ZylSerialPort 1.78

ZylSerialPort is a Delphi / C++Builder thread based asynchronous serial port component.
Use ZylSerialPort component to easily communicate with external devices on serial port connection, such as modems, bar code readers and so on.

The demo version is fully functional in Delphi and C++Builder IDE, but it displays a nag dialog (the licensed version will, of course, not have a nag dialog and will not be limited to the IDE). The package includes demo programs for Delphi and C++Builder and a help file with the description of the component.

Supported Operating Systems: Windows 2000/XP/Server2003/Vista/Server2008/7/8/Server2012/10
Available for: Delphi 11.0 Alexandria (Win32 & Win64), Delphi 10.4 Sydney (Win32 & Win64), Delphi 10.3 Rio (Win32 & Win64), Delphi 10.2 (Win32 & Win64), Delphi 10.1 (Win32 & Win64), Delphi 10 (Win32 & Win64), Delphi XE8 (Win32 & Win64), Delphi XE7 (Win32 & Win64), Delphi XE6 (Win32 & Win64), Delphi XE5 (Win32 & Win64), Delphi XE4 (Win32 & Win64), Delphi XE3 (Win32 & Win64), Delphi XE2 (Win32 & Win64), Delphi XE, Delphi 2010, Delphi 2009, Delphi 2007, Delphi 2006, Delphi 7, Delphi 6, Delphi 5, C++Builder 11.0 Alexandria (Win32 & Win64), C++Builder 10.4 Sydney (Win32 & Win64), C++Builder 10.3 Rio (Win32 & Win64), C++Builder 10.2 (Win32 & Win64), C++Builder 10.1 (Win32 & Win64), C++Builder 10 (Win32 & Win64), C++Builder XE8 (Win32 & Win64), C++Builder XE7, C++Builder XE6, C++Builder XE5, C++Builder XE4, C++Builder XE3, C++Builder XE2, C++Builder XE, C++Builder 2010, C++Builder 2009, C++Builder 2007, C++Builder 2006, C++Builder 6, Turbo Delphi, Turbo C++
Remarks:
- The Delphi 2006 version is fully compatible with Turbo Delphi
- The C++Builder 2006 version is fully compatible with Turbo C++

Insatallation:
If you have a previous version of the component installed, you must remove it completely before installing this version. To remove a previous installation, proceed as follows:
-Start the IDE, open the packages page by selecting Component - Install Packages
-Select ZylSerialPortPack package in the list and click the Remove button
-Open Tools - Environment Options - Library and remove the library path pointing to ZylSerialPort folder
-Close the IDE
-Browse to the folder where your bpl and dcp files are located (default is $(DELPHI)\Projects\Bpl for Delphi, $(BCB)\Projects\Bpl for C++ Builder). -Delete all of the files related to ZylSerialPort
-Delete or rename the top folder where ZylSerialPort is installed
-Start regedit (click Start - Run, type "regedit.exe" and hit Enter). Open the key

HKEY_CURRENT_USER\Software\Borland\<compiler>\<version>\Palette and delete all name/value items in the list related to ZylSerialPort. (<compiler> is either "Delphi" or "C++Builder", <version> is the IDE version you have installed)

-Unzip the zip file and open the ZylSerialPortPack.dpk file in Delphi (ZylSerialPortPack.bpk file in C++Builder), compile and install it
and add to Tools/Environment Options/Library (in older Delphi/C++Builder menu) or Tools/Options/Delphi Options/Library/Library Path (in newer Delphi menu) or Tools/Options/C++ Options/Paths and Directories/Library Path & Include Path (in newer C++Builder menu) the path of the installation (where the ZylSerialPort.dcu or ZylSerialPort.dcuil file is located). The component will be added to the "Zyl Soft" tab of the component palette. After you have the component on your component palette, you can drag and drop it to any form, where you can set its properties by the Object Inspector and you can write event handlers selecting the Events tab of the Object Inspector and double clicking the preferred event.
If you still have problems in C++Builder, running an application, which contains the component, then open the project and in C++Builder menu, Project/Options/Packages and uncheck "Build with runtime packages".

