Remote COM's Best Opportunity for Remote Serial Communication, via Internet by Stand Alone Program or browser through World Wide Web

LICENSE AGREEMENT
For
RemoteCOM

IMPORTANT - PLEASE READ CAREFULLY:

This License Agreement ("Agreement") is a legally binding agreement between you and Decision Computer International Co., Ltd. ("Decision") for the PC COM software identified above ("Software"), unless you have a signed license agreement with Decision in other forms. By downloading, installing, copying or otherwise using the Software, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, you shall NOT download, install, copy or otherwise use the Software.

The Software is protected by copyright laws and international copyright treaties, as well as by other intellectual property laws and treaties. The Software is licensed, not sold.

1. Grant of License

Subject to the terms and conditions of this Agreement, Decision grants to you a non-exclusive license to:

a. Install and use the Software on a hard disk or other storage device, such as network server.

b. Make and distribute unlimited copies of the Software, including copies for bundling with the commercial distribution of other software, provided that each copy that you make or distribute shall contain this Agreement and the copyright and other proprietary notices that appear in the Software, and further provided that any commercial distribution of the Software or any other software in bundles shall obtain Decision's written consent in advance.

2. Termination

a. Decision expressly reserves the right to terminate, at its sole discretion and without prior notice, the license hereby granted to you. Upon Decision's termination of the license, all your rights and licenses shall be revoked forthwith.

b. The license granted to you under this Agreement shall automatically terminate upon any failure of you to comply with any of the terms of this Agreement.

3. Limitations and Restrictions

You may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software. In addition, you may not use the Software to develop, adapt, or otherwise modify another OCX-type software.

4. Copyright and Trademark Rights

All title and copyrights in and to the software and its documentation are owned by Decision and protected by copyright laws and international treaty provisions. [Decision and PC COM] are registered trademarks of Decision, you may use the trademarks to identify that you use the software, but such use does not give you any rights of ownership in the Software and its trademarks.

5. No Warranty

THE SOFTWARE IS LICENSED TO YOU AS IS. DECISION DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE PERFORMANCE OR RESULTS THAT YOU MAY OBTAIN BY USING THE SOFTWARE, NON-INFRINGEMENT OF THIRD PARTY RIGHTS. MERCHANTIBILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE WITH RESPECT TO THE SOFTWARE.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, UNDER NO CIRCUMSTANCES SHALL DECISION BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, OR SPECIAL DAMAGES WHATSOEVER ARISING OUT OF OR IN CONNECTION WITH THE USE OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO, LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY CLAIM BY ANY THIRD PARTY.

6. Governing Law

This Agreement is governed by the laws of the Republic of China, Taiwan, without regard to its conflict rules of laws.

YOUR ACCEPTANCE OF THIS AGREEMENT WILL BE INDICATED ONCE YOU PROCEED TO DOWNLOAD OR INSTALL THE SOFTWARE.

Remote COM solution will enhance RS 232/422 serial communication through Internet by stand-alone program or by browser through the World Wide Web. Provides data or information to monitor and or control all the data and signals from the interacting devices or a single device over the Internet. Use an unsurpassed tool for RS-232/422 application, communication control and or analysis. There are two nodes...

1) Server – provides communication between standard Com 2 or PCCOM PCI Multi port Card and the client (if more than one port).

   You need Port Local on Server Setup, Port Local must be the same in the client Port Local.

2) Client – monitors or controls available RS-232/422 serial port over Internet connection.

   User may require knowing the TCP/IP address of the Server, in order to monitor or control available RS-232/422 serial port.

Note:

Application Program(Internet) - is an application that user need to install the software, in order to operate, monitor and control, Internet connection is required.

