TMS Async
DEVELOPERS GUIDE

June 2020

Index

Availability

TMS Async is available as VCL component for Delphi and C++Builder.

TMS Async is available for Delphi 7, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney & C++Builder 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney.

TMS Async has been designed for and tested with: Windows XP, Vista, Windows 7, Windows 8, Windows 10.

Online references

TMS software website:
https://www.tmssoftware.com

TMS Async page:
https://www.tmssoftware.com/site/async32.asp

Introduction

TMS Async comprises a set of components for Delphi & C++Builder to implement serial data communications. The core component provides a light-weight, high-performance threaded buffered serial data communication mechanism for Windows serial ports (COM ports). TMS Async uses the Windows serial port communications API. For devices handling serial communications over USB, typically, a virtual COM port driver is provided with the hardware that TMS Async can use.
On top of the core serial communication layer, additional components are provided to deal with X-modem, Y-modem & Z-modem protocols, to emulate an ANSI or TTY terminal, to do data pattern matching to facilitate protocols on serial ports.

Components

TVaComm: core serial communication handling component
TVaBuffer: data buffer component
TVaCapture: pattern monitoring component
TVaWaitMessage: data monitoring component
TVaModem: component handling standard modem AT commands
TVaXModem: X-modem protocol handling component
TVaYModem: Y-modem protocol handling component
TVaZModem: Z-modem protocol handling component
TVaServer: Serial port based server component
TVaServerClient: Client component for TMS Async serial port server
TVaLEDDisplay: component showing visually the state of COM port signals
TVaTerminal: terminal component
TVaTTYEmulation: TTY emulation handling component for TVaTerminal
TVaANSIEmulation: ANSI emulation handling component for TVaTerminal
TVaConfigDialog: Dialog component to allow easy configuration of TVaComm.

Getting Started

Drop a TVaComm component on the form and select the COM port to use via VaComm.PortNum. Select the correct serial port parameters for the port that will be used, i.e. baudrate, databits, stopbits, parity. Call VaComm.Open to get access to the selected COM port. Note that in Windows, only one instance may open a serial port. If the port was already opened by another application, Open will fail and an error VaComm.OnError will be triggered. When the port was successfully opened, data communication may start using the write methods VaComm.WriteBuf(), VaComm.WriteText(), VaComm.WriteChar() or read methods VaComm.ReadBuf(), VaComm.ReadText(), VaComm.ReadChar(). Note that serial port communication is always byte-based. As such, text and char receipt or transmission is handled via ANSI (8bit) strings.

TVaComm properties, methods & events

Properties

Active	Active returns the current state of the COM port. If the port is succesfully opened Active returns True. If the state of the port is closed Active returns false.
AutoOpen	Set AutoOpen = true to open the selected COM port automatically after TVaComm is loaded from a stream, like the DFM stream for example.
Baudrate	Specifies the baud rate at which the communications device operates. Setting BaudRate to brUser will allow you to define a custom baudrate via property UserBaudRate. The predefined baud rates are: br110, br300, br600, br1200, br2400, br4800, br9600, br14400, br19200, br38400, br56000, br57600, br115200, br128000, br256000
Buffers	Access to internal read and write buffers of TVaComm. Buffers exposes following properties: type TVaIntW = 0..MaxInt; *property ReadSize: TVaIntW;* Sets and returns the size of the receive buffer in bytes. ReadSize refers to the total size of the receive buffer. The larger you make the receive buffer, the less memory you have available to your application. However, if your buffer is too small, it runs the risk of overflowing unless handshaking is used. As a general rule, start with a buffer size of 1024 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.

	property WriteSize: TValntW;
	
	Sets and returns the size, in bytes, of the transmit buffer. WriteSize refers to the total size of the transmit buffer. The larger you make the transmit buffer, the less memory you have available to your application. However, if your buffer is too small, you run the risk of overflowing unless you use handshaking. As a general rule, start with a buffer size of 512 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.
	
	property ReadTimeout: TValntW;
	
	Specifies the time-out interval, in milliseconds for reading data from the serial buffer. The function returns if the interval elapses, even if the read data request is not completed. If ReadTimeout is zero, the function tests the read state and returns immediately.
	
	property WriteTimeout: TValntW;
	
	Specifies the time-out interval, in milliseconds for writing data to the serial buffer. The function returns if the interval elapses, even if the write data request is not completed. If WriteTimeout is zero, the function tests it's write state and returns immediately.

CTS	Boolean property that determines whether you can send data by querying the state of the Clear To Send (CTS) line. Typically, the Clear To Send signal is sent from a modem to the attached computer to indicate that transmission can proceed. This property is not available at design time and is read-only at run time. The Clear To Send line is used in RTS/CTS (Request To Send/Clear To Send) hardware handshaking. The CTSHold property gives you a way to manually poll the Clear To Send line if you need to determine its state. For more information on handshaking protocols, see the FlowControl property.
CTSHold	Boolean property that sets whether transmission is waiting for the CTS (clear-to-send) signal to be sent. If this property is TRUE, transmission is waiting.
Databits	Specifies the number of bits in the bytes transmitted and received. The possible settings are: db4, db5, db6, db7, db8
DeviceName	DeviceName describes the communications port name. If a %d parameter is included the port number defined by PortNum is added to the DeviceName when the device is opened. Example: PortNum = 2 DeviceName = COM%d -> COM2
DirectWrite	If DirectWrite is set to True all data written to the serial port is send directly out of the serial port. If DirectWrite is false a separate internal writer thread is used. Increase of performance is achieved with character based transmissions when DirectWrite is false. A write thread will queue all data sent to the serial port. To enable this feature you need to set the DirectWrite property to false. Make sure you define the write buffer size large enough to store the biggest data block your application uses.
DSR	Determines the state of the Data Set Ready (DSR) line. Typically, the Data Set Ready signal is sent by a modem to its attached computer to indicate