-It is indicated to use this component with "Stop on Delphi exception" option deactivated. You can do this from Delphi / C++Builder menu, "Tools/Debugger Options/Language Exceptions/Stop on Delphi exceptions" in older versions or Tools/Options/Debugger Options/Embarcadero Debuggers/Language Exceptions/Notify on language exceptions in newer versions, otherwise you will have a break at all the handled exceptions.

64-bit platform:
Delphi/C++Builder 64-bit support is only for runtime, so you have to use it in the folllwing way:
Install the 32-bit version of the component as it described above and add to Tools/Options/Delphi Options/Library/Library Path, selected platform: 64-bit Windows the path of the Win64 subfolder of the component.
Before compiling the host application for 64-bit Windows, right click on Target Platforms, Add Platform and add 64-bit Windows (Make the selected platform active). If you compile the application in this way, it will be a native 64-bit application.

Constants:
dcb_Binary = $00000001;
dcb_ParityCheck = $00000002;
dcb_OutxCtsFlow = $00000004;
dcb_OutxDsrFlow = $00000008;
dcb_DtrControlMask = $00000030;
dcb_DtrControlDisable = $00000000;
dcb_DtrControlEnable = $00000010;
dcb_DtrControlHandshake = $00000020;
dcb_DsrSensivity = $00000040;
dcb_TXContinueOnXoff = $00000080;
dcb_OutX = $00000100;
dcb_InX = $00000200;
dcb_ErrorChar = $00000400;
dcb_NullStrip = $00000800;
dcb_RtsControlMask = $00003000;
dcb_RtsControlDisable = $00000000;
dcb_RtsControlEnable = $00001000;
dcb_RtsControlHandshake = $00002000;
dcb_RtsControlToggle = $00003000;
dcb_AbortOnError = $00004000;
dcb_Reserveds = $FFFF8000;

Types:

TCommPort = (
spNone, spCOM1, spCOM2, spCOM3, spCOM4, spCOM5, spCOM6, spCOM7, spCOM8,
spCOM9, spCOM10,
spCOM11, spCOM12, spCOM13, spCOM14, spCOM15, spCOM16, spCOM17, spCOM18,
spCOM19, spCOM20,
spCOM21, spCOM22, spCOM23, spCOM24, spCOM25, spCOM26, spCOM27, spCOM28,
spCOM29, spCOM30,
spCOM31, spCOM32, spCOM33, spCOM34, spCOM35, spCOM36, spCOM37, spCOM38,
spCOM39, spCOM40,
