Windows Standard

Serial Communications

for Visual Basic

Programmer's Manual

(WSC_4VB)

Version 6.0

March 17, 2017

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

Copyright (C) 2017
All rights reserved

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

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

MARSHALLSOFT is a registered trademark of MarshallSoft Computing.

TABLE OF CONTENTS

1 Introduction

The **Windows Standard Serial Communications Library for Visual Basic (WSC4VB)** is a toolkit that allows software developers to quickly develop 32-bit and 64-bit serial communication applications in Visual Basic or Visual Studio (VB.NET).

The **Windows Standard Serial Communications Library** (WSC) is a component DLL library used to create serial communications programs that access data from a serial port using RS232 or multi-drop RS422 or RS485 ports. **WSC** also supports virtual serial ports using Bluetooth serial and USB to serial converters. The **WSC** component library uses the Windows API for all communication and can be used to easily write applications to control serial devices such as barcode scanners, card readers, modems, lab instruments, medical devices, USB serial devices, scales, GPS navigation, etc.

The **Windows Serial Communications Library for Visual Basic (WSC4VB)** supports and has been tested with several Visual Basic compilers including Microsoft VB 4.0 through VB 6.0, Microsoft Visual Studio .NET Framework and Microsoft Visual Studio through Visual Studio 2015. WSC can also be used with any VBA (Visual Basic for Applications) language such as Excel, Access, MS Office, etc. WSC also works with PowerBuilder.

The **Windows Standard Serial Communications Programmer's Manual** provides information needed to compile programs in a Visual Basic or VB.NET programming environment.

WSC4VB includes more than 25 Visual Basic and Visual Studio (VB.NET) example programs with source that demonstrate serial port communications functions.

The **Windows Standard Communications Library SDK** includes Win64 and Win32DLLs (WSC64.DLL and WSC32.DLL). The DLLs can also be used from any language (C/C++, .NET, Delphi, Visual FoxPro, COBOL, Xbase++, dBASE, etc.) capable of calling the Windows API. **WSC4VB** runs under all versions of Windows through Windows 10.

When comparing the **Windows Standard Serial Communications Library** against our competition, note that:

1. WSC4VB is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. Win64 and Win32 DLLs are included.
3. WSC4VB does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
4. WSC is fully threadable.
5. The WSC functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **Windows Standard Serial Communications Library** for Visual C/C++ (WSC4C), Delphi (WSC4D), PowerBASIC (WSC4PB), Visual FoxPro (WSC4FP), dBASE (WSC4DB), and Xbase++ (WSC4XB). All versions of WSC use the same DLLs (WSC64.DLL and WSC32.DLL). However, the examples provided for each version are written for the specified computer programming language.

The latest versions of the **Windows Standard Serial Communications Library (WSC)** can be downloaded from our web site at
http://www.marshallsoft.com/serial-communication-library.htm

Our goal is to provide a robust serial communication library component that you and your customers can depend upon. A fully functional evaluation version is available. Contact us if you have any questions.

1.1 Features

Some of the many features of the **Windows Serial Communications Library for Visual Basic (WSC4VB)** are:

- Comes with 32-bit and 64-bit DLLs.
- Can control any serial device (scale, barcode reader, etc.) connected to the serial port.
- Can be used from GUI mode or console mode programs.
- Can control up to 256 ports simultaneously.
- Can be used with virtual serial ports using Bluetooth serial or a USB to serial converter.
- Includes 49 functions plus modem control.
- Comes with ANSI emulation and ASCII, XMODEM and YMODEM.
- Supports RS232, and multidrop RS422, and RS485 ports.
- Supports hardware and software flow control.
- Supports any baud rate.
- Ability to specify the parity, word size, and number of stop bits.
- Supports binary and text data transfer.
- Port re-entrant.
- Is fully threadable.
- Supports character peek (**SioEventChar**).
- Supports transmit and receive timeouts.
- Can send Windows messages on completion of events (incoming character, etc.)
- **Free** technical support for one year.
- License covers all programming languages.
- Royalty free distribution with a compiled application. There are no run time fees.
- Evaluation versions are fully functional. No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Supports Windows XP through Windows 10.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Is native Windows code but can also be called from managed code.
- Will run on machines with or without .NET installed
- Supports all versions of Visual Basic, from V3.0 through Visual Studio 2015.
- Works with Microsoft Visual Studio .NET Framework.
- Works with PowerBuilder.
- Can be used with VBA (Visual Basic for Applications) such as Excel, Microsoft Office and Access.
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions such as Visual C++, Visual C++ .NET, Visual FoxPro, Delphi, Xbase++, dBASE, COBOL, Access and Excel.
- Can be purchased with or without ANSI C source code to the WSC DLLs.
- Purchase a developer license for WSC4VB and use the DLLs with any other development environment (C++, Visual FoxPro, etc).
- Updates are **free** for one year (updates to source code are separate).
- Documentation online as well as in printable format.

