Windows Standard

Serial Communications

Library for Delphi

Programmer's Manual

(WSC_4D)

Version 7.0

October 3, 2019

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 Page 3
  1.1 Features Page 4
  1.2 Documentation Set Page 6
  1.3 Example Program Page 6
  1.4 Installation Page 7
  1.5 Uninstalling Page 7
  1.6 Pricing Page 7
  1.8 Updates Page 7
2 Library Overview Page 8
  2.1 Dynamic Link Libraries Page 8
  2.2 Keycode Page 8
  2.3 Limitations on COM Ports Page 9
  2.4 Using Threads Page 9
  2.5 Waiting for New Serial Data Page 9
  2.6 Using Messages Page 9
  2.7 Error Display Page 10
  2.8 SioEvent Logic Page 10
  2.9 Virtual Serial Ports Page 10
  2.10 Using the WSC Unit Page 10
  2.11 Adding WSC4D to Your Project Page 11
3 Compiler Issues Page 12
  3.1 Delphi Versions Page 12
      3.1.1 Delphi 1 Page 12
      3.1.2 Delphi 2 Page 12
      3.1.3 Delphi 3 Page 12
      3.1.4 Delphi 4, 5, and 6 Page 12
      3.1.5 Delphi 7 Page 13
      3.1.6 Delphi 2005 – 2010 and XE Page 13
      3.1.7 Delphi XE2 - XE8 Page 13
  3.2 Compiling Programs Page 14
  3.3 Compiling WSC Source Page 14
4 Example Programs Page 15
  4.1 VERSION Page 15
  4.2 EASY Page 15
  4.3 SELFTEST Page 15
  4.4 MODEM Page 16
  4.5 TERM Page 16
  4.6 FINDER Page 16
  4.7 DEVICE Page 17
  4.8 MESSAGE Page 17
  4.9 XMS and XMR Page 17
  4.10 YMS and YMR Page 17
  4.11 RS485 Page 18
  4.12 SIMPLE Page 18
  4.13 LISTER Page 18
  4.14 SCALE Page 18
  4.15 GPS Page 18
5 Revision History Page 19

1 Introduction

The **Windows Standard Serial Communications Library for Delphi (WSC4D)** is a toolkit that allows software developers to quickly develop serial communication applications in Delphi or Delphi for .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, modems, lab instruments, medical devices, USB serial devices, scales, GPS navigation, etc.

The **Windows Serial Communications Programmer's Manual for Delphi** provides information needed to compile and run programs in a Delphi programming environment.

The **Windows Serial Communications Library for Delphi (WSC4D)** component library supports and has been tested with all 32-bit and 64-bit versions of Delphi including:

- Borland Delphi (2.0, 3.0, 4.0, 5.0. 6.0 and 7.0)
- Borland Delphi 8 for .NET
- Borland Delphi 2005 & 2006
- Borland Turbo Delphi
- Codegear Delphi 2007
- Embarcadero Delphi 2009 & 2010
- Embarcadero Delphi XE/XE2/XE3/XE4/XE5/XE6/XE7/XE8, Delphi 10, Seattle/Berlin.

WSC4D includes numerous Delphi example programs with source that demonstrate serial port communications functions.

WSC runs under 32-bit and 64-bit Windows XP through Windows 10. Both Win32 and Win64 DLLs are provided. The **Windows Standard Communications Library SDK** DLLs can also be used from any development environment (Visual Basic, C++, Visual FoxPro, COBOL, Xbase++, dBase, PowerBASIC, etc.) capable of calling the Windows API.

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

1. WSC4D is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. Both WIN32 and WIN64 DLLs are included.
3. WSC 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), Visual Basic (WSC4VB), PowerBASIC (WSC4PB), Visual FoxPro (WSC4FP), Visual dBASE (WSC4DB), and Xbase++ (WSC4XB). All versions of WSC use the same DLLs. However, the examples provided for each version are written in the specified programming environment.