World Wide Web - a web site which the application is ready to load, a HTML file were function are the same with the application program. User my use where ever as long as there is a connection with the Internet browser (like Netscape and Internet Explorer)

Operating System

Windows 95 / 98 / NT environment

Specification

RemoteCOM OCX can use any windows programming that uses OCX like Visual Basic, Visual C++, Visual J++, etc…

Specification

1) UART Configuration - modify DTR and RTS, adjust

2) UART Monitored - Performs serial-to-parallel conversion of data received from a peripheral device and the parallel-to-serial conversion of data. Signals status like RTS(Request To Send), CTS(Clear To Send), DSR(Data Set Ready), CD(Carrier Detect), DTR(Data Terminal Ready), and RI(Ring Indicate). UART Compatibility – 16450, 16550, and 16650.

3) Transmit Data - transmit character/s to any remote client, by passing the transmit buffer.

4) Receive Data - acquire data from an open serial port of the remote server. puke

5) Programmer's Friendly - user can operate in less time.

6) Flow Control Selectable - is a signal system, were data can communicate right receiving device time. This will use RTS, RTSXonXoff, Xon/Xoff, and None

7) Bits Per Second Selectable - a baudrate chooses 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, and 115200

8) Data Bits Selectable - 4,5,6,7 and 8

9) Parity Selectable - Even, Odd, None, Mark and Space

10) Com Port Selectable - any available configured ports, from 1 to 4 ports.

Note: User Com Port more than one Com at the same time.

Remote COM OCX Properties

(Index)

The Index property is to set the order of the creation of objects in a collection. The index for the first object in a collection will always be one. The indexes discussed in this section are arrays of variables, declared in code. They are different from the control arrays you specify by setting the Index property of controls at design time. Arrays of variables are always contiguous; unlike control arrays, you cannot load and unload elements from the middle of the array.

The indexes values are ranging from 0 to 15 only, where 0 refers to the first Port, 1 refers to the second Port and so on.

ComPort(index)

Sets and returns the communications port number

Syntax
object.**ComPort(index)**[= *value*]

The **ComPort(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	An integer value specifying the port number.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
You can set *value* to any number between 1 and 16 at design time. However, the **RemoteCOM** control generates error 68 (`Device unavailable`) if the port does not exist when you attempt to open it with the **CliSetSerialCom** method. **Index** value number from 0 to 15 (16 Communication Port), handling specific port to use.
Warning you must set the **ComPort(index)** property before envoking the CliSetSerialCom method.

Data Type
Integer

Sample Program
```
RemCom.ComPort(0)  =  2    'object remcom has set the 1st Port to
                           COM2
RemCom.ComPort(3)  =  5    'object remcom has set the 4th Port to
                           Com5
```

DataBits(index)
Sets and returns the data bit.

Syntax
object.**DataBits(index)** [= *value*]

The **DataBits(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A string expression representing the communications data bits, as described below.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
If *value* is not valid when the port is opened, the **RemoteCOM(0)** control generates error 380 (`Invalid property value`).
Value is the number of data bits.
The default value of *value* is: `"8"`

Setting
4
5
6
7
8 (Default)

Data Type
String

Sample Program
```
RemCom.DataBits(0)  =  8    'object remcom has set to DataBits 8
```

FlowControl(index)
Sets and returns the hardware handshaking protocol.

Syntax
object.**FlowControl(index)**[= *value*]

The **FlowControl(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	An integer expression specifying the handshaking protocol, as described in Settings.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Settings
The settings for *value* are:

Setting	Description
None	(Default) No handshaking.
OnXOff	XON/XOFF handshaking.
RTS	RTS/CTS (Request To Send/Clear To Send) handshaking.
RTSXOnXOff	Both Request To Send and XON/XOFF handshaking.

Remarks
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 **FlowControl** 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.

Data Type
String (Integer internally)

Sample Program
```
RemCom.FlowControl(0) = "None"   ‘object remcom has set the FlowControl to None
```

HostRemote
Returns or sets the remote machine to which a control sends or receives data. You can either provide a host name, for example, "FTP://ftp.microsoft.com," or an IP address string in dotted format, such as "100.0.1.1".

Syntax
object.**HostRemote** = *string*

The **RemoteHost** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
String	The name or address of the remote computer.

Remarks
When this property is specified, the **URL** property is updated to show the new value. Also, if the host portion of the URL is updated, this property is also updated to reflect the new value.