A good selection of Visual Basic (and VB.Net) example programs with full source code is included. Refer to Section 4 and 5 for more details on each of the example programs.

```
WSCVER   :  Program that displays the WSC version number.
EASY     :  A simple terminal emulator program.
SELFTEST :  Performs COM port functionality testing.
MODEM    :  Same as EASY but controls flow control, modem lines, etc.
TERM     :  Terminal emulator with XMODEM, YMODEM, and ANSI support.
FINDER   :  Finds a modem connected to one of the serial ports.
LISTER   :  Lists all serial ports.
ATOK     :  Sends "AT" to modem, uses WSC class (WscClass.cls).
DEVICE   :  Sends ASCII text string to serial device.
ProXR    :  Reads relays on ProXR device.
MESSAGE  :  Send Windows message when incoming bytes are detected.
XMS/XMR  :  XMODEM Send & XMODEM Receive programs.
YMS/YMR  :  YMODEM Send & YMODEM Receive programs.
PUTTYPE  :  Transmits user defined data type.

VERSION  :  Visual Studio version of the WSC version program (WSCVER).
EASY     :  Visual Studio version of the simple.frm terminal program.
FINDER   :  Visual Studio version of finder.frm program.
SELFTEST :  Visual Studio version of the COM port functionality program.
XMS      :  Visual Studio version of the XMODEM/Send program.
XMR      :  Visual Studio version of the XMODEM/Receive program.
ReadGPS  :  Read GPS sentences (Visual Studio)
```

1.2 Documentation Set

The complete set of documentation consists of four manuals. This is the first manual (WSC_4VB) in the set.

- WSC_4VB Programmer's Manual (WSC_4VB.PDF)
- WSC User's Manual (WSC_USR.PDF)
- WSC Reference Manual (WSC_REF.PDF)
- Serial Communications User Manual (SERIAL.PDF)

The WSC_4VB Programmer's Manual is the language specific (Visual Basic and VB.NET) manual. All language dependent programming issues are discussed in this manual. Information needed to compile your programs in a Visual Basic programming environment is provided in this manual.

The WSC User's Manual (WSC_USR) discusses language independent serial communications programming issues including modem control. Purchasing and license information is also provided.

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

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

Each manual comes as an Adobe PDF file.

The documentation is also provided on our web site at

http://www.marshallsoft.com/wsc4vb.htm

1.3 Example Program

The following example program segment sets DTR (Data Terminal Ready).

```
' pass the keycode (0 for evaluation version)
Code = SioKeyCode(WSC_KEY_CODE)
' open port COM1
Code = SioReset(COM1, 512, 512)
If Code < 0 Then
' process error
  Call SayError(EXAMPLE, Code)
  Call ShutDown
  Exit Sub
End If
' set DTR
Code = SioDTR(COM1, ASC("S"))
```

Refer to Sections 4.0, 5.0 and 6.0 for complete examples with source.

1.4 Installation

(1) Before installation of WSC4VB, your Visual Basic compiler should already be installed on your system and tested. Note that Visual Basic 4.0 or above is required in order to create Win32 programs.

(2) Unzip WSC4VB60.ZIP, (evaluation version) or WSCxxxxx.ZIP (where xxxxx is the developer's Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE which will install all WSC4VB files, including copying WSC32.DLL and WSC64.DLL to the Windows directory.

Note that no DLL registration is required.

WSC4VB comes configured for 32-bit Visual Basic (VB4 and above). VB 4.0, VB 5.0 and VB 6.0 project filenames end with "32.vbp" and VB.NET and Visual Studio project filenames end with "vbproj."

1.5 Uninstalling

Uninstalling WSC4VB is very easy. First, delete the WSC4VB project directory created when installing WSC4VB. Next delete WSC64.DLL and WSC32.DLL from the Windows directory, typically C:\WINDOWS.

1.6 Pricing

A developer license for WSC4VB can be registered for $115 (or $195 with ANSI C source code to the library DLL's). Purchasing details can be found in the WSC User's Manual (WSC_USR), Section 1.3, "How to Purchase" (http://www.marshallsoft.com/wsc_usr.pdf). Also see INVOICE.TXT provided with the evaluation person or order directly on our web site at

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

1.7 Updates

When a developer license is purchased for WSC4VB, the developer will received a new set of registered DLLs plus a license file (WSCxxxx.LIC). The license file is needed to download updates to the registered DLL's 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. The developer 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.
- $75 if the update is ordered after three years of the original purchase.

If source code was previously purchased, updates to the source code can be purchased for $40 along with the DLL update. Note that the registered DLL's never expire.

2 Library Overview

The **Windows Standard Serial Communications Library (WSC)** has been tested on multiple computers running Windows XP through Windows 10.

The WSC4VB library has also been tested with several Visual Basic compilers, from VB 3.0 through VB 6.0, Microsoft Visual Basic .NET (**VB.Net**.) and Microsoft Visual Studio (2005, 2008, 2010, 2012, 2013, 2015). WSC can also be used with any VBA language such as Excel, Access, MS Office as well as PowerBuilder.

The SETUP installation program will copy the DLLs to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the WSC4VB files are copied to the directory specified (default \WSC4VB). Three sub-directories are created, as follows:

```
DOCS - All documentation files
APPS - All example code
DLLS - All DLL's
```

2.1 Dynamic Link Libraries

The **WSC4VB serial communication library component** includes Win64 and Win32 dynamic link libraries (DLL). 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 the traditional static library that is bound to each and every application that uses it at link time.

An important advantage that DLLs have over other "popular" library formats such as VBX or OCX is that DLLs are callable by all Windows applications. Since DLLs are the building blocks of the Windows Operating System, they will not be replaced by a "newer technology".

The following files can be found in the DLL sub-directory when SETUP is run:

```
wsc32.dll   - Win32 version of WSC
wsc64.dll   - Win64 version of WSC
```

2.2 Keycode (License Key)

Each WSC DLL has a keycode encoded within it. The keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.BAS and KEYCODE.VB. The keycode for the evaluation version is 0. The developer will receive a new keycode and a set of new DLL's after purchasing a license. The KEYCODE is passed to **SioKeyCode**.