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 Delphi** 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 52 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.
- State driven Xmodem and Ymodem on multiple ports simultaneously.
- Supports character peek (**SioEventChar**).
- Port re-entrant.
- Is fully thread safe.
- 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. Purchase a developer license for WSC4D and use the DLLs with any other development environment (C++, Visual FoxPro, etc).
- Royalty free distribution with your 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.
- Includes boy Win32 and Win64 DLLs.
- 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 Borland/Codegear/Embarcadero Delphi, from Delphi 1 thru Embarcadero Delphi XE8.
- Supports Embarcadero Delphi 10 Seattle.
- Works with Delphi for .NET.
- 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, Visual Basic, VB.NET, Xbase++, dBASE, COBOL, Access and Excel.
- Can be purchased with or without ANSI C source code to the WSC DLLs.
- Updates are **free** for one year (updates to source code are separate).
- Documentation online as well as in printable format.

A selection of Delphi example programs with full source code is included. Refer to section 4.0 for more details on each of the example programs.

```
VERSION :  Displays WSC version number.
EASY    :  A simple RS232 terminal program.
RS485   :  A simple RS485 terminal 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.
MESSAGE :  Same as EASY except uses SioMessage to await incoming data.
XMS/XMR :  XMODEM send/receive example programs.
YMS/YMR :  YMODEM send/receive example programs.
DEVICE  :  Sends text string to serial device.
SIMPLE  :  Delphi 2005/2006/2007/2009/2010 .NET terminal program.
SCALE   :  Sends command to digital scale then reads response.
GPS     :  Reads NMEA 183 GPS sentences.
```

WSC4D contains 52 functions and modem control. All functions return a negative number if an error condition is detected. For more details, consult the WSC Reference Manual (WSC_REF) and the Serial User's Manual (SERIAL).

1.2 Documentation Set

The complete set of documentation consists of four manuals in Adobe PDF format. This is the first manual (WSC_4D) in the set.

- WSC_4D Programmer's Manual (WSC_4D.PDF)
- WSC User's Manual (WSC_USR.PDF)
- WSC Reference Manual (WSC_REF.PDF)
- Serial User's Manual (SERIAL.PDF)

The WSC_4D Programmer's Manual is the language specific manual. All language dependent programming issues are discussed in this manual. Information needed to compile your programs in a Delphi 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's Manual (SERIAL) contains background information on serial port hardware.

The documentation is provided in the \WSC4D\DOCS folder and on our web site at

http://www.marshallsoft.com/wsc4d.htm

1.3 Example Program

The following example program segment transmits an "AT" to a modem connected to the serial port.

```
var
  Code : Integer;
begin
  {pass the key code}
  Code := SioKeyCode(WSC_KEY_CODE) < 0 then
  Code := SioReset(COM1, 1024, 1024);
  If Code < 0 Then
    begin
      WriteLn('Cannot open port');
      exit
    end
  { transmit "AT" }
  Code := SioPutc(COM1, 'A');
  Code := SioPutc(COM1, 'T');
  Code := SioPutc(COM1, Chr(13))
end;
```

Refer to Section 4.0 for complete examples with source.
Refer to the WSC Reference Manual (WSC_REF) for individual function details. Access online at

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

1.4 Installation

(1) Before installation of WSC4D, your Delphi compiler should already be installed on your system and tested. Note that Delphi 2 (or above) is required in order to create Win32 programs.

(2) Unzip WSC4D70.ZIP (evaluation version) or WSCxxxxx.ZIP (registered version; xxxxx is the Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE which will install all WSC4D files, including copying WSC32.DLL and WSC64.DLL to your Windows directory. No DLL registration is needed.

1.5 Uninstalling

Uninstalling WSC4D is very easy. First, delete the WSC4D project directory created when installing WSC4D. Next, delete WSC32.DLL and WSC64.DLL from your Windows directory, which is normally \WINDOWS.

Running the UNINSTAL.BAT batch file will also delete WSC32.DLL, MIO32.DLL, XYM32.DLL, and ASD32.DLL (and the 64 bit versions) as described above.