Sample Program
```
RemCom.HostRemote = "200.72.512.14"   'object remcom has set
                                      the HostRemote to given IP  address
```

InputBuffer(index)
Sets and returns the size of the receive buffer in bytes.

Syntax
object.**InputBuffer(index)**[= *value*]

The **InputBuffer(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	An integer expression specifying the size of the receive buffer in bytes.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
InputBuffer refers to the total size of the receive buffer. The default size is 1024 bytes.

Note :
Note that 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. Generally, start with a buffer size of 1024 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.

Data Type Integer

Sample Program
```
RemCom.InputBuffer(0) = 1025   'object remcom has set the Input       Buffer to 1025
```

InputLength(index)
Sets and returns the number of characters the **vDataReceive(0)** property reads from the receive buffer.

Syntax
object.**InputLength(index)** [= *value*]

The **InputLength(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	An integer expression specifying the number of characters the **vDataReceive(0)** property reads from the receive buffer.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
The default value for the **InputLength(0)** property is 0. Setting **InputLength(0)** to 0 causes the **RemoteCOM** control to read the entire contents of the receive buffer when **vDataReceive(0)** is used. If **InputLength(0)** characters are not available in the receive buffer, the **vDataReceive(0)** property returns a zero-length string (""). This property is useful when reading data from a machine whose output is formatted in fixed-length blocks of data.

Data Type
Integer

Sample Program
```
RemCom.InputLength(0) = 0   'object remcom has set the InputLength to 0
```

OutputBuffer(index)
Sets and returns the size, in bytes, of the transmit buffer.

Syntax
object.**OutputBuffer(index)** [= *object*]

The **OutputBuffer(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	An integer expression specifying the size of the transmit buffer.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
OutputBuffer(0) refers to the total size of the transmit buffer. The default size is 512 bytes.

Note 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. Generally, start with a buffer size of 512 bytes. If an overflow error occurs, increase the buffer size to handle your application's transmission rate.

Data Type
Integer

Sample Program
```
RemCom.OutputBuffer(0) = 512     'object remcom has set the
```

Output Buffer to 512

Parity(index)

Sets and returns the parity parameters.

Syntax
object.**Parity(index)** [= *value*]

The **Parity(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A string expression representing the communications port settings, as described below.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks

If *value* is not valid when the port is opened, the **RemoteCOM** control generates error 380 (`Invalid property value`).
The default value of *value* is: `"None"`

The following table describes the valid parity values.

Parity	Description
Even	Even parity
Mark	Mark parity
None	(Default) None parity
Odd	Odd parity
Space	Space parity

Data Type

String

Sample Program

```
RemCom.Parity(0)  = “None”     ‘object remcom has set the Parity
                                to None
```

PortRemote

Returns or sets the remote port number to connect to.

Syntax
object.**PortRemote** = *port*

The **PortRemote** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Port	The port to connect to. The default value of this property is 1024.

Data Type

Long

Remarks
The **PortRemote** property is set automatically to the appropriate default port for each protocol. Default port numbers are shown in the table below:

Port	Description
1024	Commonly used for World Wide Web connections of Decision Card.
21	FTP.

Sample Program

```
RemCom.PortRemote = 1024    'object remcom has set the Remote
                            Port to 1024
```

RxData(index)

Enable/Disable receiving data from the server.

Syntax

object.**RxData(index)**[= *value*]

The **RxData(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A Boolean expression specifying whether the Request To Send (RTS) line is enabled, as described in Settings.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Settings
The settings for *value* are:

Setting	Description
True	Enables to receive.
False	(Default) Disables to receive.

Remarks
When **RxData(index)** is set to **True**, the receiving of data is activated.

Data Type

Boolean

Sample Program

```
RemCom.RxData(0) = True    'object remcom has set the Receive Data to enabled
```

sConnectionStatus

Returns the state of the control, expressed as an enumerated type. Read-only and unavailable at design time.