	that it is ready to operate. This property is not available at design time and is read-only at run time. This property is useful when writing a Data Set Ready/Data Terminal Ready handshaking routine for a Data Terminal Equipment (DTE) machine.
DSRHold	Specifies whether transmission is waiting for the DSR (data-set-ready) signal to be sent. If this property is TRUE, transmission is waiting.
DTRControl	Determines whether to enable the Data Terminal Ready (DTR) line during communications. Typically, the Data Terminal Ready signal is sent by a computer to its modem to indicate that the computer is ready to accept incoming transmission. When DTREnable is set to True, the Data Terminal Ready line is set to high (on) when the port is opened, and low (off) when the port is closed. When DTREnable is disabled, the Data Terminal Ready always remains low. In most cases, setting the Data Terminal Ready line to low hangs up the telephone. Note: This flag is ignored if FlowControl is set to fcDtrDsr. *dtrDisable* Disables the DTR line when the device is opened and leaves it disabled. *dtrEnable* Enables the DTR line when the device is opened and leaves it on. *dtrHandshake* Enables DTR handshaking. If handshaking is enabled, it is an error for the application to adjust the line by using SetDTRState.
EventChars	This class property exposes XOnChar, XOffChar, ErrorChar, EventChar and EOFChar. *XOnChar* Specifies the value of the XON character for both transmission and reception. *XOffChar* Specifies the value of the XOFF character for both transmission and reception. *ErrorChar* Specifies the value of the character used to replace bytes received with a parity error. *EventChar* Specifies the value of the character used to signal an event. *EOFChar* Specifies the value of the character used to signal the end of data.
EventPriority	Determines the internal reader thread's scheduling priority relative to other threads in the process. Possible settings: tpIdle, tpLowest, tpLower, tpNormal, tpHigher, tpHighest, tpTimeCritical Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

FlowControl	FlowControl refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer so that your program can read it. If there is no receive buffer and your program is expected to read every character directly from the hardware, you will probably lose data because the characters can arrive very quickly. A handshaking protocol insures data is not lost due to a buffer overrun, where data arrives at the port too quickly for the communications device to move the data into the receive buffer. *OutCtsFlow: Boolean;* Specifies whether the CTS (clear-to-send) signal is monitored for output flow control. If this member is TRUE and CTS is turned off, output is suspended until CTS is sent again. *OutDsrFlow: Boolean;* Specifies whether the DSR (data-set-ready) signal is monitored for output flow control. If this member is TRUE and DSR is turned off, output is suspended until DSR is sent again. *ControlDtr: TVaControlDtr;* dtrDisabled: Disables the DTR line when the device is opened and leaves it disabled. dtrEnabled: Enables the DTR line when the device is opened and leaves it on. dtrHandshake: Enables DTR handshaking. If handshaking is enabled, it is an error for the application to adjust the line by using the SetDTR function. *ControlRts: TVaControlRts;* rtsDisabled: Disables the RTS line when the device is opened and leaves it disabled. rtsEnabled: Enables the RTS line when the device is opened and leaves it on. rtsHandshake: Enables RTS handshaking. The driver raises the RTS line when the "type-ahead" (input) buffer is less than one-half full and lowers the RTS line when the buffer is more than three-quarters full. If handshaking is enabled, it is an error for the application to adjust the line by using the SetRTS function. rtsToggle: Specifies that the RTS line will be high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line will be low. *XOnXOffOut: Boolean;* Specifies whether XON/XOFF flow control is used during transmission. If this member is TRUE, transmission stops when the XOffChar character is received and starts again when the XOnChar character is received. *XOnXOffIn: Boolean;* Specifies whether XON/XOFF flow control is used during reception. If this member is TRUE, the XOffChar character is sent when the input buffer comes within XOffLim bytes of being full, and the XOnChar character is sent when the input buffer comes within XOnLim bytes of being empty.

	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.
	
	TxContinueOnXoff: Boolean;
	Specifies whether transmission stops when the input buffer is full and the driver has transmitted the XOffChar character. If this member is TRUE, transmission continues after the input buffer has come within XOffLim bytes of being full and the driver has transmitted the XOffChar character to stop receiving bytes. 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 XOnChar character to resume reception.
	
	Overview of basic settings in various flowcontrol scenarios:
	
	fcNone, no flowcontrol
	OutCtsFlow: false;
	OutDsrFlow: false;
	ControlDtr: dtrDisabled;
	ControlRts: rtsDisabled;
	XOnXOffOut: false;
	XOnXOffIn: false;
	
	fcRtsCts or hardware flowcontrol
	OutCtsFlow: True;
	OutDsrFlow: false;
	ControlDtr: dtrDisabled;
	ControlRts: rtsHandshake;
	XOnXOffOut: false;
	XOnXOffIn: false;
	
	fcDtrDsr
	OutCtsFlow: false;
	OutDsrFlow: True;
	ControlDtr: dtrHandshake;
	ControlRts: rtsDisabled;
	XonXoffOut: false;
	XonXoffIn: false;
	
	fcXOnXOff or software flowcontrol
	OutCtsFlow: false;
	OutDsrFlow: false;
	ControlDtr: dtrDisabled;
	ControlRts: rtsDisabled;
	XonXoffOut: True;
	XonXoffIn: True;

Handle	Public property returning a handle that identifies the communications device. This property is read-only. Use this value when calling any communications routines in the Windows API.
MonitorEvents	MonitorEvents determines which events are handled during serial communications. MonitorEvents is a set of event sources from: ceBreak, ceCts, ceDsr, ceError, ceRing, ceRlsd, ceRxChar, ceTxEmpty, ceRxFlag, ceRx80Full, ceEvent1, ceEvent2, cePErr

	ceBREAK A break was detected on input. *ceCTS* The CTS (clear-to-send) signal changed state. *ceDSR* The DSR (data-set-ready) signal changed state. *ceERROR* A line-status error occurred. Line-status errors are CE_FRAME, CE_OVERRUN, and CE_RXPARITY. *ceRING* A ring indicator was detected. *ceRLSD* The RLSD (receive-line-signal-detect) signal changed state. *ceRXCHAR* A character was received and placed in the input buffer. *ceRXFLAG* The event character was received and placed in the input buffer. The event character is specified in EventChars structure, which is applied to a serial port *ceTXEMPTY* The last character in the output buffer was sent. *ceRX80FULL* Receiver buffer is 80% full *ceEVENT1* Provider specific event 1. This event may vary between different types of hardware. *ceEVENT2* Provider specific event 2. This event may vary between different types of hardware. *cePERR* A serial printer error occured.
Options	coErrorChar Specifies whether bytes received with parity errors are replaced with the character specified by the ErrorChar property. If this property is TRUE and the fParity property is TRUE, replacement occurs. coNullStrip Specifies whether null bytes are discarded. If this property is TRUE, null bytes are discarded when received. coParityCheck Specify whether parity check on incoming data is performed