1.6 Pricing

A developer license for the Windows Standard Serial Communications Library can be purchased for $115 (or $195 with ANSI C source code to the library DLL's). Purchasing details can be found in Section_1.3 "How to Purchase" in the WSC User's Manual (WSC_USR.PDF). See 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, the developer receives a set of registered DLLs plus a license file (WSCxxxxx.LIC). The license file is used 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, your license must be updated if you want to be able to download updates. Your license can be updated for:

- $33 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.
- $77 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 do not expire.

2 Library Overview

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

The WSC4D library supports all versions of Borland (Codegear/Embarcadero) Delphi (Delphi 1 – Delphi 8, Delphi 2005 – 2010, Delphi XE/XE2/XE3/XE4/XE5/XE6/XE7/XE8, Delphi 10, Seattle and Berlin.

The SETUP installation program will copy the DLLs to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the WSC4D files are copied to the directory specified (default \WSC4D). Three sub-directories are created, as follows:

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

Please examine the WSC32.PAS and WSC64.PAS declaration files. Note that COM1 is defined as port zero, not port one. The user must assume the responsibility for passing the correct information when calling WSC4D functions.

2.1 Dynamic Link Libraries

The **Windows Serial Communications (WSC)** library component includes both Win32 and a Win64 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 DelphiX 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".

2.2 Keycode

WSC32.DLL and WSC64.DLL each have a keycode encoded within them. The keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.PAS. The keycode for the evaluation (demo) version is 0. A new keycode and a set of new DLL's are provided after purchasing a developer 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 WSC DLL. After purchasing, it is best to remove the evaluation version of the WSC32 and WSC64 DLLs from the Windows search path or delete them.

2.3 Limitations on COM Ports

WSC can use any real or virtual port from COM1 to COM256, provided that the port is known to Windows.

2.4 Using Threads

WSC4D is thread safe, and can be used from any Windows Win32/Win64 application capable of using threads.

2.5 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:

2.5.1 Polling Method: For Win32 and Win64 programs (including Delphi for .NET).

The most straightforward method is to use a Delphi timer to check the receive queue 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.

2.5.2 Message Method: For Win32 programs only (including Delphi for .NET).

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

2.5.3 Event Method: For Win32/Win64 programs (including Delphi for .NET).

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 Delphi.

2.5.4 EventWait Method: For Win32/Win64 programs (including Delphi for .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.

2.6 Using Messages

WSC is capable (**SioMessage** function) of sending windows messages in response to specified serial events. Like using threads, 32-bit Delphi is required.

Refer to the MESSAGE program (MESS_PRJ) for an example of using messages with WSC4D.

2.7 Error Display

The error message text associated with **WSC** error codes can be displayed by calling procedure DisplayError found in file Display.pas.

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.

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 work with USB ports directly but will work with most USB-to-serial port converters as well as with Bluetooth serial.

More information about Virtual serial ports can be found in the WSC User's Manual, Section 2.12, "Virtual Serial Ports" (http://www.marshallsoft.com/wsc_usr.pdf).

2.10 Using the WSC Unit

The **Windows Standard Serial Communications (WSC)** library is written in ANSI C (like Windows itself). In C, strings are zero terminated. Note the manner in which strings are passed to **WSC** functions.

```
var
   Code : Integer;
   BufferLen : Integer;
   BufferStr : AnsiString;
   BufferPtr : PChar;
begin
   BufferStr := 'AT' + Chr(13);
   BufferLen := Length(BufferStr);
   StrPCopy(BufferPtr, BufferStr);
   {pass buffer to WSC function}
   Code := SioPuts(COM1, BufferPtr, BufferLen);
end;
```

Buffers can also be converted to Delphi strings with StrPas. For example:

```
 Text := StrPas(BufferPtr);
```

2.11 Adding WSC4D to Your Project

Copy WSC32.PAS (if running 32-bit Delphi) or WSC64.PAS (if running 64-bit Delphi) to the same directory (folder) as the application program to which you want to add WSC code. You will find these files in the APPS directory (folder) created when you ran SETUP, usually C:\WSC4D\APPS.