spCOM41, spCOM42, spCOM43, spCOM44, spCOM45, spCOM46, spCOM47, spCOM48,
spCOM49, spCOM50,
spCOM51, spCOM52, spCOM53, spCOM54, spCOM55, spCOM56, spCOM57, spCOM58,
spCOM59, spCOM60,
spCOM61, spCOM62, spCOM63, spCOM64, spCOM65, spCOM66, spCOM67, spCOM68,
spCOM69, spCOM70,
spCOM71, spCOM72, spCOM73, spCOM74, spCOM75, spCOM76, spCOM77, spCOM78,
spCOM79, spCOM80,
spCOM81, spCOM82, spCOM83, spCOM84, spCOM85, spCOM86, spCOM87, spCOM88,
spCOM89, spCOM90,
spCOM91, spCOM92, spCOM93, spCOM94, spCOM95, spCOM96, spCOM97, spCOM98,
spCOM99, spCOM100,
spCOM101, spCOM102, spCOM103, spCOM104, spCOM105, spCOM106, spCOM107,
spCOM108, spCOM109, spCOM110,
spCOM111, spCOM112, spCOM113, spCOM114, spCOM115, spCOM116, spCOM117,
spCOM118, spCOM119, spCOM120,
spCOM121, spCOM122, spCOM123, spCOM124, spCOM125, spCOM126, spCOM127,
spCOM128, spCOM129, spCOM130,
spCOM131, spCOM132, spCOM133, spCOM134, spCOM135, spCOM136, spCOM137,
spCOM138, spCOM139, spCOM140,
spCOM141, spCOM142, spCOM143, spCOM144, spCOM145, spCOM146, spCOM147,
spCOM148, spCOM149, spCOM150,
spCOM151, spCOM152, spCOM153, spCOM154, spCOM155, spCOM156, spCOM157,
spCOM158, spCOM159, spCOM160,
spCOM161, spCOM162, spCOM163, spCOM164, spCOM165, spCOM166, spCOM167,
spCOM168, spCOM169, spCOM170,
spCOM171, spCOM172, spCOM173, spCOM174, spCOM175, spCOM176, spCOM177,
spCOM178, spCOM179, spCOM180,
spCOM181, spCOM182, spCOM183, spCOM184, spCOM185, spCOM186, spCOM187,
spCOM188, spCOM189, spCOM190,
spCOM191, spCOM192, spCOM193, spCOM194, spCOM195, spCOM196, spCOM197,
spCOM198, spCOM199, spCOM200,
spCOM201, spCOM202, spCOM203, spCOM204, spCOM205, spCOM206, spCOM207,
spCOM208, spCOM209, spCOM210,
spCOM211, spCOM212, spCOM213, spCOM214, spCOM215, spCOM216, spCOM217,
spCOM218, spCOM219, spCOM220,
spCOM221, spCOM222, spCOM223, spCOM224, spCOM225, spCOM226, spCOM227,
spCOM228, spCOM229, spCOM230,
spCOM231, spCOM232, spCOM233, spCOM234, spCOM235, spCOM236, spCOM237,
spCOM238, spCOM239, spCOM240,
spCOM241, spCOM242, spCOM243, spCOM244, spCOM245, spCOM246, spCOM247,
spCOM248, spCOM249, spCOM250,
spCOM251, spCOM252, spCOM253, spCOM254, spCustom);