Syntax
object.**sConnectionStatus**
using string Value
or
object.**ConnectionStatus**
using number Value (#)

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Data Type
String

Settings
The settings for the **sConnectionStatus** property are:

#	Value	Description
0	Closed	Default. Closed is the status
1	Open	Open is the status
2	Listening	Listening is the status
3	Connection pending	Connection pending is the status
4	Resolving host	Resolving host is the status
5	Host resolved	Host resolved is the status
6	Connecting	Connecting is the status
7	Connected	Connected is the status
8	Peer is closing the connection	Peer is closing the connection is the status
9	Error	Error is the status

SetDTR(index)

Determines whether to enable the Data Terminal Ready (DTR) line during communications on the server. 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.

Syntax
object.**SetDTR(index)**[= *value*]

The **SetDTR(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A Boolean expression specifying whether to enable the Data Terminal Ready (DTR) line, as described in Settings.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Settings
The settings for *value* are:

Setting	Description
True	Enable the Data Terminal Ready line.

False (Default) Disable the Data Terminal Ready line.

Remarks

When **SetDTR(index)** 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 **SetDTR(index)** is set to **False**, the Data Terminal Ready always remains low.

Note :In most cases, setting the Data Terminal Ready line to low hangs up the telephone.

Data Type
Boolean

Sample Program

```
RemCom.SetDTR(0) = True    'object remcom has set to Enable the Data Terminal Ready line
```

SetRTS(index)

Determines whether to enable the Request To Send (RTS) line. Typically, the Request To Send signal that requests permission to transmit data is sent from a computer to its attached modem.

Syntax
object.**SetRTS(index)**[= *value*]

The **SetRTS(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A boolean expression specifying whether the Request To Send (RTS) line is enabled, as described in Settings.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Settings

The settings for *value* are:

Setting	Description
True	Enables the Request To Send line.
False	(Default) Disables the Request To Send line.

Remarks

When **SetRTS(index)** is set to **True**, the Request To Send line is set to high (on) when the port is opened, and low (off) when the port is closed. The Request To Send line is used in RTS/CTS hardware handshaking. The **SetRTS(index)** property allows you to manually poll the Request To Send line if you need to determine its state.
For more information on handshaking protocols, see the **FlowControl(index)** property.

Data Type
Boolean

Sample Program

```
RemCom.SetRTS(0) = True    'object remcom has set to Enables the Request To Send line.
```

SignalEnable(index)

Determines whether to enable the UART (CD, CTS, DSR, RI) line. Typically, the Request To Send signal that requests permission to transmit data is sent from a computer to its attached modem.

Syntax

object.**SignalEnable(index)**[= *value*]

The **SignalEnable(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A boolean expression specifying whether the UART line is enabled.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Settings

The settings for *value* are:

Setting	Description
True	Enables the CD, CTS, DSR & RI.
False	(Default) Disables the CD, CTS, DSR & RI.

Remarks

When **SignalEnable(index)** is set to **True**, the CD, CTS, DSR & RI line is activated when the client is connected to the server and the serial setup is already done.
The **SignalEnable(index)** property allows you to determine its UART state.

For more information on handshaking protocols, see the **FlowControl(index)** property.

Data Type

Boolean

Sample Program

```
RemCom.SignalEnable(0) = True    'object remcom has set to Enables the CD, CTS, DSR & RI.
```

StopBits(index)

Sets and returns stop bit parameter.

Syntax

object.**StopBits(index)**[= *value*]

The **Settings(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.

Value	A string expression representing the communications port settings, as described below.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks

If *value* is not valid when the port is opened, the **RemoteCOM** control generates error 380 (`Invalid property value`).

The default value of *value* is:`"1"`

The following table lists the valid stop bit values.

Setting	
1	(Default)
1.5	
2	

Data Type
String

Sample Program

```
RemCom.StopBit(0) = “1”  ‘object remcom has set the Stop Bit to 1
```

TxData(index)

Sets the data to transmit.