For 32-bit Delphi, add `wsc32` (wsc32uc for Delphi .NET) and `keycode` to your "uses" clause in your source program (*.PAS). For example,

```
uses
  wsc32, keycode, ...
```

You can leave 'keycode' out above if you put your numerical keycode value (found in `keycode.pas`) directly into the call to `SioKeyCode`. Also add `wsc32` to your project file (*.DPR). For example,

```
uses
  wsc32 in 'wsc32.pas', ...

 {pass the key code}
 Code := SioKeyCode(123456789) < 0 then
```

3 Compiler Issues

3.1 Delphi Versions

Applications written with Delphi link with the same identical DLL's as for applications written in all other supported languages, such as C/C++ and Visual Basic.

3.1.1 Delphi 1

Beginning with this version (5.2), WSC4D no longer supports Win16.

3.1.2 Delphi 2

Delphi version 2 and above generates Win32 code. Therefore, applications written using Delphi 2 will access WSC32.DLL. Strings can be up to 2GB rather than 255 bytes as in Delphi 1.

Delphi 2 seems to have a problem with some of the PChar string functions. Although the default is "large strings", some of the string functions (such as StrPCopy) copy only 255 bytes. The MYSTRING.PAS unit contains a replacement unit to use instead of StrPCopy.

3.1.3 Delphi 3

Delphi 3 also has some problems with PChar string functions such as StrPCopy. See the above section.

3.1.4 Delphi 4, 5, and 6.

There are no known Delphi problems impacting our example programs in Delphi version 4 and above. Applications written using Delphi 4 through Delphi 6 will access WSC32.DLL.

3.1.5 Delphi 7.

Beginning in Delphi 7, the filename of a unit must match the unit name. Applications written using Delphi 7 will access WSC32.DLL.

3.1.6 Delphi 2005, 2006, 2007, 2009, 2010, XE

Delphi 2005, 2006, 2007, 2009 and 2010 are Borland's (Codegear/ Embarcadero) latest Delphi products with support for both Win32 and the Microsoft .NET Framework. Application programs written using Delphi 2005, 2006, 2007, 2009, 2010, and XE will access WSC32.DLL.

When loading Win32 Delphi projects with Delphi 2005-2010 or XE, a window entitled "Project Upgrade" will be displayed:

This project must be upgraded before it can be opened. Please select which project type you wish to target:

```
( ) Delphi for .NET
( ) Delphi for Win32
```

Choose "Delphi for Win32" for all projects except "simple_project.bdsproj", which is a Delphi for .NET project.

Note: Beginning with Delphi 2009, the definition of "Char" was changed from a 8-bit character (AnsiChar) to a 16-bit character (WideChar). The example code was modified so that it would compile with all 32-bit versions of Delphi through Delphi XE. Refer to the Win32 example programs in the APPS directory.

3.1.7 Embarcadero Delphi XE2 through Delphi 10 Seattle/Berlin

Embarcadero Delphi XE2, XE3, XE4, XE5, XE6, XE7, XE8 and 10 Seattle can create both 32-bit and 64-bit executables. Note that although a 32-bit executable can run on both 32-bit and 64-bit Windows machines, a 64-bit executable can run only on a 64-bit Windows machine. **Note: The file X64.ZIP contains the example projects modified for 64-bit Delphi.**

Step 1: Edit Files

1. Open file *_PGM.PAS using any text editor.
2. Replace WSC32 with WSC64 in the "uses" clause.
3. Open file *_PRJ.DPR using any text editor.
4. Replace WSC32 with WSC64.

Step 2: Delete Win32 files

1. Delete all *.DCU files that may have been created under Win32 Delphi.
2. Delete all *.DPROJ files that may have been created under Win32 Delphi.

Step 3: Change Win32 to Win64 in Delphi XE2, XE3 or XE4 Project Manager

Start Delphi XE2, load the project *_PRJ.DPR, then open the Delphi Project Manager.

1. Right click on "Target Platforms (Win32)" and click on "Add Platform...".
2. When the "Select Platform" window is displayed, click "64-bit Windows".
3. Click [+] box to left of "Target Platforms (Win32)"
4. Right click on "32-bit Windows", then click "Remove Platform"

3.2 Compiling Programs

The example programs are compiled from the Delphi development environment using the provided Delphi project files (*.DPR).

