
Tip/Ring Signal Simulator



TRsSim - Remote Control Programmers Guide

Copyright 2009 - Advent Instruments Inc. All rights reserved.

Printed in Canada

Advent Instruments Inc.
111 - 1515 Broadway Street
Port Coquitlam, BC, V3C6M2
Canada

Internet: techsupport@adventinst.com
 sales@adventinst.com

Web Site: <http://www.adventinstruments.com>

Telephone: (604) 944-4298
Fax: (604) 944-7488

Contents

1. Introduction	1
1.1. Files Included	1
2. Known Limitations	2
3. Establishing a Connection to TRsSim	3
4. TrsRmt .Net Assembly Reference	6
4.1. What's Included	6
4.2. System Requirements	6
4.3. Namespace	6
4.4. TrsRmt Object	7
4.4.1. Connect	7
4.4.2. Disconnect	8
4.4.3. IsConnected	8
4.4.4. SetTimeout	8
4.4.5. OpenFile	9
4.4.6. Reset	9
4.4.7. Exit	9
4.4.8. GetDevices	10
4.4.8.1. DeviceInfo Object	10
4.4.9. GetLogs	10
4.4.9.1. Log Object	11
4.4.10. SetScriptSource	11
4.4.11. SendScriptCommand	11
4.4.12. DoStatements	12
4.4.13. GetStatus	12
4.4.14. GetEvent	13
4.4.15. GetWaveformInfo	13
4.4.15.1. WaveformInfo Structure	14
4.4.16. GetWaveform	14
4.5. Event Object	15
4.6. Revision History	15
5. TrsRmt 'C' DLL Reference	16
5.1. Files Included	16
5.2. Coding Conventions	17
5.3. System and Error Handling Functions	18
5.3.1. trsrmt_Get_Version	18
5.3.2. trsrmt_Get_FuncErrorMsg	18
5.3.3. trsrmt_Get_ERROR_Msg	18
5.3.4. trsrmt_Get_WARNING_Msg	19
5.4. TCP/IP Connection Functions	20
5.4.1. trsrmt_Connect	20

5.4.2. trsrmt_Disconnect	20
5.4.3. trsrmt_Set_Timeout	20
5.5. Low-Level Communication Functions	22
5.5.1. trsrmt_SendCmd	22
5.5.2. trsrmt_WaitCmdResp	22
5.5.3. trsrmt_Get_RespLen	23
5.5.4. trsrmt_Get_Response	23
5.6. General Remote Control Functions	24
5.6.1. trsrmt_GET_DEVICES	24
5.6.2. trsrmt_Get_DeviceInfo	24
5.6.3. trsrmt_GET_LOG	25
5.6.4. trsrmt_Get_LogInfo	25
5.6.5. trsrmt_Get_LogContents	26
5.6.6. trsrmt_GET_STATUS	27
5.6.7. trsrmt_RESET	27
5.6.8. trsrmt_EXIT	28
5.6.9. trsrmt_DO	28
5.6.10. trsrmt_FILE_OPEN	28
5.6.11. trsrmt_SCRIPT_Ctrl	29
5.6.12. trsrmt_SCRIPT_SOURCE	29
5.6.13. trsrmt_DEV_GET_EVENT	30
5.6.14. trsrmt_Get_EventTime	31
5.6.15. trsrmt_Get_EventWaveInfo	32
5.6.16. trsrmt_Get_EventNumFields	32
5.6.17. trsrmt_Get_EventFieldName	33
5.6.18. trsrmt_Get_EventFieldValue	33
5.6.19. trsrmt_DEV_GET_WAVEINFO	34
5.6.20. trsrmt_DEV_GET_WAVEBLOCK	35
5.6.21. trsrmt_Get_WaveformData	35
5.7. Revision History	37

1. Introduction

The TRsSIM software is able to be controlled remotely over a TCP/IP connection by means of a simple query language (please see the TRSSIM documentations for a complete specification). This document outlines two different Application Programming Interfaces (APIs) which have been designed to abstract the details of this TCP/IP query language into an easy to use programming interface. These APIs include:

- A C++ Dynamic Link Library for use with C compatible programming environments
- A .Net assembly for the Microsoft .Net development environment.

Each of these APIs is capable of:

- Connecting to and disconnecting from the TRSSIM software
- Returning device information for devices which are connected to TRSSIM
- Executing script programs and/or statements in TRSSIM
- Reading script log contents
- Reading event information
- Downloading waveform information

1.1. Files Included

The entire TRSSIM remote control package is distributed as a .zip archive which can be downloaded from www.adventinstruments.com. The files within this archive are organized into the following directories

- **Help** - this directory contains a copy of this file
- **DLL** – this directory contains all files related to the C++ DLL.
- **dotNet** – this directory contains all files related to the .Net assembly