Parity	Specifies the parity scheme to be used. The parity scheme can be any of these values: paNone, paOdd, paEven, paMark, paSpace
PortNum	Sets and returns the communications port number. You can set value to any number between 1 and 16 at design time (the default is 0, no port). However, the TVaComm control generates an error (Device unavailable) if the port does not exist when you attempt to open it with the "Open" method.
ReadBufFree	Public property returning the number of free characters in the receive buffer. ReadBufFree refers to the number of characters that can be received by the TVaComm without overflowing the input buffer.
ReadBufUsed	Public property returning the number of characters waiting in the receive buffer. This property is not available at design time. ReadBufUsed refers to the number of characters that have been received by the TVaComm and are waiting in the receive buffer for you to take them out.
StopBits	Specifies the number of stop bits to be used. The stopbits can be any of these values: sb1, sb15, sb2
UpdateDCB	When setting UpdateDCB to false TVaComm will use the default setup of the serial port defined by your Windows environment. Otherwise TVaComm will update the settings of the serial port (active Windows session only).
UserBaudrate	UserBaudrate can be used to define a custom baudrate speed. This must be done in two steps. First set the Baudrate property to brUser. Secondly, enter a valid baudrate > 0.
Version	Read-only property returning the version of the component
WriteBufFree	Public property returning the number of free characters in the write buffer. WriteBufFree refers to the number of characters that can be sent to TVaComm without overflowing the output buffer.
WriteBufUsed	Public property returning the number of characters waiting in the output buffer. WriteBufUsed refers to the number of characters that are waiting for transmission.
WritePriority	Determines the internal write thread's scheduling priority relative to other threads in the process. Possible settings: tpIdle, tpLowest, tpLower, tpNormal, tpHigher, tpHighest, tpTimeCritical Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

Events

OnBreak	Event triggered when a break was detected on input.
OnClose	Event triggered when the serial port specified by PortNum was closed succesfully.
OnCts	Event triggered when the CTS (clear-to-send) signal changed state.
OnDsr	Event triggered when the DSR (data-set-ready) signal changed state.
OnError	Event triggered when a line-status error occurred. Line-status errors are CE_FRAME, CE_OVERRUN, and CE_RXPARITY.
OnEvent1	Event triggered when provider specific event 1 occurred. This event may vary for different types of hardware.
OnEvent2	Event triggered when provider specific event 2 occurred. This event may vary for different types of hardware.
OnOpen	Event triggered when the serial port specified by PortNum was opened succesfully.

OnPErrr	Event triggered when a serial printer error occured.
OnRing	Event triggered when a ring indicator was detected.
OnRlsd	Event triggered when the RLSD (receive-line-signal-detect) signal changed state.
OnRx80Full	Event triggered when the receiver buffer is 80% full.
OnRxChar	Event triggered when a character was received and placed in the input buffer.
OnRxFlag	Event triggered when an event character was received and placed in the input buffer. The event character is specified in the EventChars structure, which is applied to a serial port.
OnTxEmpty	Event triggered when the last character in the output buffer was sent.

Methods

Close	Sets and returns the state of the communications port as closed. Not available at design time. Calling "close" closes the port and clears the receive and transmit buffers. TVaComm automatically closes the serial port when your application is terminated. If either the DTREnable or the RTSEnable properties is set to True before the port is opened, the DTR and RTS signals are lowered when the port is closed. Otherwise, the DTR and RTS lines remain in their previous state.
PurgeRead	PurgeRead terminates all outstanding read operations and returns immediately, even if the read operations have not been completed and clears the input buffer (if the device driver has one).
PurgeReadWrite	PurgeReadWrite is a combination of PurgeRead and PurgeWrite. It clears all input and output buffers and terminates all pending transmissions.
PurgeWrite	PurgeWrite terminates all outstanding write operations and returns immediately, even if the write operations have not been completed and clears the output buffer (if the device driver has one).
ReadBuf	function ReadBuf(var Buf; Count: Integer): Integer; Buf is any variable, Count is an expression of type Integer. Read reads Count or fewer bytes from the com port. The actual number of bytes read (less than or equal to Count) is returned in its Result. If a read operations fails it returns 0. Never try to read bytes manually by specifying the count parameter yourself. Use the Count parameter returned in "OnData Event".
ReadChar	function ReadChar(var Ch: AnsiChar): Boolean; ReadChar returns a single character. If there are no characters waiting in the receiver buffer ReadChar returns false.
ReadText	function ReadText: string; ReadText returns all received characters as a string. If there are no characters waiting in the receiver buffer an empty string is returned.
ResetDev	ResetDev will reset the serial device/hardware if possible. This is handled by Windows itself and TVaComm only supplies the routine.
SetBreak	procedure SetBreak(Value: Boolean); Sets or clears the break signal state. When set to True, the SetBreak method sends a break signal. The break signal suspends character transmission and places the transmission line in a break state until you call SetBreak with False.
SetDTR	procedure SetDTR(Value: Boolean); Calling SetDTR with false clears the DTR (data-terminal-ready) signal. Otherwise sends the DTR (data-terminal-ready) signal. In most cases, setting the Data Terminal Ready line to low (false) hangs up the telephone.
SetRTS	procedure SetRTS(Value: Boolean);