If you get an error message (value -108) when calling **SioKeyCode**, it means that the keycode in your application does not match the keycode in the DLL. After registering, it is best to remove the evaluation version of the WSC32.DLL and WSC64.DLL from the Windows search path or delete them.

2.3 Dynamic Strings

2.3.1 Passing VB Strings to DLL Functions

When passing a string to a DLL function, the DLL looks for a null character to terminate the string. This null character CHR(0) is normally present in Visual Basic strings, but can be lost if the string is modified by Visual Basic at runtime. This problem can be overcome by appending CHR(0) to the end of strings passed to **WSC** functions.

2.3.2 Passing VB String Buffers to DLL Functions.

The Visual Basic language uses a technique known as "garbage collection" to manage string space at runtime, and may be called internally at any time by the Visual Basic runtime, asynchronous to what you may be doing in your code.

When passing a string buffer to a DLL function into which text will be copied, it is strongly recommended that the local string be allocated immediately before use. For example, a string buffer is passed to the dllGetMessage function, which copies a text message into it. Note that `SPACE(80)` is called immediately before dllGetMessage.

```
Dim Code As Integer
Dim Buffer As String * 80
' allocate buffer just before call to dllGetMessage
Buffer = SPACE(80)
' copy message into 'Buffer'
Code = dllGetMessage(Buffer, 80)
' message text is now in 'Buffer'
```

This technique is not necessary for passing a string to a DLL function, only when passing a buffer to a DLL into which data is to be placed by the DLL function.

2.4 Limitations on COM Ports

WSC4VB can use any port from COM1 to COM256, provided that the port is known to Windows 95/98/NT/2000/2003-2012/Me/XP/Vista/Windows 7/Windows 8/ Windows 10 and there is physical hardware present.

2.5 Error Display

The error message text associated with WSC error codes can be displayed by calling the function **SayError**, which can be found in the files **Errors32.bas**. Each sample program contains examples of error processing.

2.6 Waiting for New Serial Data

All serial data is moved from the UART's buffer to the receive queue in memory (by the Windows serial port driver) under interrupt control. Similarly, all out going serial data is moved to the transmit queue in memory.

There are several methods that can be used to wait for new incoming serial data, as follows:

2.6.1 Polling Method: For Win32/Win64 programs (including VB.NET).

The most straightforward method is to use a VB timer to check every so often. The timer interval should be set between 50 milliseconds and 250 milliseconds. Setting it much less than 50 milliseconds would consume considerable system resources polling, and setting it greater than 250 milliseconds will result in sluggish menu response times. A good compromise is to set the timer interval to 125 milliseconds.

2.6.2 Message Method: For Win32 programs (VB 4 and above).

The "Message Method" is probably the most natural method to use with Visual Basic (recall that WSC works with many different computer languages). In this method, the SioMessage function is called which sends a Windows message to a VB button when new data is ready to be read.

2.6.3 Event Method: For Win32/Win64 programs (VB 5).

This method uses SioEvent in a thread (background process) which blocks (efficiently waits) until new data is available. This method requires creating a thread in VB. However, although it is possible to create a thread in VB5, it is not recommended since VB5 is not thread safe.

2.6.4 EventWait Method: For Win32/Win64 programs (VB 4 and above, including VB.NET).

This method uses the SioEventWait function in a timer procedure to block (efficiently wait) until new data is available or until the timer period expires. A good choice for the timer interval is 250 milliseconds because it allows quick response to user input and also at the same time minimizes polling. Note that a 250 millisecond interval represents quite a large interval of CPU time.

2.7 SioEvent Logic

SioEvent, **SioEventChar**, and **SioEventWait** will block until the specified event occurs. If a call to **SioEvent**, **SioEventChar**, or **SioEventWait** is placed in a thread, then the thread will block but the application calling the thread will not.

SioEventWait is used in the EASY example program.

2.8 Virtual Serial Ports

A "virtual" serial port is 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.

The two most common virtual ports are those created for USB/serial port converters and Blue Tooth. WSC will work with most USB to serial port converts and with Bluetooth serial.

More information about virtual serial ports can be found in WSC User's Manual (WSC_USR), Section 2.13 (http://www.marshallsoft.com/wsc_usr.pdf).

2.9 WSC Class

The WSC class "wscClass" (wscClass.cls) is a Visual Basic class wrapper for making calls to WSC32.DLL and WSC64.DLL. The class name for each function is the same as the DLL function, except the leading "**wsc**" is replaced by "**f**".

Those functions that return strings do so by use of the "String Result" property. Instantiate **wscClass** as any other class in VB:

```
Dim X As New wscClass
```

The use of **wscClass** is limited to Visual Basic 5.0 and above since previous versions of Visual Basic do not support classes. Refer to the ATOK example project for an example of using **wscClass**

2.10 Using Microsoft Visual Studio (VB.Net)

There are a few differences between VB 4/5/6 and Visual Studio that affect writing programs that use WSC.

(1) Variables that are declared “As Long” in VB 4/5/6 are declared “As Integer” in Visual Studio (VB.NET).

(2) Fixed length strings are not supported in VB.Net. When calling any WSC function that can return a string (SioGets, SioPuts, and SioWinError), memory for the string variable must be allocated first. For example:

```
Buffer = Space(80) ' allocate memory for "Buffer"
Code = SioWinError(Buffer, 80)
```

(3) Some VB functions must be fully qualified. For example, instead of LEFT, use `Microsoft.VisualBasic.Left`

(4) The module WSC32.VB (not WSC32.BAS) must be included in all VB.NET programs. For Win64 programs, include WSC64.VB instead.