TBaudRate = (br000075, br000110, br000134, br000150, br000300, br000600, br001200,

br001800,
br002400, br004800, br007200, br009600, br014400, br019200, br038400, br057600,
br115200, br128000, br230400, br256000, br460800, br921600, brCustom);

TStopBits = (sb1Bit, sb1_5Bits, sb2Bits);
TDataWidth = (dw5Bits, dw6Bits, dw7Bits, dw8Bits);
TParityBits = (pbNone, pbOdd, pbEven, pbMark, pbSpace);
THwFlowControl = (hfNONE, hfDTRDTS, hfRTSCTS);
TSwFlowControl = (sfNONE, sfXONXOFF);
TLineStatus = (lsCTS, lsDSR, lsRING, lsCD);
TConnectEvent = procedure(Sender: TObject; Port: TCommPort) of object;
TLineStatusSet = set of TLineStatus;TConnectEvent = procedure(Sender: TObject; Port: TCommPort) of object;
TSendReceiveEvent = procedure(Sender: TObject; Buffer: AnsiString) of object;
TLineStatusEvent = procedure(Sender: TObject; LineStatus: TLineStatusSet) of object;

EZylSerialPortException = class(Exception); //custom exception class

Properties:

AutoReceive: Boolean - if this property is true, all incoming data is automatically received in the OnReceive event (default), otherwise you have to use the ReadBuffer method or other Read methods, to read it. The default value is true.
AutoReconnect: Boolean - Set this property to true, if you want to automatically reconnect to the serial port after a OnFault event, when the port is available again. Set AutoReconnect to true, before the port is faulted, otherwise it will have no effect.
The default value is false.
AutoReconnectCheckInterval: Integer - The time interval in milliseconds the serial port is trying to periodically reconnect, after a OnFault event occur, if AutoReconnect is set to true. It must be a positive value. The default value is 4000.
Port: TCommPort - serial port name. If you change the port, you need to call the method Open again.
ConnectionTime: TDateTime - Indicates when the last connection was established.
CustomPortName: AnsiString - custom port name, when Port property is set to spCustom. Now you can open ports with any name.
BaudRate: TBaudRate - baud rate value at which the communication device operates
CustomBaudRate - baud rate value at which the communication device operates, when BaudRate property is set to brCustom
DataWidth: TDataWidth - number of bits in the bytes transmitted and received
StopBits: TStopBits - number of stop bits to be used
Parity: TParityBits - parity scheme to be used
PacketSize: LongWord - the size of the packets in bytes you want to receive automatically in OnReceive event. Use this property, if you want to receive packets of equal size in the OnReceive event. If the value is 0, this property will be ignored. It works only for AutoReceive = true. Default value is 0.
EnableDTROnOpen: Boolean - enable / disable DTR when the port is open
EnableRTSOnOpen: Boolean - enable / disable RTS when the port is open
HwFlowControl: THwFlowControl - hardware flow control
SwFlowControl: TSwFlowControl - software flow control
IsReceiving: Boolean - this property is true when the component is receiving data from the port, otherwise is false.
IsSending: Boolean - this property is true when the component is sending data to the port, otherwise is false.
IsFaulted: Boolean - indicates that the last connection was faulted.
InputBuffer: Cardinal - recommended size of the device's internal input buffer, in bytes.
MaxLineLength: Integer - the maximum length of a line, when ReadStringLine or

ReadStringUpToEndChars methods are used in synchronous mode (AutoReceive = False). If maximum length is reached, the methods return with empty string. If this value is 0, there is no maximum length. The default value is 0.
NeedSynchronization: Boolean - set this property to true for thread safety. If you use the component in ActiveX environment, set this property to false. The default value is false.
NewLine: AnsiString - Gets or sets the value used to interpret the end of a call to the SendStringLine methods.
OutputBuffer: Cardinal - recommended size of the device's internal output buffer, in bytes.
PacketSize: LongWord - The size of the packets in bytes you want to receive automatically in the OnReceive event. Use this property, if you want to receive packets of equal size in the OnReceive event. If the value is 0, this property will be ignored. It works only for AutoReceive = true. Default value is 0.
ReadIntervalTimeout: Longint - Maximum time allowed to elapse between the arrival of two characters on the communications line, in milliseconds. During a ReadFile operation, the time period begins when the first character is received. If the interval between the arrival of any two characters exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used. A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the characters that have already been received, even if no characters have been received.
ReadTotalTimeoutMultiplier: Longint - Multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.
ReadTotalTimeoutConstant: Longint - Constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes.
A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.
WriteTotalTimeoutMultiplier: Longint - Multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.
WriteTotalTimeoutConstant: Longint - Constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written.
A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations.
Priority: TThreadPriority - priority of the reader thread
Delay: Word - Time interval between two receivings in milliseconds (Delay = 1000* (1/FrequencyInHertz))
LogInput: Boolean - if this property is true a logfile will be created with any valid received data.
LogDateTime: Boolean - if this property is true, the datetime will be added to every line of the log file.
LogFile: String - name and path of the log file.
EvtChar: AnsiChar - Value of the character used to signal an event (ASCII: 0 - 255). The default value is #26.
XoffChar: AnsiChar - specifies the value of the XOFF character for both transmission and reception. The default value is #19.
XonChar: AnsiChar - specifies the value of the XON character for both transmission and reception. The default value is #17;
XoffLim: Word - specifies the maximum number of bytes allowed in the input buffer before the XOFF character is sent. The maximum number of bytes allowed is calculated by subtracting this value from the size, in bytes, of the input buffer. The default value is 1.
XonLim: Word - specifies the minimum number of bytes allowed in the input buffer before the XON character is sent. The default value is 0, which means InputBuffer div 4.
ErrorChar: AnsiChar - specifies the value of the character used to replace bytes received with a