Syntax
object.**TxData(index)**[= *value*]

The **TxData(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A string expression representing the RemoteCOM settings, as described below.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks

The default value of *value* is blank

Data Type

String

Sample Program

```
RemCom.TxData(0)  =  “R”    ‘object remcom has set the transmission with character R.
```

vDataReceiveX

Store data after receive.

Syntax
value = *[object*.**vDataReceiveX]**

The **vDataReceiveX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	The receive data
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Remarks
The default value of *value* is blank

Data Type
String

Sample Program
```
MyData = object.vDataReceive1     ' Variable MyData, store the
         Value receive
```

vDataReceiveX
Store data after receive.

Syntax
value = *[object*.**vDataReceiveX]**

The **vDataReceiveX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	The receive data
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Remarks
The default value of *value* is blank

Data Type
String

Sample Program
```
MyData = object.vDataReceive1     ' Variable MyData, store the Value recieve
```

PassWord
RemoteCOM password

Syntax
object.**PassWord** [= *value*]

The **PassWord** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	Constant value – password

Remarks
Decision Computer International Co., LTD. OCX password, trademark of the company. In order to run the application program, protection for any pirating act.

Default Password
Decision Computer

Data Type
String

Sample Program

```
RemCom.Password = "Decision Computer"    'object remcom has
                                          set OCX password
```

sNumberOfPort

Store Number of Port after the method GetNumberOfPort

Syntax
object.**sNumberOfPort**

The **sNumberOfPort** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.

Remarks
GetNumberOfPort, a method that evaluate available port in the Server, then store in **sNumberOfPort**.

Data Type
String

Sample Program

```
RemCom.sNumberOfPort    'object remcom stores the value done by the GetNumberOfPort

MyNumPort  =  RemCom.sNumberOfPort
```

BitsPerSecondX

Number of times per second that the carrier signal shifts value or a measurement of how fast data is moved from one place to another.

Syntax

object.**BitsPerSecondX** [= *value*]

The **BitsPerSecondX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A string expression representing the communications data bits, as described below.
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Remarks

For example a 1200 bit-per-second modem actually runs at 300 baud, but it moves 4 bits per baud (4 x 300 = 1200 bits per second).

Value

110	4800	38400
300	9600	57600
600	14400	115200 (default)
1200	19200	
2400	28800	

Data Type

String

Sample Program

```
RemCom.BitsPerSecond1  = 115200   'object RemCom has set baudrate to 115200
```

b_CTS_OnX

Indicates that the Modem or data set is ready to exchange data.

Syntax

object.**b_CTS_OnX**[= *value*]

The **b_CTS_OnX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A Boolean expression specifying whether to enable or disable clear to send
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Settings

The settings for *value* are:

Setting	Description
True	CTS(Clear to Send) is ready
False	CTS(Clear to Send) is not ready

Data Type

Boolean

Sample Program

```
RemCom.b_CTS_On1 = True    'object remcom has set ready to
                           send.
```

b_DSR_OnX

Indicates that the Modem or data set is ready to establish the communication link with the UART.

Syntax

object.**b_DSR_OnX**[= *value*]

The **b_DSR_OnX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A Boolean expression specifying whether to enable or disable data set ready.
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Settings

The settings for *value* are:

Setting	Description
True	DSR(Data Set Ready) enable data to set
False	DSR(Data Set Ready) disable data to set

Data Type

Boolean

Sample Program

```
RemCom.b_DSR_On1 = True    'object remcom has set Data to be
                           ready
```

b_RI_OnX

Ring Indicator, indicates that the Modem set has received a telephone-ringing signal.

Syntax

object.**b_CD_OnX**[= *value*]

The **b_CD_OnX** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Value	A Boolean expression specifying whether to enable or disable rings indicator.
X	Sets of number 1 to 16. Where 1 is for Port 1, 2 is for Port 2 and so on…

Settings
The settings for *value* are:

Setting	Description
True	RI(Ring Indicator) Enable
False	RI(Ring Indicator) Disable

Data Type
Boolean

Sample Program

```
RemCom.b_RI_On1 = True    'object remcom has set to Enable
                          ringing signal
```

REMOTE COM METHODS

SetServer

Sets the remote com OCX function to Server

Syntax
object.**SetServer**

The **SetServer** property syntax has these part:

Part	Description
Object	An expression that evaluates to an object.