2.11 Visual Basic for Applications (VBA)

The Windows Serial Communications Library (WSC) component library can be used with Microsoft VBA applications such as EXCEL, ACCESS, and Microsoft Office. The VBA example module is vba32ATOK.bas and vba64ATOK.vb. The file vba32Declare.bas and vba64Declare.vb contains all constants and function declarations for using WSC4VB in VBA applications.

2.11.1 ACCESS

1. Start ACCESS
2. Open a table
3. Click "Database Tools" then double click "Visual Basic".
4. In the "Microsoft Visual Basic for Applications" window, click "Insert" then "Module"
5. Paste the VBA code.
6. Click "Run" on the VBA menu, then "Run Sub"
7. When the "Macros" menu is displayed, enter the VBA coder name then CR.

2.11.2 EXCEL and WORD

If "Developer" is not displayed on the ribbon line:

1. Start EXCEL (or WORD)
2. Click "File" then "Options"
3. Click "Customize Ribbon" in " Options" window.
4. Click "Macros"
5. Check "Developer"
6. Click "OK"

To add VBA code.

1. Start EXCEL or WORD)
2. Click "Developer"
3. Double-click "Visual Basic".
4. In the "Microsoft Visual Basic for Applications" window, click "Insert" then "Module"
5. Paste the VBA code.
6. Click "Run" on the VBA menu, then "Run Sub"
7. When the "Macros" menu is displayed, enter the VBA coder name then CR.

2.12 PowerBuilder

WSC can also be used with Power Builder applications. Refer to PBUILDER.TXT in the \APPS subdirectory for more information.

`WSC.PBI` : Power Builder declaration file.

2.13 Adding WSC4VB to a Project

To be able to call WSC functions from an application, the WSC declaration file needs to be in the same directory (folder) as the application program using WSC. Copy WSC32.BAS (if running VB 4/5/6), WSC32.VB (if running 32-bit VB.Net/Visual Studio) or WSC64.VB (if running 64-bit Visual Studio) to the application directory. WSC declaration files are located in the APPS directory (folder) created when WSC was installed, usually C:\WSC4VB\APPS.

2.13.1 Adding WSC4VB to a VB 3.0 Project

Open the existing project with "File", "Open Project". Then choose "File", "Add File", then add WSC16.BAS and KEYCODE.BAS to the project. WSC functions can now be called from your Visual Basic program.

2.13.2 Adding WSC4VB to a VB 4.0, 5.0, or 6.0 Project

Open your existing project with "File", "Open Project". Then choose "Insert", "Module", then add WSC32.BAS and KEYCODE.BAS to your project. If prompted to add "DAO 2.50 Object Library", choose "no". WSC functions can now be called from your VB program.

2.13.3 Adding WSC4VB to a Visual Basic .NET Project

Open your existing project with "File", "Open Project". Then choose "Project", "Add Module", then add WSC32.VB and KEYCODE.VB to your project. WSC functions can now be called from your VB.NET program.

2.13.4 Adding WSC4VB to a Visual Studio Project

Open the existing project with "File", "Open Project". Then choose "Project", "Add Module", then add modules WSC32.VB (or WSC64.VB) and KEYCODE.VB to your project. If empty modules are created, replace the contents of these modules with the contents of the modules of the same name provided by us.

WSC functions can now be called from your Visual Studio VB program.

2.14 Using 16-bit Visual Basic

Support for Win16 was dropped beginning with version 5.2. Version 5.1 is still available (free when purchasing the current version) for those wanting Win16 support.

2.15 Explicitly Loading a WSC DLL

When an application program runs that makes calls to a WSC DLL (WSC64.DLL or WSC32.DLL), the Windows operating system will locate the WSC DLL by searching the directories as specified by the Windows search path. If the WSC DLLs are placed in the \WINDOWS directory (or \WINNT for Windows NT/2000), they will always be found by Windows.

WSC32.DLL can be loaded from an explicit location by replacing "WSC32.DLL" in WSC32.BAS or WSC32.VB by the full path. For example, to load WSC32.DLL from C:\WSC4VB\APPS, the first entry would be:

```
Declare Function SioBaud Lib "C:\WSC4VB\APPS\WSC32.DLL"
  (ByVal Port As Integer, ByVal BaudCode As Integer) As Integer
```

The above also applies to WSC32.DLL as well as WSC64.DLL.

2.16 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as it were on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as WSC32.DLL. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

2.16.1 Visual Studio Visual Basic: Versions 2005 through 2015

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click "Build", then "Configuration Manager".
Click the drop-down button below "Active Solution Platform".
Click <New...>, then change "Any CPU" to "x86".

2.16.2 Visual Studio C++: Versions 2005 through 2015

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.17 Visual Basic Problems

If you terminate your program when running inside the Visual Basic environment, and you do not call **SioDon**e 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.

Also refer to Section 7 "Resolving Problems" in the User's Manual (WSC_USR.PDF) or online at http://www.marshallsoft.com/wsc_usr.pdf.

.

3 Compiler Issues

The **Windows Serial Communications Library for Visual Basic** supports and has been tested with all versions of Microsoft Visual Basic including:

- Visual Basic 3, 4, 5, 6
- Visual Studio .NET
- Visual Studio .NET 2003
- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010
- Visual Studio 2012
- Visual Studio 2013
- Visual Studio 2015

WSC4VB also supports Visual Basic for Applications (VBA) and PowerBuilder.