	Calling SetRTS with false clears the RTS (request-to-send) signal. Otherwise SetRTS sends the RTS (request-to-send) signal.
SetXOn	procedure SetXon(Value: Boolean); Calling SetXOn with false causes transmission to act as if an XOFF character has been received. Calling with True causes transmission to act as if an XON character has been received.
WriteBuf	function WriteBuf(var Buf; Count: Integer): Integer; Buf is any variable, Count is an expression of type Integer. Write writes Count or fewer bytes to the com port from memory, starting at the first byte occupied by Buf. The actual number of bytes written (less than or equal to Count) is returned in its Result. If a write operation fails it returns 0. If however DirectWrite is set to false an exception is raised if the data block is too large to be written. (Increase the writebuffer size)
WriteChar	function WriteChar(Ch: Char): Boolean; WriteChar writes a single character to the serial port. If WriteChar succeeds it return true, otherwise it will return false.
WriteText	function WriteText(const s: string): Boolean; WriteText writes a single string to the serial port. If WriteText succeeds it return true, otherwise it will return false.

TVaBuffer

TVaBuffer is derived from TVaCommComponent and must be hooked up with a TVaComm component in order to operate. To start using TVaBuffer, drop a TVaComm and TVaBuffer instance on the form and assign VaComm to VaBuffer.Comm.
TVaBuffer is an external circular data buffer (FIFO) used to store large amount of data received through the COM port. Besides reading the buffer, it also allows previewing of data without removing the characters from the buffer or you can write your own characters into the buffer which is usefull writing macro based events.

Properties

Active	When true, TVaBuffer listens to the data received or to be sent via TVaComm and retrieves it.
Comm	Sets the TVaComm instances the buffer uses to receive or send data
Full	Full is True when there is no more free space available to store/receive data.Trying to write data to the buffer will result in an Overflow event.
Size	Size describes the maximum buffer size. Make sure to set Size large enough in order to buffer all received bytes.

Events

OnChange	OnChange will occur if characters arrive in the buffer or when bytes are read from the buffer.
OnOverFlow	OnOverflow will occure if there is not enough room in the buffer to store all incoming bytes.

Methods

BufFree: integer	Returns the number of free characters in the buffer. This property is not available at design time. BufFree refers to the number of bytes that can be received.
BufUsed: integer	Returns the number of characters waiting in the buffer. This property is not available at design time. BufUsed refers to the number of bytes that have been received by the modem and are waiting in the buffer for you to take them out.
Clear	Removes all data available in the buffer.
Peek	function Peek(var Buf; Count: Integer): Boolean; Buf is any variable, Count is an expression of type Integer. Use Peek in order to read bytes stored in the buffer without removing them.
Remove	function Remove(Count: Integer): Boolean; Remove count bytes from the buffer.
Read	function Read(var Buf; Count: Integer): Boolean; Buf is any variable, Count is an expression of type Integer. Read reads Count bytes from the buffer.
Write	function Write(const Buf; Count: Integer): Boolean;

	Buf is any variable, Count is an expression of type Integer. Write writes Count or fewer bytes to the buffer from memory, starting at the first byte occupied by Buf.

TVaCapture

TVaCapture is derived from TVaCommComponent and must be hooked up with a TVaComm component in order to operate. To start using TVaCapture, drop a TVaComm and TVaCapture instance on the form and assign VaComm to VaCapture.Comm. The TVaCapture component is used to receive ASCII text messages. In comparison to TVaWaitMessage, TVaCapture can extract and store parts of an incoming message (protocol messages). Therefore each message must contain a header or prefix set of characters and so called terminator character(s). The prefix and terminator parts are defined with the DataStart and the DataFinish properties.

TVaCapture also handles control characters embedded in the DataStart and the DataFinish string properties. eg ^M = #13

TVaCapture can also have pattern matching specifiers for DataStart and DataFinish. This type of pattern matching expression is used when the Data*Type property is set to dtExpression

When dtExpression is chosen following special characters can be used:

A: an alphabetic uppercase character is accepted at this position
a: an alphabetic lowercase character is accepted at this position
?: just any character is accepted at this position
0: a numeric character is accepted at this position
\x: only this literal character x is accepted at this position

Expression examples:

\X000: the accepted pattern is a character X followed by 3 numbers

\A\Taa: the accepted pattern is a character A followed by T followed by 2 lowercase characters

Properties

Active	When true, TVaCapturer listens to the data received or to be sent via TVaComm and retrieves it to find matching patterns
Comm	Sets the TVaComm instances the buffer uses to receive or send data
DataStart	Sets the prefix part of a protocol message with the DataStart property.
DataStartCase	When DataStartType is dtExpression, this defines whether the expression needs to be interpreted with case sensitivity or not
DataStartType	When type is dtString, the DataStart must be identical with the incoming datastream on the COM port for it to trigger a match. When type is dtExpression, the DataStart is treated as an expression and a match based on this expression is done.
DataFinish	Sets the suffix part of a protocol message with the DataFinish property.
DataFinishCase	Same purpose as DataStartCase but for the data end pattern
DataFinishType	Same purpose as DataStartType but for the data end pattern
MaxMsgLength	The data between DataStart and DataFinish is stored. In order to prevent an overflow of the message buffer MaxMsgLen is used.

Events

OnMessage	TVaCapture can extract and store parts of an incoming message (protocol messages). Therefore each message must contain a header or prefix set of characters and so called terminator byte(s). The data between DataStart and DataFinish is stored. As soon as DataFinish is received the OnMessage event handler is called with the received message as a parameter.

TVaModem

TVaModem provides device-independent routines for accessing modems. It provides methods for initializing, configuring, dialing, answering, and other common modem functions.

Properties

Active	When true, TVaModem listens to the data received or to be sent via TVaComm and retrieves it.
AnswerTimeOut	AnswerTimeout describes the time in milliseconds for answering incoming calls.
CharDelay	CharDelay sets the time in milliseconds while sending characters to the attached modem. TVaModem will add a short pause between each character.
CommandTimeOut	CommandTimeout sets the time in milliseconds TVaModem waits for a response from the attached modem after sending modem commands like Init and Reset.