The example programs will compile and run with any version of Delphi. They have each been tested using Delphi 1 through Delphi 7 and Delphi 2005 through Delphi XE/XE2/XE3/XE4/XE5/XE6/XE7/XE8/10 Seattle/Berlin.

Respond with "OK" to the message "Cannot find resource file..." and it will be properly rebuilt.

See Section 4 "Example Programs" for more details on each of the example programs.

WSC4D may also be used with "Borland Pascal for Windows".

3.3 Compiling WSC Source

WSC is written in standard ANSI C (WSC32.C and WSC64.C), and has been compiled using Microsoft Visual C/C++. The Win32 version is compiled with the STDCALL and DECLSPEC compiler keywords. Source code for the WSC library can be purchased at the same time as a WSC developer license is purchased.

The 32-bit version of WSC may also be compiled using Borland C/C++ or Watcom C/C++ compilers. If you recompile WSC32.C using Borland or Watcom compilers, the resulting DLL can only be used by applications compiled with the same compiler, unless the STDCALL and DECLSPEC keywords are specified.

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 Example Programs

The example programs are designed to demonstrate the various capabilities of WSC4D. The best way to become familiar with WSC4D is to study and run the example programs.

4.1 VERSION

The VER ("WSC Version") example program displays the WSC version number. This is the first program to compile and build since it verifies that WSC32.DLL (or WSC64.DLL) is installed properly.

The project files are:

```
VER_PRJ.DPR : Project file.
VER_PGM.PAS : Program file.
VER_PGM.DFM : Delphi Form file.
```

4.2 EASY

EASY is a very simple communications program using WSC4D. 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 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 files are:

```
EASY_PRJ.DPR : Project file.
EASY_PGM.PAS : Program file.
EASY_PGM.DFM : Delphi Form file.
```

4.3 SELFTEST

The SELF ("selftest") 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 for an explanation of how to make a loopback adapter (without tools!).

The project files are:

```
SELF_PRJ.DPR : Project file.
SELF_PGM.PAS : Program file.
SELF_PGM.DFM : Delphi Form file.
```

4.4 MODEM

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 and detect changes in DSR and CTS. It can also check for various line errors (parity error, framing error, data overrun, receive queue overflow, and transmit buffer full).

The project files are:

```
MODM_PRJ.DPR : Project file.
MODM_PGM.PAS : Program file.
MODM_PGM.DFM : Delphi Form file.
```

4.5 TERM

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 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.

The project files are:

```
TERM_PRJ.DPR : Project file.
TERM_PGM.PAS : Program file.
TERM_PGM.DFM : Delphi Form file.
```

4.6 FINDER

The FINDER program searches for a connected modem. Your modem must be connected to one of COM1, COM2, COM3, or COM4, and must be turned on.

The project files are:

```
FIND_PRJ.DPR : Project file.
FIND_PGM.PAS : Program file.
FIND_PGM.DFM : Delphi Form file.
```

4.7 DEVICE

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:

```
DVCE_PRJ.DPR : Project file.
DVCE_PGM.PAS : Program file.
DVCE_PGM.DFM : Delphi Form file.
```

4.8 MESSAGE

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 MESSAGE example program requires Delphi 2 or above. The project files are:

```
MESS_PRJ.DPR : Project file.
MESS_PGM.PAS : Program file.
MESS_PGM.DFM : Delphi Form file.
```

4.9 XMS and XMR

XMS (XMODEM Send) and XMR (XMODEM Receive) are programs that send and receive files using the XMODEM protocol. XMS and XMR must be edited before compiling.

See XMODEM.TXT for more information on the XMODEM protocol. The project files are:

```
XMS_PRJ.DPR, XMR_PRJ.DPR : Project files.
XMS_PGM.PAS, XMR_PGM.PAS : Program files.
XMS_PGM.DFM, XMS_PGM.DFM : Delphi Form files.
```

4.10 YMS and YMR

YMS (YMODEM Send) and YMR (YMODEM Receive) are programs that send and receive files using the YMODEM protocol. YMS and YMR must be edited before compiling.