3.1 Visual Basic Project Files

3.1.1 Visual Studio

Visual Studio (32-bit and 64-bit) uses the file extension ".vbproj" for project files.

3.1.2. Visual Basic 4, 5, 6.

Visual Basic 4, 5 and 6 (32-bit) use the file extension ".vbp" for project files.

3.2 Compiling Example Programs

3.2.1 Compiling Visual Studio (VB.NET) Programs

Example programs designed for Visual Studio (VB.NET) end with the extension ".vbproj". The 32-bit VS2008 specific project files will contain the string "(VS2008)" in their project filename. The 64-bit VS2008 specific project files will contain the string "(VS2008)x64" in their project filename. For example, the Visual Studio project files for the WSC Version program (`wscver.vb`) are:

```
wscver.vbproj              Visual Studio (all 32-bit versions)
wscver(VS2008).vbproj      Visual Studio 2008 (32-bit)
wscver(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
wscver(VS2010).vbproj      Visual Studio 2010 (32-bit)
wscver(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
wscver(VS2012).vbproj      Visual Studio 2012 (32-bit)
wscver(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
wscver(VS2013).vbproj      Visual Studio 2013 (32-bit)
wscver(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

3.2.2 Compiling Visual Basic 4/5/6 (32-bit) Programs

Example programs designed for Visual Basic 4, 5, and 6 end with the extension ".vbp". The project file is stored in VB 4.0 format so that it can be loaded from VB 4.0, VB 5.0, and VB 6.0. For example, the Visual Basic project file for the WSC Version program (wscver32.frm) is:

```
wscver32.vbp
```

When saving the example programs in VB version 5.0 or VB 6.0 format, answer "no" if asked to add the "Microsoft DAO v2.5 library".

3.2.3 Compiling Visual Basic 3 (16-bit) Programs

Support for Win16 was dropped beginning with version 5.2. Version 5.1 is still available (free when purchasing the current version) for those wanting Win16 support.

3.3 Compiling WSC Source

WSC32.DLL and WSC64.DLL were written in standard ANSI C and have been compiled using Microsoft C. Source code (ANSI C) for the WSC library is provided in the registered version (if ordered) only.

For more information on the C/C++ version of WSC, download the latest version of WSC4C from our web site at http://www.marshallsoft.com/wsc4c.htm

4 Visual Studio (VB.Net) Examples

All Visual Studio project files end with the extension ".vbproj".

4.1 WSCVER.VB

The WSCVER example program displays the WSC library version number & build and verifies that WSC DLL can be found and loaded by Windows at runtime. The project files are

```
wscver.vbproj             Visual Studio (all 32-bit versions)
wscver(VS2008).vbproj     Visual Studio 2008 (32-bit)
wscver(VS2008)x64.vbproj  Visual Studio 2008 (64-bit)
wscver(VS2010).vbproj     Visual Studio 2010 (32-bit)
wscver(VS2010)x64.vbproj  Visual Studio 2010 (64-bit)
wscver(VS2012).vbproj     Visual Studio 2012 (32-bit)
wscver(VS2012)x64.vbproj  Visual Studio 2012 (64-bit)
wscver(VS2013).vbproj     Visual Studio 2013 (32-bit)
wscver(VS2013)x64.vbproj  Visual Studio 2013 (64-bit)
```

4.2 EASY.VB

The EASY example program is a simple terminal program. Everything typed at the keyboard is sent out the serial port, and everything incoming from the serial port is displayed on the screen
The easiest way to test EASY is to connect to a modem. Typing 'AT' should result in an 'OK' being displayed.

A null-modem cable can also be used to connect two computers together with their serial ports. Run EASY on both machines. Whatever is typed on one machine will be displayed on the other.

The EASY project files are

```
easy.vbproj             Visual Studio (all 32-bit versions)
easy(VS2008).vbproj     Visual Studio 2008 (32-bit)
easy(VS2008)x64.vbproj  Visual Studio 2008 (64-bit)
easy(VS2010).vbproj     Visual Studio 2010 (32-bit)
easy(VS2010)x64.vbproj  Visual Studio 2010 (64-bit)
easy(VS2012).vbproj     Visual Studio 2012 (32-bit)
easy(VS2012)x64.vbproj  Visual Studio 2012 (64-bit)
easy(VS2013).vbproj     Visual Studio 2013 (32-bit)
easy(VS2013)x64.vbproj  Visual Studio 2013 (64-bit)
```

4.3 SELFTEST.VB

The SELFTEST example program performs a serial port I/O functionality test. Either a pair of ports on the same computer (using a null modem cable) or a single port (using a loopback adapter) can be tested.

Refer to LOOPBACK.TXT in the **DOCS** directory for an explanation of how to make a loopback adapter (without tools!).

The SELFTEST project files are

```
selftest.vbproj             Visual Studio (all 32-bit versions)
selftest(VS2008).vbproj     Visual Studio 2008 (32-bit)
selftest(VS2008)x64.vbproj  Visual Studio 2008 (64-bit)
selftest(VS2010).vbproj     Visual Studio 2010 (32-bit)
selftest(VS2010)x64.vbproj  Visual Studio 2010 (64-bit)
selftest(VS2012).vbproj     Visual Studio 2012 (32-bit)
selftest(VS2012)x64.vbproj  Visual Studio 2012 (64-bit)
selftest(VS2013).vbproj     Visual Studio 2012 (32-bit)
selftest(VS2013)x64.vbproj  Visual Studio 2012 (64-bit)
```

4.4 FINDER.VB

The FINDER program searches for a connected modem. Your modem must be connected to one of COM1, COM2, …,COM256 and must be turned on.

The FINDER project files are