Remarks
You can set OCX to server directly, during this setting a follow up for **PortLocal** which require the user to input Server Setup Local Port. Terminal Type for server must define as a Server.

Warning
Make sure that the **PortLocal** is not use by other device, Port Local 1024 is a default for the **RemoteCOM**, you may see local port information for sure.

Sample Program

```
RemCom.SetServer    'object remcom has set to configure Server
```

DisConnect

Sets function to disconnect communication between the Server and the Client.

Syntax
object.**DisConnect**
The **DisConnect** property syntax has these part:

Part	Description
Object	An expression that evaluates to an object.

Remarks

After setting up connection to client or to server, user can disconnect communication, data or information will no longer be attend.

Sample Program

```
RemCom.DisConnect    'object remcom has set to disconnect
communication
```

SetConnect

Sets Port and Host function, to identify connection.

Syntax

object.**SetConnect**

The **SetConnect** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.

Remarks

During operation **RemotePort** and **HostIP** is required, to allow and identify which Port to use and which Host computer to communicate. To view connection status, add status function.

Sample Program

```
RemCom.SetConnect  'object remcom has request to set connection
```

GetSerialSetup

Settings on serial COM, Flow Control, etc…

Syntax

object.**GetSerialSetup**

The **GetSerialSetup** property syntax has this part:

Part	Description
Object	An expression that evaluates to an object.

Remarks

To allow and determine features of serial communication this includes Bits Per Second, Data Bits, Parity, Stop Bits, Flow Control, and Serial Com Port.

Sample Program

```
RemCom.GetSerialSetup    'setting serial communication
```

CliTxData

Send data collected from TxData

Syntax

object.**CliTxData**

The **CliTxData** property syntax has this part:

Part	Description
Object	An expression that evaluates to an object.

Remarks
During transmission of Data, string or any Data Types must be store in **TxData,** after the storage data is ready to transmit.

Sample Program

```
RemCom.TxData = "The Quick Brown Fox"     ' Data to transmit
RemCom.CliTxData        'transmission of Data
```

CliOpenNthCom(index)

Open selected communication port number.

Syntax

object.**CliOpenNthCom(index)**

The **CliOpenNthCom(index)** property syntax has these parts

Part	Description
Object	An expression that evaluates to an object.
Method	A method expression representing the communications to Open Port number.
Index	Returns or sets the number that uniquely identifies an object in a collection.

Remarks
After connecting to the Server, during setup communication command, **CliOpenNthCom** will open selected Port Communication.

Sample Program

```
RemCom.CliOpenNthCom(0)     'object remcom stores the value done by the CliOpenNthCom(0)
```

CliCloseNthCom(index)

Close selected communication port number.

Syntax

object.**CliCloseNthCom(index)**

The **CliCloseNthCom(index)** property syntax has these parts:

Part	Description
Object	An expression that evaluates to an object.
Method	A method expression representing the communications to Close Port number.
Index	Returns or sets the number that uniquely identifies an object in a collection.

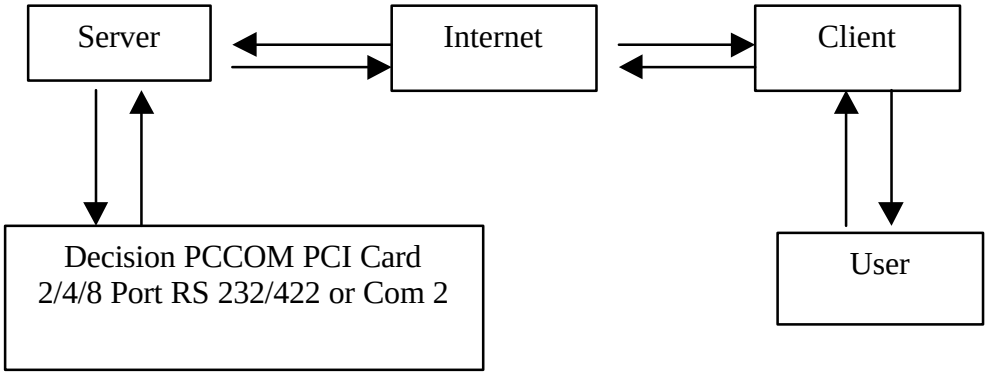
Remarks
Data assign or Open by **CliOpenNthCom(index)**, during unloading or a command using **CliCloseNthCom(index)**, will close current communication port.