parity error. The default value is #0.
EofChar: AnsiChar - specifies the value of the character used to signal the end of data. The default value is #0.
ParityCheck: Boolean - specifies whether parity checking is enabled. If this member is TRUE, parity checking is performed and parity errors are reported. This should not be confused with the Parity member, which controls the type of parity used in communications. The default value is false;
ReplaceParityErrors: Boolean - specifies whether bytes received with parity errors are replaced with the character specified by the ErrorChar member. If this member is true and the ParityCheck member is true, replacement occurs. The default value is false;
SkipIfSetupFailed: Boolean - when this property is true, if setting serial parameters like DCB in the Open method fails, will ignore this.
DiscardNulls: Boolean - specifies whether null bytes are discarded. If this member is true, null bytes are discarded when received. The default value is false;
DsrSensitivity: Boolean - specifies whether the communications driver is sensitive to the state of the DSR signal. If this member is true, the driver ignores any bytes received, unless the DSR modem input line is high. The default value is false;
TXContinueOnXoff: Boolean - specifies whether transmission stops when the input buffer is full and the driver has transmitted the Xoff character. If this member is true, transmission continues after the Xoff character has been sent. If this member is false, transmission does not continue until the input buffer is within XonLim bytes of being empty and the driver has transmitted the Xon character. The default value is false;
CloseWhenLineStatusIsZero: Boolean - when this property is true and GetLineStatus returns an empty set, the port will be closed automatically, if line status was not empty, after you opened the port.
IdleInterval: Cardinal - idle time interval in milliseconds. If the elapsed time from the last receive is higher than IdleInterval, OnIdle event occurs. If IdleInterval is 0, Idle event never occurs. It works only for AutoReceive = true.
IsIdle - true, when the connection is in idle state.

Methods:

function BaudRateToInt(pBaudRate: TBaudRate): Integer - converts TBaudRate type to integer
constructor Create(AOwner: TComponent) - constructor
destructor Destroy() - destructor
function DetectDevice(const sentToken, receivedToken: AnsiString; const startBaudRate, endBaudRate: TBaudRate; var pPort: AnsiString, var pBaudRate: TBaudRate): Boolean - Detects the serial port where the device is connected to and
returns as output parameters the communication port and baud rate. In the detection process senToken will be sent and receivedToken is expected to receive. Only baud rate values between startBaudRate and endBaudRate are checked.
function DetectDevice(const sentToken, receivedToken: AnsiString; var pPort: TCommPort; var pBaudRate: TBaudRate): Boolean - Detects the serial port where the device is connected to and returns as output parameters
the communication port and baud rate. In the detection process senToken will be sent and receivedToken is expected to receive.
procedure Open() - starts the communication
procedure Close() - stops the communication
function IntToBaudRate(Value: Integer): TBaudRate - converts integer to TBaudRate type
function IsConnected: TCommPort - returns the comm port where the component is connected to
function StringToCommPort(Port: AnsiString): TCommPort - converts String to TCommPort
function CommPortToString(Port: TCommPort): AnsiString - converts TCommPort to String
function SendByte(const number: Byte): Cardinal - sends a byte to the serial port and returns the number of sent bytes.
function SendChar(const character: AnsiChar): Cardinal - sends a char to the serial port and returns the number of sent bytes.
function SendData(Data: Pointer; dataSize: DWORD): DWORD - sends a memory block to the