```
finder.vbproj             Visual Studio (all 32-bit versions)
finder(VS2008).vbproj     Visual Studio 2008 (32-bit)
finder(VS2008)x64.vbproj  Visual Studio 2008 (64-bit)
finder(VS2010).vbproj     Visual Studio 2010 (32-bit)
finder(VS2010)x64.vbproj  Visual Studio 2010 (64-bit)
finder(VS2012).vbproj     Visual Studio 2012 (32-bit)
finder(VS2012)x64.vbproj  Visual Studio 2012 (64-bit)
finder(VS2013).vbproj     Visual Studio 2013 (32-bit)
finder(VS2013)x64.vbproj  Visual Studio 2013 (64-bit)
```

4.5 LISTER.VB

The LISTER program list all serial ports.

The LISTER project files are

```
finder.vbproj             Visual Studio (all 32-bit versions)
finder(VS2008).vbproj     Visual Studio 2008 (32-bit)
finder(VS2008)x64.vbproj  Visual Studio 2008 (64-bit)
finder(VS2010).vbproj     Visual Studio 2010 (32-bit)
finder(VS2010)x64.vbproj  Visual Studio 2010 (64-bit)
finder(VS2012).vbproj     Visual Studio 2012 (32-bit)
finder(VS2012)x64.vbproj  Visual Studio 2012 (64-bit)
finder(VS2013).vbproj     Visual Studio 2013 (32-bit)
finder(VS2013)x64.vbproj  Visual Studio 2013 (64-bit)
```

4.6 XMS.VB & XMR.VB

XMS sends a file to the receiver (such as XMR) using the XMODEM protocol.

The project files are

```
xms.vbproj              Visual Studio (all 32-bit versions)
xms(VS2008).vbproj      Visual Studio 2008 (32-bit)
xms(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
xms(VS2010).vbproj      Visual Studio 2010 (32-bit)
xms(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
xms(VS2012).vbproj      Visual Studio 2012 (32-bit)
xms(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
xms(VS2013).vbproj      Visual Studio 2013 (32-bit)
xms(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)

xmr.vbproj              Visual Studio (all 32-bit versions)
xmr(VS2008).vbproj      Visual Studio 2008 (32-bit)
xmr(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
xmr(VS2010).vbproj      Visual Studio 2010 (32-bit)
xmr(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
xmr(VS2012).vbproj      Visual Studio 2012 (32-bit)
xmr(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
xmr(VS2013).vbproj      Visual Studio 2013 (32-bit)
xmr(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

4.7 YMS.VB & YMR.VB

YMS sends a file to the receiver (such as YMR) using the YMODEM protocol.

The project files are

```
yms.vbproj              Visual Studio (all 32-bit versions)
yms(VS2008).vbproj      Visual Studio 2008 (32-bit)
yms(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
yms(VS2010).vbproj      Visual Studio 2010 (32-bit)
yms(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
yms(VS2012).vbproj      Visual Studio 2012 (32-bit)
yms(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
yms(VS2013).vbproj      Visual Studio 2013 (32-bit)
yms(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)

ymr.vbproj              Visual Studio (all 32-bit versions)
ymr(VS2008).vbproj      Visual Studio 2008 (32-bit)
ymr(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
ymr(VS2010).vbproj      Visual Studio 2010 (32-bit)
ymr(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
ymr(VS2012).vbproj      Visual Studio 2012 (32-bit)
ymr(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
ymr(VS2013).vbproj      Visual Studio 2013 (32-bit)
ymr(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

4.8 DEVICE.VB

The DEVICE example program is designed to send a text string to a serial device. A carriage return is appended to the end of the string.

The DEVICE program can be used to send commands to serial devices which use ASCII commands, such as bar code readers, XY-plotters, etc.

The project files are

```
device.vbproj              Visual Studio (all 32-bit versions)
device(VS2008).vbproj      Visual Studio 2008 (32-bit)
device(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
device(VS2010).vbproj      Visual Studio 2010 (32-bit)
device(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
device(VS2012).vbproj      Visual Studio 2012 (32-bit)
device(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
device(VS2013).vbproj      Visual Studio 2013 (32-bit)
device(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

4.9 ProXR.VB

The ProXR example program demonstrates how to control the ProXR relays boards manufactured by ControlAnything.com

The project files are

```
ProXR.vbproj              Visual Studio (all 32-bit versions)
ProXR(VS2008).vbproj      Visual Studio 2008 (32-bit)
ProXR(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
ProXR(VS2010).vbproj      Visual Studio 2010 (32-bit)
ProXR(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
ProXR(VS2012).vbproj      Visual Studio 2012 (32-bit)
ProXR(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
ProXR(VS2013).vbproj      Visual Studio 2013 (32-bit)
ProXR(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

4.10 ReadGPS.VB

The ReadGPS console mode program read lines from a device that is outputting NMEA 183 GPS sentences, although this program will read complete lines from any serial device that outputs such lines.

The project files are

