Windows Standard

Serial Communications

Library for Xbase++

Programmer's Manual

(WSC4XB)

Version 6.0

March 27, 2017

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

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

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

TABLE OF CONTENTS

1 Introduction

The **Windows Standard Serial Communications Library for Xbase++ (WSC4XB)** is a toolkit that allows software developers to quickly develop serial communication applications in Alaska Xbase++.

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, modems, lab instruments, medical devices, USB serial devices, scales, GPS navigation, etc.

The **Windows Serial Communications Library for Xbase++ (WSC4XB)** component library supports and has been tested with all versions of Alaska Xbase++. WSC4XB includes numerous example programs with source that demonstrate serial port communications functions.

The **WSC SDK** runs under 64-bit and 32-bit Windows (through Windows 10). A Win32 DLL is provided with WSC4XB. A Win64 DLL is available. The **Windows Standard Communications Library SDK** DLLs (WSC64.DLL and WSC32.DLL) can also be used from any development environment (Visual Basic, C++, Delphi, COBOL, Visual FoxPro, dBase, PowerBASIC, etc.) capable of calling the Windows API.

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

1. WSC4XB is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. WSC does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
3. The WIN32 version of WSC is fully thread safe.
4. 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), Visual Basic (WSC4VB), Visual FoxPro (WSC4FP), Visual dBASE (WSC4DB), and PowerBASIC (WSC4PB). All versions of WSC use the same DLL (WSC32.DLL). However, the examples provided for each version are written in the specified computer environment. Development time is shortened in multi-language projects because programmers need only learn one interface.

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** are:

- Comes with a 32-bit DLL. (64-bit DLL is available.)
- Can control any serial device (scale, barcode reader, etc) connected to the serial port.
- Can control multiple ports (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 multi-drop RS422, and RS485 serial.
- Supports hardware and software flow control.
- Supports any baud rate (32-bit version).
- Ability to specify the parity, word size, and number of stop bits.
- Supports binary and text data transfer.
- State driven Xmodem and Ymodem on multiple ports simultaneously.
- Supports character peek (**SioEventChar**).
- Port re-entrant.
- Is fully threadable.
- 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.
- Supports all versions of Alaska Xbase++.
- 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 FoxPro, Delphi, Visual Basic, dBase, COBOL, Access and Excel.
- Can be purchased with or without ANSI C source code to the WSC DLL.
- Purchase a developer license for WSC4XB and use the DLLs with any other development environment (C++, Visual FoxPro, Visual Basic, .NET, etc).
- Updates are **free** for one year (updates to source code are separate).
- Documentation online as well as in printable format.

Also see WSC versions for other supported languages:

- http://www.marshallsoft.com/wsc4c.htm (C/C++,C#,.NET)
- http://www.marshallsoft.com/wsc4d.htm (Delphi)
- http://www.marshallsoft.com/wsc4vb.htm (Visual Basic, VB.Net, VBA languages)
- http://www.marshallsoft.com/wsc4fp.htm (Visual FoxPro)
- http://www.marshallsoft.com/wsc4db.htm (dBase)
- http://www.marshallsoft.com/wsc4pb.htm (PowerBASIC)

A selection of Xbase++ programs with full source code is included. Refer to Section 4 for more details on each of the example programs.

```
[PROGRAM]     [DESCRIPTION]
WSCVER      :  Program that displays the WSC version number.
SIMPLE      :  A simple terminal emulator.
FINDER      :  Finds a modem connected to one of your serial ports.
LISTER      :  Lists all serial ports.
XMS         :  XMODEM send program.
XMR         :  XMODEM receive program.
YMS         :  YMODEM send program.
YMR         :  YMODEM receive program.
DEVICE      :  Program sends string to serial device.
DIALER      :  Dials up host (or BBS).
ATOK        :  Forms based program that sends "AT" to modem.
SELFTEST    :  Performs port functionality testing.
EVENT       :  A terminal emulator that uses SioEventWait.
PROXR       :  Reads relays from ProXR relay device.
ReadGPS     :  Reads NMEA 183 GPS data.
```

1.2 Documentation Set

The complete set of documentation consists of four manuals in Adobe PDF format. This is the first manual (WSC_4XB) in the set.