Config	Contains a name/value pair list of AT modem command strings for typical actions:
	
	Commands:
	RESETCMD=Resets the modem
	DIALCMD=Prefix for dialing a phone number
	DIALTERM=Terminator character for the dial string parameter
	HANGUPCMD=Command to hangup the modem
	INITCMD=Configures the modem
	ANSWERCMD=Used to answer an incoming call
	
	Responses:
	ROK=
	RCONNECT=
	RBUSY=
	RVOICE=
	RNOCARRIER=
	RNODIALTONE=
	RERROR=
	RRING=
	
	Default configuration is:
	
	RESETCMD=ATZ^M
	DIALCMD=ATDT
	DIALTERM=^M
	HANGUPCMD=+++~~~ATH0^M
	INITCMD=ATZE1M1^M
	ANSWERCMD=ATA^M
	ROK=OK
	RCONNECT=CONNECT
	RBUSY=BUSY
	RVOICE=VOICE
	RNOCARRIER=NO CARRIER
	RNODIALTONE=NO DIALTONE
	RERROR=ERROR
	RRING=RING
	RTERM=^M

DialTimeOut	DialTimeout describes the time in milliseconds for dialing operations.
DtrDropDelay	Timeinterval used while performing a DTR modem hangup.

RingDetect	Returns the ring waiting status of the modem. Return values can be: rdLine, rdMsg, rdNone rdLine TVaModem waits for a hardware ring signal returned by the comport component. rdMsg TVaModem will wait for the modem to return a modem ring response. rdNone TVaModem will not respond to any ring events.
RingWaitTimeOut	RingWaitTimeout describes the time in milliseconds between two ring events or messages. After a RingTimeout answering an incoming call is aborted and the ring count is set to zero.

Events

OnAnswerTimeOut	OnAnswerTimeout is called as soon the time expires indicated by AnswerTimeout while answering an incoming phone call.
OnBusy	OnBusy is called as soon as TVaModem receives the busy message response from the attached modem.
OnCommandTimeOut	OnCommandTimeout is called as soon the time expires indicated by CommandTimeout while sending command strings to the attached modem.
OnConnect	OnConnect is called as soon as TVaModem receives the Connect Message repsonse from the attached modem. In most cases the Connect keyword is followed by connection parameters like baudrate and protocol. The received connect string message is stored in the ConnectString property which is also passed through with the connectstring parameter.
OnDialTimeOut	OnDialTimeout is called as soon the time expires indicated by DialTimeout while dialing a phone number until connect.
OnError	OnError is called as soon as TVaModem receives the error Message response from the attached modem.
OnNoCarrier	OnNoCarrier is called as soon as TVaModem receives the NoCarrier Message response from the attached modem.
OnNoDialTone	OnNoDialTone is called as soon as TVaModem receives the NoDialTone Message response from the attached modem.
OnOK	OnOK is called as soon as TVaModem receives the OK message repsonse from the attached modem.
OnRingDetect	The OnRingDetect event is called as soon as a ring event is received from the attached modem. The Rings parameter shows the number of rings received. With the AcceptCall parameter you can answer the incoming call. If AcceptCall is true TVaModem will answer then incoming call by sending the Answer string.
OnRingWaitTimeOut	OnRingWaitTimeout is called as soon the time expires indicated by RingWaitTimeout while waiting for an additional ring event.
OnVoice	OnVoice is called as soon as TVaModem receives the Voice Message repsonse from the attached modem.

Methods

Answer	Answer causes the modem to answer an incoming call.

	This routine sends the string specified by the AnswerCmd property to the modem. It then sets up the modem dispatcher to wait until a connection has been received or a certain number of seconds (determined by the AnswerTimeout property) has elapsed. An answer operation can be cancelled at any time by calling Cancel.
Cancel	Cancel will abort all pending modem operations like dialing or answering.
Dial	Dial sends the string specified by the DialCmd property to the modem, followed by the Number parameter. It then initializes the modem component's triggers to wait for a connection or until "DialTimeout" seconds have elapsed. A dial operation can be cancelled at any time by calling Cancel.
Hangup	Hangup breaks an existing modem connection. Hangup sends the string specified by HangupCmd to the modem. This is a command option only. If the DropDTR parameter is true the DTR method of the TVaComm component is used instead in order to perform a hardware hangup, also called dropping the DTR signal.
Init	The Init method sends the string specified by the Config property to the modem. Then it will begin waiting for an OK or ERROR response. Because of this, Init does not return immediately. Instead it sends the INICMD command to the modem, waiting for each modem response to be properly handled before returning. Init sends all of the configuration strings even if one or more returns an error.
PutCommand	PutCommand sends any command string specified by Cmd to the modem. A command string can contain normal characters or special characters. A special character is prefixed with a '^', which indicates that the modem should send the control character specified by the character after the '^'. For instance, the command string ATZ^M would send the characters 'ATZ' followed by ^M, or ASCII 13.
Reset	The Reset method sends the string specified by the Config property to the modem. Then it will begin waiting for an OK or ERROR response.
WaitForResponse	WaitForResponse is used for waiting for modem responses. The Delay parameter is the time in milliseconds before a timeout event occurs. Use WaitForResponse in combination with the PutCommand method. WaitForResponse returns immediatly. Use Cancel in order to abort the pending WaitForResponse operation.

TVaXModem

XModem is a protocol for transferring files during direct dial-up communications. Developed by Ward Christensen in 1977, XModem has basic error checking to ensure that information isn't lost or corrupted during transfer; it sends data in 128-byte blocks. XModem has undergone a couple of enhancements: XModem CRC uses a more reliable error-correction scheme, and XModem-1K transfers data faster by sending it in 1,024-byte blocks.

To start using the XModem protocol, set VaXModem.Comm to the VaComm instance handling the communication. The select the operation to perform, i.e. upload of download with the VaXModem.Mode property, set the filename and call Execute.