serial port and returns the number of sent bytes.
function SendString(str: AnsiString): DWORD - sends string to the serial port and returns the number of sent bytes.
function SendStringLine(const str: AnsiString): Cardinal - Sends an ANSI string and the NewLine value to the serial port. Returns the number of sent bytes.
function SendCharsLine(const str: AnsiString): Cardinal - Sends an ANSI string and the NewLine value to the serial port. Returns the number of sent bytes.
procedure SendStringOnNewThread(const str: AnsiString) - sends string to the serial port on a new thread.
function SendChars(str: AnsiString): DWORD - sends string to the serial port (null characters #0 as well) and returns the number of sent bytes.
procedure SendCharsOnNewThread(const str: AnsiString) - sends string to the serial port (null characters #0 as well) on a new thread.
function SendWord(const number: Word): Cardinal - sends a word to the serial port and returns the number of sent bytes.
function GetExistingCommPorts: TCommPortSet - returns the existing serial ports of the system
procedure GetExistingCommPortNames(Strings: TStrings) - resturns in the Strings parameter the names of the existing serial ports of the system.
function GetCTS(): Boolean - returns the state of CTS line
function GetDSR(): Boolean - returns the state of DSR line
function GetRING(): Boolean - returns the state of RING line
function GetDCD(): Boolean - returns the state of DCD line
function GetLastReadBuffer(): AnsiString - returns the last received buffer
function GetEntireBuffer(): AnsiString - returns the entire buffer, starting from the last open of the port (only if KeepEntireBuffer property is set to true)
function GetAvailableInputBufferSize(): Cardinal - Use this method to get the size of available unread data in bytes. It's useful when AutoReceive property is false.
function GetAvailableOutputBufferSize: Cardinal - Returns the number of bytes of data in the send buffer (bytes to write).
function ClearInputBuffer(): Boolean - clears input buffer. It returns true if succeeded.
function ClearOutputBuffer(): Boolean - clears output buffer. It returns true if succeeded.
function SetBreak(OnOff: Boolean): Boolean - sets hardware line break. It returns true if succeeded.
function SetDTR(OnOff: Boolean): Boolean - sets state of DTR line. It returns true if succeeded.
function SetRTS(OnOff: Boolean): Boolean - sets state of RTS line. It returns true if succeeded.
function SetXonXoff(OnOff: Boolean): Boolean - sets XOnXoff state. It returns true if succeeded.
function TestDevice(sentToken, receivedToken: String; timeout: Integer = 60000): Boolean - tests if the connection is alive sending sentToken and expecting to receive receivedToken. If the ellapsed time is higher than timeout, then it will return with false (antiblocking protection). If timeout is 0, then there is no antiblocking protection.
function ReadBuffer(): AnsiString - Use this method to read incoming data, if AutoReceive property is false.
function ReadChar(): AnsiString - Synchronously reads one char from the serial port. It works only if AutoReceive property is false. Otherwise returns character empty string.
function ReadChars(noChars: Integer; timeout: Integer = 60000): AnsiString - Reads numberOfChars chars from the serial port into string and discards them. It works only if AutoReceive property is false. If the ellapsed time is higher than timeout, then it will return with empty result (antiblocking protection). If timeout is 0, then there is no antiblocking protection.
function ReadByte(): Byte - Synchronously reads one byte from the serial port. It works only if AutoReceive property is false. Otherwise returns 0.
function ReadStringLine(): AnsiString - Synchronously reads the serial port, till the first NewLine sequence appears. NewLine is not included in the result.
function ReadStringUpToEndChars(endChars: String; timeout: Integer = 60000): AnsiString - Synchronously reads the serial port, till the first endChars sequence appears. endChars is not included in the result. If the ellapsed time is higher than timeout, then it will return with empty result