- WSC_4XB Programmer's Manual (WSC_4XB.PDF)
- WSC User's Manual (WSC_USR.PDF)
- WSC Reference Manual (WSC_REF.PDF)
- SERIAL User's Manual (SERIAL.PDF)

The WSC_4XB Programmer's Manual is the language specific (Xbase++) manual. All language dependent programming issues are discussed in this manual. Information needed to compile programs in an Xbase++ 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 Manual (SERIAL) contains background information on serial port hardware.

Documentation is also provided online at
http://www.marshallsoft.com/wsc4xb.htm

1.3 Example Program

The following example demonstrates the use of some of the library functions:

```
#INCLUDE "DLL.CH"
#INCLUDE "WSC32.CH"

Procedure Main()
LOCAL Version
LOCAL A, B, C

? "WSCVER 27 March 2017"
?
* pass the key code
if xSioKeyCode(WSC_KEY_CODE) < 0
  ?"ERROR: Bad Key Code!"
  return
endif
Version = xSioInfo(ASC("V"))
* Compute WSC version
A = int(Version / 256)
Version = Version - (256 * A)
B = int(Version / 16)
C = Version - (16 * B)
? "WSC Version: " + LTRIM(Str(A)) + "." + LTRIM(Str(B)) + "." + LTRIM(Str(C))
return
```

Refer to the WSC Reference Manual (WSC_REF) for individual function details.

1.4 Installation

(1) Before installation of WSC4XB, an Xbase++ compiler should already be installed on your system and tested.

(2) Unzip WSC4XB60.ZIP (evaluation version) or WSCxxxxx.ZIP (registered version where xxxxx is your Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE which will install all WSC4XB files, including copying WSC32.DLL to your Windows directory.

Note that no DLL registration is required.

1.5 Uninstalling

Uninstalling WSC4XB is very easy.

First, run UNINSTAL.BAT, which will delete WSC32.DLL from your Windows directory, normally C:\WINDOWS. Next, delete the WSC project directory created when WSC4XB was installed.

1.6 Pricing

A developer license for WSC4XB can be registered for $115 (or $195 with ANSI C source code to the library DLL). 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 version or order directly on our web site at

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

1.7 Updates

When a developer license is purchased for WSC4XB, the developer will received a new registered DLL plus a license file (WSCxxxx.LIC). The license file is needed to download updates to the registered DLL 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 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 never expires.

2 Library Overview

The **Windows Standard Serial Communications Library (WSC)** has been tested on multiple computers running Windows XP through Windows 10. The **WSC4XB** library works with all versions of Alaska Xbase++.

The SETUP installation program will copy the DLLs to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the WSC4XB files are copied to the directory specified (default \WSC4XB). 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 WSC4XB serial communication library component includes a Win32 dynamic link library (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 file can be found in the DLL sub-directory when SETUP is run:

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

2.2 Keycode

WSC32.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.CH. The keycode for the evaluation version is 0. The developer will receive a new keycode and new DLL after purchasing a license. The KEYCODE is passed to **SioKeyCode**.

If an error message (value -108) is received 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 versions of WSC32.DLL from the Windows search path or delete them.

2.3 Limitations on COM Ports

The 32-bit version of WSC4XB (WSC32.DLL) can use any port from COM1 to COM256, provided that the port is known to Windows and there is physical hardware present.

2.4 INCLUDE Files

All example programs include the file `WSC32.CH`. The file `WSC32.CH` contains all the necessary constants and function declarations for WSC4XB. The Xbase++ include file DLL.CH is also required. For example,

```
#INCLUDE "DLL.CH"
#INCLUDE "WSC32.CH"
```

2.5 Dynamic Strings

A string in the C language (in which WSC and Windows are written) consists of a pointer to the first byte of a character buffer in which a zero byte ends the string characters.