See YMODEM.TXT for more information on the YMODEM protocol. The project files are:

```
YMS_PRJ.DPR, YMR_PRJ.DPR : Project files.
YMS_PGM.PAS, YMR_PGM.PAS : Program files.
YMS_PGM.DFM, YMS_PGM.DFM : Delphi Form files.
```

4.11 RS485

The RS485 example console mode program operates like SIMPLE, except that it assumes an RS485 port. RTS is set before transmitting data, and cleared after the last bit of the last byte has been sent.

The project files are:

```
RS485PRJ.DPR : Project file.
RS485PGM.PAS : Program file.
Rs485PGM.DFM : Delphi Form file.
```

4.12 SIMPLE

The SIMPLE example program is the Delphi.NET (2005/ 2006/2007/2009/2010) equivalent of EASY, and can only be loaded using Delphi.NET 2005.

The project files are:

```
SIMPLE_PROJECT.BDSPROJ    : Project file.
SIMPLE_WINFORM.PAS        : Program file.
```

4.13 LISTER

The LISTER program lists all serial ports on your computer.

The project files are:

```
LIST_PRJ.DPR : Project file.
LIST _PGM.PAS : Program file.
LIST _PGM.DFM : Delphi Form file.
```

4.14 SCALE

The SCALE example sends commands to a scale or balance then reads the response.

The project files are:

```
SCALE_PRJ.DPR : Project file.
SCALE_PGM.PAS : Program file.
SCALE_PGM.DFM : Delphi Form file.
```

4.15 GPS

The GPS example 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:

```
GPS_PRJ.DPR : Project file.
GPS_PGM.PAS : Program file.
GPS_PGM.DFM : Delphi Form file.
```

5 Revision History

NOTE: Version 2.0 was the first Delphi version of WSC.

Version 2.0: February 17, 1997.

- Includes Win16 (Delphi 1) and Win32 (Delphi 2) libraries.
- Added XMODEM & YMODEM DLL (XYM.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: October 20, 1997.

- New XYM code fixes bugs.
- Added xyGetFileName function to XYM.
- Supports up to 16 ports.
- WSC4D runs under Windows NT.

Version 2.3: August 19, 1998

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

Version 2.4: June 7, 1999

- Improvements made to XYDRIVER (XMODEM and YMODEM).
- Added SioEvent function (Win32 only).
- Added EVENT example program (Win32 only).

Version 3.0: September 1, 2000

- Increased default to 32 ports.
- Added several new example programs (VER, MESS, XMS,XMR,YMS,YMR)
- Added WORD and HTML documentation.
- Added SioMessage function.

Version 3.1: May 8, 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 16, 2002.

- Default for RESETDEV is "not called". SioDebug('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('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('B') to get build number.

Version 4.0: December 12, 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 that 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: March 3, 2006.

- Added support for Delphi 2005 .NET
- 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 28, 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).

Version 4.4: January 28, 2009.

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

Version 5.0: December 11, 2009

- Supports 64-bits
- Added SioHexView function
- Example programs updated for Delphi 2009-2010

Version 5.1: September 2, 2011

- Added SioRxWait function
- Added LISTER example program
- Example programs updated for Delphi XE

Version 5.2: July 20, 2012

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

Version 5.3: November 19, 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.
- Examples converted to make calls to WSC functions more efficiently.
- Added support for Delphi XE3 and Delphi XE4.

Version 5.4: September 9, 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 Embarcadero Delphi XE5, XE6, XE7 and XE8.
- Added support for Embarcadero Delphi 10 Seattle.

Version 7.0: October 3, 2019

- Fixed: SioErrorText() now returns text length from call to SioWinError()
- Fixed: SioGets() would never timeouts when overlapped I/O was enabled.
- Added: function SioOpen - same as SioReset(Port, 1280, 1280)
- Added: function SioClose - same as SioDone.
- Added: function SioGetsQ - reads port until no incoming data for specified "quiet" time.
- Added Scale program.

Version 6.0: April 6, 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 GPS example program.