Example:

Upload
```
  VaXModem1.FileName := ‘filetoupload’;
  VaXModem1.Mode := tmUpload;
  VaXModem1.Execute;
```

Download
```
  VaXModem1.FileName := ‘filetodownoad’;
  VaXModem1.Mode := tmDownload;
  VaXModem1.Execute;
```

Properties

BufferSize	BufferSize is used to define the internal size of the protocol's receiver buffer. Default is 4096.
ErrorCode	ErrorCode is a read-only property describing the last type of error which has occured.
ErrorCount	ErrorCount is a read-only property describing the number of errors which have occured.
ExitCode	ExitCode is used to indicate the status of the file transfer. ExitCode is set as soon as the transfer has ended. It can stop due to an error or because the transfer was completed succesfully. Possible errorcodes 0:no error, transfer completed successfully. -1:a timeout occured -2:an error occured while writing to the comport (obsolete) -3:Local system cancelled the transfer -4:Remote system cancelled the transfer -5:Out of sync, unexpected data received -6:A checksum or CRC error occured (obsolete) -7:Error reading data from a file stream -8:Error writing data to a file stream -9:Maximum errors reached -10:An invalid startup code was received -11:Error creating a file for writing -12:Error opening a file for reading -14:A critical buffer overrun occured -15:An invalid or unexpected packet received -16:An invalid or unknown character received
FileName	FileName is the name of the file to transfer to or from another computer.

	Before calling upload or download to start the transfer a valid filename must be specified.
MaxErrors	MaxErrors sets the number of errors which may occure during a file transfer.
Mode	Use tmUpload for sending and tmDownload for receiving.
Protocol	The protocol type can be any of following values: XModem, XModem1K, XModemG XModem 128k block with simple checksum Each block is acknowledged with an ACK. One file at a time transfers only XModem1K 1024k block with 32bit CRC checking Each block is acknowledged with an ACK. One file at a time transfers only XModem-G 1024k block with 32bit CRC checking There is no acknowledge with an ACK. One file at a time transfers only
Timeout	Timeout is the time in milliseconds used to wait for a single incoming character. If a timeout occures the OnError event handler is called.

Events

OnError	OnError is called as soon an error occures during a file transfer. ErrorCode contains the last error. See also ExitCode for the possible values.
OnFileInfo	OnFileInfo is called as soon as a file is opened (upload) or created (download).
OnPacketEvent	OnPacketEvent is called during file transfers to indicate the progress made. Packet is the current packet number and ByteCount describes the number of bytes received.
OnTransferStart	OnTransferStart is triggered as soon as one of the methods Upload or Download are called.
OnTransferEnd	OnTransferEnd is called when a file transfer is complete or a when critical error occured.

Methods

Cancel	Use cancel to abort a file transfer in progress.
Execute	Depending on the Mode property Execute will send or receive a file to or from a remote system.

TVaYModem

YModem is a protocol for transferring files during direct dial-up communications. So named because it builds upon the earlier XModem protocol, YModem sends data in 1,024-byte blocks and is consequently faster than XModem. However, it doesn't work well on noisy phone lines, unlike its successor, ZModem. YModem has undergone a few enhancements: YModem-Batch can send several files in one session; YModem-G drops software error correction, which speeds up the process by leaving hardware-based error correction in modems.

To start using the YModem protocol, set VaYModem.Comm to the VaComm instance handling the communication. The select the operation to perform, i.e. upload of download with the VaYModem.Mode property, set the filename and call Execute.

Example:

```
Upload
  VaYModem1.Files.Clear;
  VaYModem1.Files.Add(file1);
  VaYModem1.Mode := tmUpload;
  VaYModem1.Execute;

Download
  VaYModem1.TargetFolder := ‘folderwheretoreceive’;
  VaYModem1.Mode := tmDownload;
  VaYModem1.Execute;
```

Properties

BufferSize	BufferSize is used to define the internal size of the protocol's receiver buffer. Default is 4096.
ErrorCode	ErrorCode is a read-only property describing the last type of error which has occured.
ErrorCount	ErrorCount is a read-only property describing the number of errors which have occured.
ExitCode	ExitCode is used to indicate the status of the file transfer. ExitCode is set as soon as the transfer has ended. It can stop due to an error or because the transfer was completed succesfully. Possible errorcodes 0:no error, transfer completed successfully. -1:a timeout occured -2:an error occured while writing to the comport (obsolete) -3:Local system cancelled the transfer -4:Remote system cancelled the transfer -5:Out of sync, unexpected data received -6:A checksum or CRC error occured (obsolete) -7:Error reading data from a file stream -8:Error writing data to a file stream -9:Maximum errors reached -10:An invalid startup code was received -11:Error creating a file for writing -12:Error opening a file for reading -14:A critical buffer overrun occured -15:An invalid or unexpected packet received -16:An invalid or unknown character received

Files	Files is a list of FileNames. In compare to the XModem protocol YModem supports batch transfers. This means you can sent multiple files in one call. For downloading the received filenames are stored in the Files property. For uploading all filenames must be added.
MaxErrors	MaxErrors sets the number of errors which may occure during a file transfer.
Mode	Use tmUpload for sending and tmDownload for receiving.
Protocol	The protocol type can be any of following values: YModem, YModemG YModem batch 1024k block with 32bit CRC checking Each block is acknowleged with an ACK. Supports multiple files YModemG batch 1024k block with 32bit CRC checking No block is acknowleged with an ACK. Supports multiple files
TargetFolder	Sets the folder where incoming files will be downloaded.
Timeout	Timeout is the time in milliseconds used to wait for a single incoming character. If a timeout occures the OnError event handler is called.

Events

OnError	OnError is called as soon an error occures during a file transfer. ErrorCode contains the last error. See also ExitCode for the possible values.
OnFileInfo	OnFileInfo is called as soon as a file is opened (upload) or created (download).
OnPacketEvent	OnPacketEvent is called during file transfers to indicate the progress made. Packet is the current packet number and ByteCount describes the number of bytes received.
OnTransferStart	OnTransferStart is triggered as soon as one of the methods Upload or Download are called.
OnTransferEnd	OnTransferEnd is called when a file transfer is complete or a when critical error occured.