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 user defined dllGetMessage function , which copies a text message into it. Note that `SPACE(80)` is called immediately before **`xSioWinError`**.

```
Procedure SayError(ErrCode)
   LOCAL Code
   LOCAL Buffer
   if ErrCode < 0
     ? "ERROR " + STR(ErrCode)
     Buffer = SPACE(128)
     Code = xSioWinError(@Buffer, 128)
     if Code > 0
       ? Left(Buffer,Code)
     endif
   endif
return
```

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.

2.6 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 receive incoming serial data. **SioGetc** and **SioGets** can be called directly. Note that if there is no input data available, SioGetc returns WSC_NO_DATA while **SioGets** returns zero. Also see Section 2.9 “SioEvent Logic”.

2.7 Error Display

The error message text associated with WSC error codes can be displayed by calling **SayError** (refer to Errors.prg). Each sample program contains examples of error processing.

2.8 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 Event.prg example program.

2.9 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 does not access the USB directly but will work with most USB-to serial port converters and with Bluetooth serial.

More information about Virtual serial ports can be found in Section 2.12 of the WSC User’s Manual (WSC_USR). (http://www.marshallsoft.com/wsc_usr.pdf).

2.10 WSC Declaration File

All WSC functions are declared in the WSC declaration file WSC32.CH. This file can be copied to the Xbase++ INCLUDE directory (where Xbase++ can find it) if so desired.

Note that each function is declared with the prefix character of 'X'

2.11 Adding WSC to an Xbase++ Program

Add the two INCLUDE statements

```
#INCLUDE "WSC32.CH"
#INCLUDE "KEYCODE.CH"
```

after

```
#INCLUDE "DLL.CH"
```

Be sure that **SioKeyCode** is the first WSC function called. Note that the key code constant passed to **SioKeyCode** is found in the file KEYCODE.CH. This value will be 0 for the evaluation version and an 8 to 10 digit value for the purchased version.

Compile and link with WSC32.LIB. For example, to compile and link the (console mode) example program SIMPLE.PRG:

```
xpp simple.prg
alink /subsystem:console simple.obj
```

3 Compiler Issues

The **WSC4XB** library works with all versions of Alaska Xbase++.

3.1 Xbase++ Compiler

If you don't have the Alaska Software Xbase++ compiler, you can find it on the web at

http://www.alaska-software.com

3.2 Compiling and Linking Programs

Details about each of the example programs are provided in Section 4.0 "Example Programs".

To compile and link console mode programs such as SIMPLE.PRG, use:

```
xpp simple.prg
alink /subsystem:console simple.obj
```

To compile and link windows GUI programs such as ATOK.PRG, use:

```
xpp atok.prg
alink /subsystem:windows atok.obj
```

Some programs (such as XMR.PRG, XMS.PRG, YMR.PRG, and YMS.PRG) require the XYM32.DLL module and must be linked with XYM32.LIB. For example,

```
alink /subsystem:console xms.obj
```

Also, some programs (such as FINDER.PRG) require the MIO32 module and must be linked with MIO32.LIB. For example,

```
alink /subsystem:console finder.obj
```

4 Example Programs

Before writing your own programs, compile and run the example programs.

4.1 WSCVER

The first example program is the program WSCVER (WSC Version) that displays the WSC library version number.

```
xpp wscver.prg
alink /subsystem:console wscver.obj
```

4.2 SIMPLE

SIMPLE is a very simple communications program using WSC4XB. Everything that is typed on the keyboard is sent to the serial port, and everything incoming from the serial port is displayed on the screen.

