FOR ANY SUCH DAMAGES EVER EXCEED THE PRICE PAID FOR THE LICENSE TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM. THE PERSON USING THE SOFTWARE BEARS ALL RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE.

Some states do not allow the exclusion of the limit of liability for consequential or incidental damages, so the above limitation may not apply to you.

This agreement shall be governed by the laws of the State of Alabama and shall inure to the benefit of MarshallSoft Computing, Inc. and any successors, administrators, heirs and assigns. Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction located in Madison County, Alabama. The parties hereby consent to in personam jurisdiction of said courts.

9 WSC Function Summary

The WSC Reference Manual (WSC_REF.PDF) provides detailed information on the WSC functions as well as a list of WSC error codes. A one line summary of each WSC function is included here.

There are 46functions in the WSC library.

```
SioBaud          Sets the baud rate of the selected port.
SioBrkSig        Asserts, cancels, or detects BREAK signal.
SioByteToShort   Converts 8-bit ASCII to 16-bit Unicode ASCII.
SioCountWait     Waits for a specified number of bytes before returning.
SioCRC16         Computes the 16-bit CCITT CRC.
SioCRC32         Computes the 32-bit CCITT CRC.
SioCTS           Reads the Clear to Send (CTS) modem status bit.
SioDCD           Reads the Data Carrier Detect (DCD) modem status.
SioDebug         Set and/or reads debug data.
SioDone          Terminates further serial processing.
SioDSR           Reads the Data Set Ready (DSR) modem status bit.
SioDTR           Set, clear, or read the Data Terminal Ready (DTR).
SioEvent         Waits for specified event.
SioEventChar     Waits for specified character in serial input stream.
SioEventWait     Waits for specified event until timeout occurs.
SioFlow          Enables / disables hardware flow control.
SioGetc          Reads the next character from the serial line.
SioGetReg        Gets registration string
SioGets          Receives a string of characters.
SioHexView       Constructs hexadecimal string.
SioInfo          Returns information such as library version.
SioKeycode       Passes the "key code" value to the DLL.
SioMessage       Post message when event occurs.
SioParms         Sets parity, stop bits, and word length.
SioPutc          Transmit a character over a serial line.
SioPuts          Transmits a string of characters.
SioQuiet         Waits until line is quiet (no incoming data).
SioRead          Reads any UART register.
SioReset         Initialize a serial port for processing.
SioRI            Reads the Ring Indicator (RI) modem status bit.
SioRTS           Sets, clears, or reads the Request to Send (RTS)
SioRxClear       Clears the receive buffer.
SioRxQue         Returns the number of characters in the RX queue.
SioRxWait        Waits for a specified number of incoming characters.
SioSetInteger    Sets port specific integer parameters.
SioSetTimeouts   Sets TX and RX timeouts.
SioShortToByte   Converts 16-bit Unicode ASCII to 8-bit ASCII.
SioSleep         Sleeps specified number of milliseconds.
SioStatus        Returns the serial port line status.
SioTimeMark      Returns the # milliseconds since bootup.
SioTimer         Returns the system times in milliseconds.
SioTxClear       Clears the transmit buffer.
SioTxQue         Returns the number of characters in the TX queue.
SioUnGetc        "Ungets" (puts back) a specified character.
SioWaitFor       Waits for incoming byte.
SioWinError      Returns Win32 error message as text.
```

There are also 18 additional functions for Modem I/O.

10 Error Codes

10.1 WSC Error Codes

[NAME]	:	[FUNCTION]
WSC_ABORTED	:	The evaluation version of WSC corrupted.
WSC_BUFFERS	:	Cannot allocate memory for buffers.
WSC_EXPIRED	:	Evaluation version expired.
WSC_KEYCODE	:	Bad key code value.
WSC_NO_DATA	:	No incoming serial data is available.
WSC_RANGE	:	A parameter is out of range.
WSC_THREAD	:	Cannot start thread.
WSC_WIN32ERR	:	Win32 system error.
WSC_IE_BADID	:	No such port.
WSC_IE_BAUDRATE	:	Unsupported byte size.
WSC_IE_BYTESIZE	:	Unsupported byte size.
WSC_IE_DEFAULT	:	Error in default parameters
WSC_IE_HARDWARE	:	COM port hardware not present
WSC_IE_MEMORY	:	Cannot allocate memory.
WSC_IE_NOPEN	:	Port not opened. Call SioReset first.
WSC_IE_OPEN	:	Port already opened.
WSC_IO_ERROR	:	An I/O event error has occurred.

The WSC_ABORTED error occurs in the evaluation version only if there is a problem displaying the software info screen.

The WSC_WIN32ERR error code is returned only for Win32 system errors. Call **SioWinError** to retrieve the error message.

10.2 XYM Error Codes

Error codes are always negative, except for "no error". Most of these error conditions rarely occur. Also note that XYDRIVER functions can return WSC errors. An error message is queued when an error occurs which can be retrieved with **xyGetMessage**.

[NAME]	:	[FUNCTION]
XY_NO_ERROR	:	No error.
XY_UNKNOWN_ERROR	:	Unknown error.
XY_ALREADY_ACTIVE_ERROR	:	Port already acquired.
XY_CANNOT_OPEN_ERROR	:	Cannot open specified file.
XY_EMPTY_FILE_ERROR	:	Specified file is empty.
XY_NO_STARTUP_CHAR_ERROR	:	Must specify NAK, 'C', or 'G'.
XY_NOT_NCG_ERROR	:	Expected NAK, 'C', or 'G'.
XY_DISK_READ_ERROR	:	Error reading disk.
XY_NO_EOT_ACK_ERROR	:	EOT was not ACK'ed.
XY_INTERNAL_ERROR	:	Internal error!
XY_CANCELLED_ERROR	:	Other side canceled.
XY_OUT_OF_SYNC_ERROR	:	Protocol has lost synchronization.
XY_RETRIES_ERROR	:	Packet retry limit was exceeded.
XY_BAD_PACKET_NBR_ERROR	:	Incorrect packet number.
XY_TIMED_OUT_ERROR	:	Timed out waiting for other side.
XY_NO_SUCH_FILE_ERROR	:	No such file.
XY_NOT_ACTIVE_ERROR	:	Port not acquired by xyAcquire.
XY_PORT_RANGE_ERROR	:	Port number out of range.

The numerical value for each error code is listed in the file wscErrors.txt located in the \DOCS subdirectory.