Methods

Cancel	Use cancel to abort a file transfer in progress.
Execute	Depending on the Mode property Execute will send or receive a file to or from a remote system.

TVaZModem

The ZModem file tranfer protocol provides reliable file tranfsers with complete data integrity between application programs. ZModem's 32 bit CRC catches errors that continue to sneak into the even most advance networks.

ZModem provides advanced file management features including AutoDownload (Download initiated without user intervention) and crash recovery.

To start using the ZModem protocol, set VaZModem.Comm to the VaComm instance handling the communication. The select the operation to perform, i.e. upload of download with the VaZModem.Mode property, set the filename and call Execute.

Example:

```
Upload
  VaZModem1.Files.Clear;
  VaZModem1.Files.Add(file1);
  VaZModem1.Mode := tmUpload;
  VaZModem1.Execute;

Download
  VaZModem1.TargetFolder := ‘folderwheretoreceive’;
  VaZModem1.Mode := tmDownload;
  VaZModem1.Execute;
```

Properties

BufferSize	BufferSize is used to define the internal size of the protocol's receiver buffer. Default is 4096.
ErrorCode	ErrorCode is a read-only property describing the last type of error which has occured.
ErrorCount	ErrorCount is a read-only property describing the number of errors which have occured.
ExitCode	ExitCode is used to indicate the status of the file transfer. ExitCode is set as soon as the transfer has ended. It can stop due to an error or because the transfer was completed succesfully. Possible errorcodes 0:no error, transfer completed successfully. -1:a timeout occured -2:an error occured while writing to the comport (obsolete) -3:Local system cancelled the transfer -4:Remote system cancelled the transfer -5:Out of sync, unexpected data received -6:A checksum or CRC error occured (obsolete) -7:Error reading data from a file stream -8:Error writing data to a file stream -9:Maximum errors reached -10:An invalid startup code was received -11:Error creating a file for writing -12:Error opening a file for reading -14:A critical buffer overrun occured -15:An invalid or unexpected packet received -16:An invalid or unknown character received

	-17:File seek error -19:Unknown Header received -20:Remote resent header request -21:Remote filesystem error occured -22:Application terminated (force exit) -23:Data packet to large (up to 1024 bytes) -24:Remote terminated to early -25:Error in receiving command sequence
Files	Files is a list of FileNames. In compare to the XModem protocol YModem supports batch transfers. This means you can sent multiple files in one call. For downloading the received filenames are stored in the Files property. For uploading all filenames must be added.
MaxErrors	MaxErrors sets the number of errors which may occure during a file transfer.
Mode	Use tmUpload for sending and tmDownload for receiving.
TargetFolder	Sets the folder where incoming files will be downloaded.
Timeout	Timeout is the time in milliseconds used to wait for a single incoming character. If a timeout occures the OnError event handler is called.

Events

OnError	OnError is called as soon an error occures during a file transfer. ErrorCode contains the last error. See also ExitCode for the possible values.
OnFileInfo	OnFileInfo is called as soon as a file is opened (upload) or created (download).
OnPacketEvent	OnPacketEvent is called during file transfers to indicate the progress made. Packet is the current packet number and ByteCount describes the number of bytes received.
OnTransferStart	OnTransferStart is triggered as soon as one of the methods Upload or Download are called.
OnTransferEnd	OnTransferEnd is called when a file transfer is complete or a when critical error occured.

Methods

Cancel	Use cancel to abort a file transfer in progress.
Execute	Depending on the Mode property Execute will send or receive a file to or from a remote system.

TVaWaitMessage

TVaWaitMessage is derived from TVaCommComponent and must be hooked up with a TVaComm component in order to operate. The TVaWaitMessage component is used to receive simple ASCII text messages. As soon as a predefined message is received the OnMessage event is triggered. TVaWaitMessage also handles control characters embedded in the Strings property. eg ^M = #13

Properties

Active	When true, TVaBuffer listens to the data received or to be sent via TVaComm and retrieves it.
CaseSensitive	CaseSensitive determines if a string must match it's uppercase and lowercase characters. If CaseSensitive is False, TVaWaitmessage compares the strings passed to it case-insensitively.
Comm	Sets the TVaComm instances the buffer uses to receive or send data
Strings	Strings is a list of ASCII messages to wait for.

Events

OnMessage	The OnMessage event is called as soon as an ascii message defined in the strings property is received. Index will indicate which message it concerns.

Methods

ResetStrings	ResetStrings will clear all received data and puts the component in its initial state.

TVaServer & TVaServerClient

A TVaServer is used in combination with a TVaServerClient component. TVaServerClient is derived from TVaCustomComm and therefore contains all properties, methods and events used by the regular TVaComm component. Both TVaServer & TVaServerClient components are used to build a comm server application which can support as many COM ports as required.

First define the server client and intialize all other properties. Finally you can hook it up with a server component. This can be done in design-time as well during run-time. To hook up a client to a server, set VaServerClient.Server := VaServer;

Due to the many different types of hardware there is no hardware specific functionallity inside the server component except for the BroadCast method. This means that each client must be configured and opened manually.

TVaServer Properties

ClientCount	Public property to retrieve the number of connected clients
ServerClient[Index: integer]	Array of connected clients TVaServerClient to the server

TVaServer Events

OnClientBreak	Event triggered when client receives a break
OnClientCts	Event triggered when client cts signal raises
OnClientDtr	Event triggered when client dtr signal raises
OnClientError	Event triggered when client receives an error
OnClientRing	Event triggered when client ring signal raises
OnClientRlsd	Event triggered when client rlsd signal raises
OnClientRxChar	Event triggered when client receives a char
OnClientRxFlag	Event triggered when client receive flag raises
OnClientTxEmpty	Event triggered when client transmit empty flag raises

TVaServer Methods

Broadcast	procedure Broadcast(var Buf; Count: Integer); Broadcast will sent data defined by Buf to all active server clients.
CloseAll	Closes all active server clients.