Sample Program

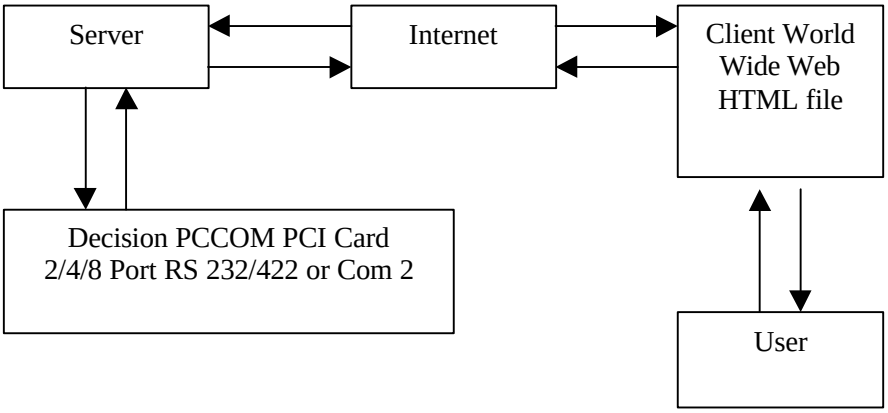
```
RemCom.CliCloseNthCom(0)     'object remcom assign to close current connected
communication port
```

BLOCK DIAGRAM

Application Program (Internet)



World Wide Web



Sample program
Using VB

CLIENT

Before running the client, make sure that the Server is already serving. Server IP address is required in order to communicate or sending Data from Client to Server. And also make sure that the Host Port Number are the same.

```
' Connect to Server, required IP address of the Server
Private Sub cmdConnect_Click()
   rcClient.TerminalType = "Client"
   rcClient.SetConnect
End Sub

' To set or assign Communication Port to the Server
Private Sub cmdSetup_Click()
   rcClient.GetSerialSetup
   rcClient.CliOpenNthCom (1)
End Sub

' Viewing Client Status
Private Sub cmdStatus_Click()
   rcClient.ConnectStatus
   MsgBox rcClient.sConnectionStatus
End Sub

' View available Port in the Server
Private Sub cmdNumCom_Click()
   rcClient.GetNumberOfPort
   MsgBox "The available ports is/are: " & Chr(13) & rcClient.sNumberOfPort
End Sub

' Disconnect from the Server
Private Sub cmdDisConnect_Click()
   rcClient.DisConnect
End Sub

' To Enabled and ready to Set Data
Private Sub cmdRead_Click()
   rcClient.CliRxRequest (1)
   tmrClient.Enabled = True
End Sub

'  Watch any character transmit to receive in the client
Private Sub tmrClient_Timer()
   txtReceive.Text = txtReceive.Text & rcClient.vDataReceive1 'vDataReceive1
   rcClient.vDataReceive1 = ""
End Sub
```

SERVER

```
'  Set to Server
Private Sub cmdConfigure_Click()
   rcserver.SetServer
```

```
End Sub

' To view current status
Private Sub cmdStatus_Click()
   rcserver.ConnectStatus
   MsgBox rcserver.sConnectionStatus
End Sub

' Refresh Server if there is any client who disconnect
Private Sub cmdReset_Click()
   rcserver.BeServer
End Sub

' Disconnect current operation
Private Sub cmdDisconnect_Click()
   rcserver.DisConnect
End Sub
```