```
ReadGPS.vbproj              Visual Studio (all 32-bit versions)
ReadGPS(VS2008).vbproj      Visual Studio 2008 (32-bit)
ReadGPS(VS2008)x64.vbproj   Visual Studio 2008 (64-bit)
ReadGPS(VS2010).vbproj      Visual Studio 2010 (32-bit)
ReadGPS(VS2010)x64.vbproj   Visual Studio 2010 (64-bit)
ReadGPS(VS2012).vbproj      Visual Studio 2012 (32-bit)
ReadGPS(VS2012)x64.vbproj   Visual Studio 2012 (64-bit)
ReadGPS(VS2013).vbproj      Visual Studio 2013 (32-bit)
ReadGPS(VS2013)x64.vbproj   Visual Studio 2013 (64-bit)
```

5 Visual Basic 4/5/6 Example Programs

All (32-bit) Visual Basic project files end with the extension ".vbp".

5.1 WSCVER32.FRM

The WSCVER example program displays the WSC library version and build number and verifies that WSC32.DLL can be found and loaded by Windows at runtime.

The project file is

```
wscver32.vbp
```

5.2 EASY32.FRM

The EASY example program is a simple terminal program. Everything typed at the keyboard is sent out the serial port, and everything incoming from the serial port is displayed on the screen.

The easiest way to test EASY is to connect to a modem. Typing 'AT' should result in an 'OK' being displayed.

A null-modem cable can also be used to connect two computers together with their serial ports. Run EASY on both machines. Whatever is typed on one machine will be displayed on the other.

The project file is

```
easy32.vbp
```

5.3 SELFTEST (SELF32.FRM)

SELFTEST performs a serial port I/O functionality test. Either a pair of ports on the same computer (using a null modem cable) or a single port (using a loopback adapter) can be tested.

Refer to LOOPBACK.TXT in the **DOCS** directory for an explanation of how to make a loopback adapter (without tools!).

The project file is

```
self32.vbp
```

5.4 MODEM32.FRM

MODEM is similar to EASY, but with enhanced capability. It can set flow control (hardware, software, or none), DTR line (set or clear), RTS line (set or clear), display the transmit and receive queue sizes, detect a break signal, detect changes in DSR and CTS, as well as check for various line errors (parity error, framing error, data overrun, receive queue overflow, and transmit buffer full).

The project file is

```
modem32.vbp
```

5.5 TERM32.FRM

TERM is a simple terminal emulator suitable for calling up a BBS and downloading or uploading files using XMODEM or YMODEM. The TERM program uses MIO.DLL for modem control commands, ASD.DLL for the ASCII protocol, and the XYM.DLL for XMODEM & YMODEM protocol.

Selecting 'Dial' from the menu bar will result in a pop-up dialog requesting the phone number to dial. Once entered, the number is dialed, and the program will wait for up to 60 seconds for the 'CONNECT' string from the modem. This wait can be terminated at any time by choosing 'BREAK' on the menu bar.

Once logged on, files can be uploaded or downloaded by selecting 'Send' or 'Receive' from the menu bar. To abort a file transfer, choose 'BREAK' from the menu bar then type a series of Ctrl-X (^X) characters from the keyboard. Refer to TERM.TXT for more information.

The project file is

```
term32.vbp
```

5.6 FINDER32.FRM

The FINDER program searches for a connected modem. Your modem must be connected to one of COM1, COM2,…,COM256 and must be turned on.

The project file is

```
finder32.vbp
```

5.7 LISTER32.FRM

The LISTER program lists all serial ports.

The project file is

```
lister32.vbp
```

5.8 ATOK32.FRM

The ATOK program demonstrates the use of the Visual Basic class "wscClass.cls". The program transmits an "AT" to the serial port, and receives the response. If the serial port is connected to a modem, sending an "AT" should result in "OK" being received and displayed.

The ATOK program uses classes, which were added to Visual Basic beginning with version 5

The project file is

```
atok32.vbp
```

5.9 DEVICE32.FRM

The DEVICE example program is designed to send a text string to a serial device. A carriage return is appended to the end of the string.

The DEVICE program can be used to send commands to serial devices which use ASCII commands, such as bar code readers, XY-plotters, etc.

The project file is

```
device32.vbp
```

5.10 ProXR32.FRM

The ProXR example program demonstrates how to control the ProXR relays boards manufactured by ControlAnything.com

The project file is

```
ProXR32.vbp
```

5.11 MESSAGE32.FRM

The MESSAGE program is similar to EASY, except that rather than using a timer as in EASY, it uses the SioMessage function to request that WSC send a "Left Button Down" Windows message whenever any new serial data is available.

The project file is

```
message32.vbp
```

5.12 XMS32.FRM & XMR32.FRM

The XMS (XMODEM Send) and XMR (XMODEM Receive) programs are standalone programs which implement the XMODEM protocol.

The project files are

```
xms32.vbp     (XMODEM/Send)
xmr32.vbp     (XMODEM/Receive)
```

5.13 YMS32.FRM & YMR32.FRM

The YMS (YMODEM Send) and YMR (YMODEM Receive) programs are standalone programs that implement the YMODEM protocol.

The project files are

```
yms32.vbp     (YMODEM/Send)
ymr32.vbp     (YMODEM/Receive)
```

5.14 PUTTYPE32.FRM

The PUTTYPE example program demonstrates how to send user defined data across a serial port. Edit it as necessary before compiling.

The project file is