TVaServerClient Properties

TVaServerClient inherits all properties, methods and events from TVaCustomComm and adds one extra property:

Server	Server describes the TVaServer component which the component is linked to.

TVaTerminal

TVaTerminal is a visual control that can function as a terminal. When connected to a TVaComm instance, it can act as a terminal control to interact with the COM port. Default, TVaTerminal displays the incoming characters from the COM port and will send the characters typed to the COM port. When an Emulation component is connected to TVaTerminal.Emulation, special character sequences are interpreted, for example, character sequences to set colors or blinking that the TVaANSIEmulation is able to handle.

Properties in addition to standard TControl properties

Blinktime	Time in milliseconds for blinking characters to turn on and off.
BufferBackground	BufferBackground[X, Y: Integer]: TColor Returns the background color at X,Y position in the receive buffer
BufferChar	BufferChar[X, Y: Integer]: AnsiChar Returns the character at X,Y position in the receive buffer
BufferColor	BufferColor[X, Y: Integer]: TColor Returns the text color at X,Y position in the receive buffer
Capture	When true, the incoming data stream from the connected TVaComm is retrieved and stored in a file specified by CaptureFile.
CaptureAppend	When true, the data is appened to the CaptureFile if it already contained data, otherwise, a new CaptureFile is created.
CaptureFile	Sets the file to use to store all incoming data.
CaretX	Returns the cursor X position
CaretY	Returns the cursor Y position
Character	Character[X, Y: Integer]: AnsiChar Returns the character at X,Y position in the terminal
CharacterBackground	CharacterBackground[X, Y: Integer]: TColor Returns the background color at X,Y position in the terminal
CharacterColor	CharacterColor[X, Y: Integer]: TColor Returns the text color at X,Y position in the terminal
Color	Sets the default background color of the terminal.
Columns	Sets the maximum number of characters that can be horizontally displayed in the terminal.
Comm	Sets the instance of TVaComm through which the TVaTerminal will communicate.
CursorType	Selects the type of cursor. Cursor type can be crsBlock, crsUnderline or crsNone.
Emulation	Sets the non-visual emulation component that can be used with TVaTerminal. With TMS Async, TVaTTYEmulation and TVaANSIEmulation are included.
LocalEcho	When true, the characters typed are also displayed in the terminal, otherwise, charaters typed are only sent via TVaComm.
LocalEchoColor	Sets the color of the characters typed in TVaTerminal.
Rows	Sets the maximum number of rows that can be vertically displayed in the terminal.
ScrollBack	When true, scrolling back in the terminal is enabled.
ScrollBackRows	Sets the maximum number of rows that can be scrolled back.
WantAllKeys	When true, special characters like TAB,CR, ESC are also sent.

Events in addition to standard TControl properties

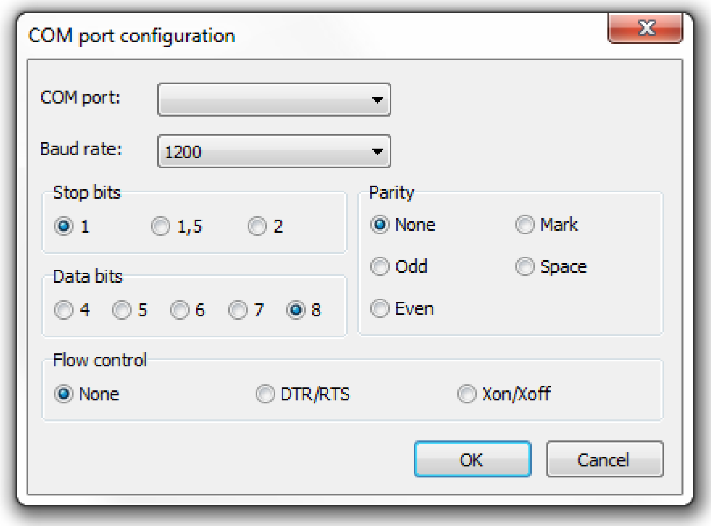
OnDataReceived	Event triggered when the TVaTerminal client has received data from TVaComm

Methods

AddText	procedure AddText(const Text: string); Adds text at cursor position to the terminal and sends it via TVaComm
AddControlText	procedure AddControlText(const Text: string; const Color, Bkg: TColor); Adds text at cursor position to the terminal with specified text and background color and sends it via TVaComm
Clear	Clears the terminal
ClearAll	Clears the terminal + its internal buffers
ClearAllBuffer	Clears the terminal + its internal buffers without repainting
ClearScreenBuffer	Clears the terminal without repainting
GotoXY	Moves the caret to X,Y on the terminal
InsertAndScrollF	procedure InsertAndScrollF(nrLines: Integer); Inserts void lines and moves screen pointer down. Updates the display.
Scroll	function Scroll(nrLines: Integer): Integer; Moves back screen pointer; returns the number of lines actually scrolled (depends on the nr. of scrollback rows). Returns the number of rows effectively scrolled back.
WriteBuf	function WriteBuf(var Buf; Count: Integer): Integer; Sends the data to the connected TVaComm
WriteChar	function WriteChar(Ch: AnsiChar): Boolean; Sends the character to the connected TVaComm
WriteText	function WriteText(const s: AnsiString): Boolean; Sends the text to the connected TVaComm

TVaConfigDialog

A TVaConfigDialog facilitates adding a dialog for end-users to configure serial port settings without needing to write any code.To use the component, drop TVaConfigDialog on the form, assign a TVaComm instance to VaConfigDialog.Comm and show the configuration dialog via VaConfigDialog.Execute.



COM port:

Baud rate: 1200

Stop bits: [✓ 1] 1,5 2

Data bits: 4 5 6 7 [✓ 8]

Parity: [✓ None] Mark Odd Space Even

Flow control: [✓ None] DTR/RTS Xon/Xoff

TVaConfigDialog will display the COM port settings as defined in the connected TVaComm component and after changing the settings via the dialog and closing the dialog with OK, the updated settings will be applied to TVaComm.