(antiblocking protection). If timeout is 0, then there is no antiblocking protection.
function ReadStringUpToEndChars(endCharsList: TStrings; timeout: Integer = 60000): AnsiString - Synchronously reads the serial port, till one of the endChars sequence appears (one element of endCharsList). endChars is not included in the result. If the ellapsed time is higher than timeout, then it will return with empty result (antiblocking protection). If timeout is 0, then there is no antiblocking protection.
function ReadStringUpToEndChars(endCharsList: TStrings; timeout: Integer; var partialResult: AnsiString): AnsiString - Synchronously reads the serial port, till one of the endChars sequence appears (one element of endCharsList). endChars is not included in the result. If the ellapsed time is higher than timeout, then it will return with empty result (antiblocking protection). If timeout is 0, then there is no antiblocking protection. partialResult is empty string, if endChars sequence was found, otherwise contains the discarded data.
function ReadStringUpToEndChars(endCharsList: TStrings; timeout: Integer; var partialResult: AnsiString; var timedOut: Boolean; var endCharsFound: AnsiString): AnsiString - Synchronously reads the serial port, till one of the endChars sequence appears (one element of endCharsList). endChars is not included in the result. If the ellapsed time is higher than timeout, then it will return with empty result (antiblocking protection). If timeout is 0, then there is no antiblocking protection. timedOut indicates if the read process was timed out.
partialResult is empty string, if endChars sequence was found, otherwise contains the discarded data.
endCharsFound is the terminator characters sequence, which was reached.
procedure ResetIdleState - resets the idle state of the port.
function ReplaceControlChars(strInput: AnsiString): AnsiString - Replaces control characters with readable text.
function InsertControlChars(strInput: AnsiString): AnsiString - Replaces readable text with control characters.
// #0 = '[NUL]'
// #1 = '[SOH] '
// #2 = '[STX]'
// #3 = '[ETX]'
/// (char)4 = '[EOT]'
/// (char)5 = '[ENQ]'
/// (char)6 = '[ACK]'
/// (char)7 = '[BEL]'
/// (char)8 = '[BS]'
/// (char)9 = '[TAB]'
/// (char)10 = '[LF]'
/// (char)11 = '[VT]'
/// (char)12 = '[FF]'
/// (char)13 = "[CR]"
/// (char)14 = "[SO]"
/// (char)15 = "[SI]"
/// (char)16 = "[DLE]"
/// (char)17 = "[DC1]"
/// (char)18 = "[DC2]"
/// (char)19 = "[DC3]"
/// (char)20 = "[DC4]"
/// (char)21 = "[NAK]"
/// (char)22 = "[SYN]"
/// (char)23 = "[ETB]"
/// (char)24 = "[CAN]"
/// (char)25 = "[EM]"
/// (char)26 = "[SUB]"
/// (char)27 = "[ESC]"
/// (char)28 = "[FS]"
/// (char)29 = "[GS]"

/// (char)30 = "[RS]"
/// (char)31 = "[US]"
procedure SetFault - Emulates faulted state, raises the Faulted event and close the port. If AutoReconnect is true, it will try to reconnect.

Events:
OnReceive: TSendReceiveEvent=procedure(Sender: TObject; Buffer: AnsiString) - fires when new data was received. The parameter Buffer contains the received data.
OnSend: TSendReceiveEvent=procedure(Sender: TObject; Buffer: AnsiString) - fires after new string was sent. The parameter Buffer contains the sent string.
OnLineStatusChange: procedure(Sender: TObject; LineStatus: TLineStatusSet) - fires when status lines were changed. The parameter LineStatus contains the current status of the lines.
OnConnect: TConnectEvent=procedure(Sender: TObject; Port: TCommPort) - fires after a new connection was established. The Port parameter contains the serial port where the component is connected to.
OnDisconnect: TConnectEvent=procedure(Sender: TObject; Port: TCommPort) - fires before a disconnection. The Port parameter contains the serial port where the component was connected to.
OnFault: TConnectEvent=procedure(Sender: TObject; Port: TCommPort) - occurs when the serial port communication is faulted. E.g.: when you unplug an USB device which communicates on a virtual serial port.
OnDetect: TDetectEvent=procedure(Sender: TObject; const Port: TCommPort; const BaudRate: TBaudRate; var Cancel: Boolean) - occurs when the serial port detection process is in progress and there are new values of port or baud rate to be checked. If you set the Cancel parameter to true, then the detection will be cancelled.
OnReconnect: TReconnectEvent=procedure(Sender: TObject; const Port: TCommPort; const BaudRate: TBaudRate; var Cancel: Boolean) - occurs when the serial port is trying to reconnect, after fault. AutoReconnect must be true. If you set the Cancel parameter to true, then the reconnection will be cancelled.
OnIdle: TNotifyEvent - If the elapsed time from the last receive is higher than IdleInterval (milliseconds), Idle event occurs. If IdleTinerval is 0, this event will never occur. It works only for AutoReceive = true.
OnResume: TNotifyEvent - Occurs when the receiver is idle and data is received. It works only for AutoReceive = true.

Visit product page

http://www.zylsoft.com
info@zylsoft.com