```
PutType32.vbp
```

6 Revision History

NOTE: Version 2.0 was the first Visual Basic version.

Version 2.0: February 17, 1997.

- Includes Win16 and Win32 libraries.
- Added XMODEM & YMODEM DLL (XYDRV.DLL).
- Added TERM example program.

Version 2.1: June 9, 1997.

- Screen display uses MEMO class.
- WIN32 version can display error text from Win32 Windows.
- Added FIND example program.
- Added SioRead function.
- SioInfo can return seconds to expiration [SHAREWARE].

Version 2.2: November 1, 1997.

- XYDRV code fixes bugs.
- Added xyGetFileName function to XYDRV.
- Supports up to 16 ports.
- WSC4VB runs under Windows NT.
- Added ASCII protocol (ASDRV).

Version 2.3: August 14, 1998

- Improvements to XYDRIVER (XMODEM and YMODEM).
- SioTimer function added.
- SioBaud and SioParms can be called before SioReset.
- Declaration files for Power Builder added.

Version 2.4: June 1, 1999

- Improvements made to XYDRIVER (XMODEM and YMODEM).
- Added SioEvent function (Win32 only).
- WSC Class added (VB 5.0 & up)
- Added ATOK example program (VB 5.0 & up).

Version 3.0: July 26, 2000

- Increased default to 32 ports.
- Added several new example programs (WSCVER, THREAD, and DEVICE).
- Added WORD and HTML documentation.
- Added SioMessage function.

Version 3.1: May 1, 2001.

- RESETDEV Win API call not called (allows USB/serial converters).
- SioPutc and SioPuts return immediately.
- New example programs (XMS, XMR, YMS, YMR, and PUTTYPE).
- XYM (XMODEM/YMODEM) allows local upload/download directory to be specified.

Version 3.2: July 30, 2002.

- Added support for **VB.Net**
- Default for RESETDEV is "not called". SioDebug(ASC("R")') to enable.
- SioGetc & SioGets zero unused bits (DataBits 5,6,7).
- Corrected problem with SioBaud(-1, BaudRateCode).
- SioDebug returns -1 if no match.
- Added SioDebug(ASC("W")) toggle SioPuts wait for I/O completion.
- Added code to detect active threads & to close thread handles.
- Added USE_THREADS, so can compile version of WSC32.C without threads.
- Comm handle not saved in SioReset unless it is good.
- SioEvent returns mask that caused the event.
- Added SioInfo(ASC("B")) to get build number.

Version 4.0: November 21, 2003.

- Can now order either with or without source code to the DLLs.
- Added SioSetInteger function to set port specific integer parameters.
- Added SioKeyCode function to pass the key code to the DLL.
- Added SioGetReg function to return the registration string.
- Added "Burst Size" parameter for setting the TX burst size.
- Added ability to signal blocked thread which was blocked by SioEvent.

Version 4.1: August 12, 2004

- Fixed problem with SioTxClear.
- Added overlapped I/O (for non-Win95) so can signal threads to exit w/o killing them.
- Increased default burst size to 256.
- SioFlow returns WSC_RANGE if cannot recognize parameter.
- Adjusted XModem/YModem timing for faster transfers.

Version 4.2: February 21, 2006.

- SioFlow returns 1 if OK.
- SioSetInteger(Port, 'S', 1) always forces SioEvent to unblock.
- Event mutex code added to EventThread() to prevent race conditions.
- Message box displays error if SioWinError(Buffer, 0) called.
- Major change in overlapped I/O
- Fixed problem: SioEvent returning wrong code.
- SioRxClear clears byte saved by SioUnGet.
- Number of supported ports increased to a maximum of 256.
- Added SioEventChar() and SioEventWait() functions.

Version 4.3: September 27, 2007.

- Fixed problem with SioTxQue returning wrong values.
- Changed SioParms so it checks the range of passed arguments.
- Port is verified in SioEventChar.
- SioStatus returns -1 if port is not functioning (USB/serial port disconnected).
- Added SioByteToShort and SioShortToByte (WSC32 only) to handle Unicode ASCII.

Version 4.4: January 20, 2009

- Added SioSetTimeouts() function (sets TX and RX time-outs).
- Provide documentation files in Adobe PDF format.

Version 5.0: November 23, 2009

- Added SioHexView() function.
- Supports 64-bits (WSC64.DLL).
- Added several Visual Studio examples.

Version 5.1: August 30, 2011

- Added SioRxWait() function.
- Added support for Visual Studio 2010 VB compiler.
- Added LISTER example program.

Version 5.2: July 4, 2012

- Added function SioQuiet()
- Added function SioWaitFor()
- Added example program ProXR.c
- Win16 support dropped, but available on request when ordering.

Version 5.3: November 11, 2013

- Added SioLRC() that computes the "longitudinal redundancy check" per ISO 1155.
- SioQuiet() and SioWaitFor() verify the passed port number.
- SioWaitFor() verifies that the passed baud rate is > 0.
- SioSetInteger() no longer requires an open port for global (all ports) parameters.
- Modified SioReset() to make it more tolerant opening slow virtual ports.

Version 5.4: August 20, 2015

- Added SioCRC16() that computes 16-bit CCITT CRC (polynomial 1021 hex).
- Added SioCRC32() that computes 32-bit CCITT CRC (polynomial 04C11DB7 hex).
- Added support for Visual Studio 2013 and Visual Studio 2015.

Version 6.0: March 15, 2017

- Added additional error codes: WSC_BUFFER_RANGE,WSC_BUFLEN_RANGE,WSC_BAD_CMD
- Added additional error codes: WSC_BAD_PARITY.WSC_BAD_STOPBIT,WSC_BAD_WORDLEN
- Added SioErrorText() that returns text associated with specified error codes.
- Added SioPortInfo() that returns baud in BPS (bits per sec) and the theoretical port CPS (char per sec).
- Added SioGetsC() that receives an entire line through the stop (end-of-line) character (usually CR).
- Added ReadGPS example program.