The easiest way to test SIMPLE 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 SIMPLE on both machines. Whatever is typed on one machine will be displayed on the other.

```
xpp simple.prg
alink /subsystem:console simple.obj
```

4.3 XMS and XMR

XMS (XMODEM Send) and XMR (XMODEM Receive) are programs that send and receive files using the XMODEM protocol.

```
xpp xms.prg
alink /subsystem:console xms.obj

xpp xmr.prg
alink /subsystem:console xmr.obj
```

4.4 YMS and YMR

YMS (YMODEM Send) and YMR (YMODEM Receive) are console mode programs that send and receive files using the YMODEM protocol.

```
xpp yms.prg
alink /subsystem:console yms.obj

xpp ymr.prg
alink /subsystem:console ymr.obj
```

4.5 FINDER

The FINDER program is a program that searches for a connected modem. Your modem must be connected to one of COM1 through COM4, and the modem must be turned on. FINDER takes no arguments. Note that FINDER uses the MIO module.

```
xpp finder.prg
alink /subsystem:console finder.obj
```

4.6 LISTER

The LISTER program lists all serial ports.

```
xpp lister.prg
alink /subsystem:console lister.obj
```

4.7 ATOK

The ATOK program was developed using the Forms Designer (XPPFD) and is the only graphical example.

The ATOK program sends "AT" to a connected modem and expects an "OK" back.

```
xpp atok.prg
alink /subsystem:windows atok.obj
```

4.8 DEVICE

The DEVICE program is designed for talking to an arbitrary serial device. Use this program as a guide when communicating with all serial devices other than modems and other computers.

```
xpp device.prg
alink /subsystem:console device.obj
```

4.9 DIALER

The DIALER program dials up a HOST (or BBS) program, connects, and then operates like SIMPLE; in which all serial input is copied to the screen and all keyboard input is sent out the serial port.

```
xpp dialer.prg
alink /subsystem:console dialer.obj
```

4.10 SELFTEST

SELFTEST performs basic serial port I/O functionality testing using a loopback adapter. Refer to LOOPBACK.TXT for an explanation of how to make a loopback adapter (without tools!).

```
xpp selftest.prg
alink /subsystem:console selftest.obj
```

4.11 EVENT

The EVENT example program is similar to the SIMPLE example program, except that it uses SioEventWait to block (efficiently wait) waiting for incoming serial data.

```
xpp event.prg
alink /subsystem:console event.obj wsc32.lib
```

4.12 ProXR

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

```
xpp ProXR.prg
alink /subsystem:console ProXR.obj
```

4.13 ReadGPS

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.

```
xpp ReadGPS.prg
alink /subsystem:console ReadGPS.obj
```

5 Revision History

The WSC DLL (WSC32.DLL) is written in ANSI C. All language versions of WSC (C/C++, Delphi, Xbase++, PowerBASIC, FoxPro, dBase, Xbase++, and COBOL) use the same identical DLLs.

Version 3.0: August 7, 2000

- The initial release of the Xbase++ version of WSC.

Version 3.1: May 31, 2001.

- RESETDEV Win API call not called (allows USB/serial converters).
- SioPutc and SioPuts return immediately (optionally).
- XYM (XMODEM/YMODEM) allows local upload/download directory to be specified.

Version 3.2: August 27, 2002.

- Added SELFTEST example program.
- 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: January 22, 2004.

- 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: April 4, 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: October 8, 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).

Version 4.4: February 3, 2009

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

Version 5.0: December 18, 2009

- Supports 64-bits
- Added SioHexView function
- Added SioSleep function

Version 5.1: September 13, 2011

- Added SioRxWait function
- Added LISTER example program.

Version 5.2: July 17, 2012

- Added function SioQuiet()
- Added function SioWaitFor()
- Added example program ProXR.prg

Version 5.3: November 20, 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: September 4, 2015

- Added SioCRC16() that computes 16-bit CCITT CRC (polynomial 1021 hex).
- Added SioCRC32() that computes 32-bit CCITT CRC (polynomial 04C11DB7 hex).

Version 6.0: March 27, 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.