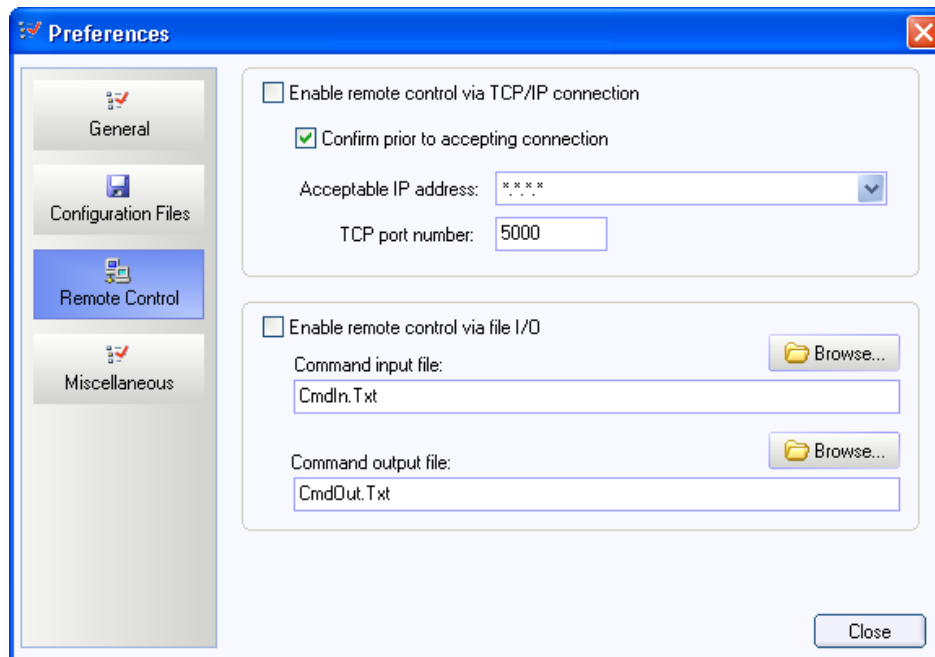
2. Known Limitations

At the time of authoring this document, the TRsSIM Remote Control has the following known issues:

1. If a remote control command causes a modal dialog box to be displayed (such as a message box), communications are halted and all communications will timeout until the dialog box is closed. For this reason it is unwise to remotely load scripts which use message boxes or intentionally cause modal dialog boxes to be displayed.

3. Establishing a Connection to TRsSim

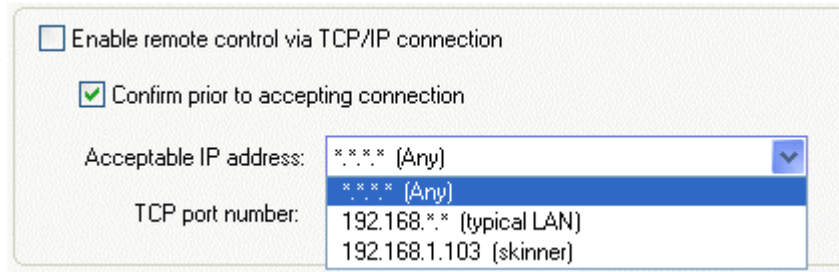
By default the TRsSim software does not accept any TCP/IP connections regardless of the source IP address or port. In order to enable a connection, the default TCP/IP settings must be changed. To view the current settings, select the "**Options**" command from the "**Configuration**" menu. Once the window is visible, click the "**TCP/IP**" tab. The following figure shows the default settings



The top most check box enables or disables TCP/IP connections. Unless it is checked, connections from other applications are not allowed. The next setting determines if a confirmation message is displayed before allowing a TCP/IP connection. If checked (default), the user must click a "Yes" button after the confirmation message is displayed. If "No" is clicked, then the TCP/IP connection is refused.

The next two settings are used to control the acceptable IP address of the controlling device, and the TCP port number.

By default, any IP address is allowed as the controlling device. The drop down list box includes two other settings. The second selection represents common LAN IP addresses, while the third selection is the IP address of the PC running the TRsSim software. If available, the PC's "name" is displayed in brackets following the IP address. In the example figure below, the PC called "wiggins" is located at address 192.168.1.106.



Any IP address can be entered into the above text box. Use the '*' character to represent a wildcard value.

The default TCP Port used is 5000. However this value can be changed if conflicts occur with other applications. The maximum allowable port value is 65535. Normally port values should be above 1024 to avoid conflicts with common applications.

All of the above settings are saved when the TRsSim program shuts down, and reloaded when started again. Loading configuration files or selecting the "Restore Default Settings" menu command has no effect on the TCP/IP settings.

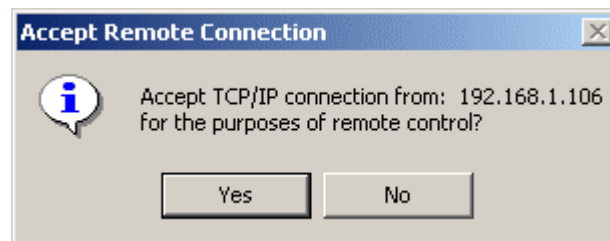
If TCP/IP connections are enabled then the lower right corner of the TRsSim software displays a small icon. The icon indicates that it is listening on the specified TCP port for a controlling application to initiate a connection. It is important to note that only a single connection is accepted by the TRsSim software. Once a connection is established, requests by other devices for a connection will be refused.



The following conditions must be met, before a connection can be established.

- TRsSim software is running.
- TRsSim software settings have enabled TCP/IP connections.
- No connection is currently established with another program.
- The acceptable IP address setting matches the IP address in the ServerIP) argument to trsmt_Connect .
- The port values match in both the TRsSim software and the port parameter of trsmt_Connect.

Calling trsmt_Connect in the DLL results in a connection attempt. If the TRsSim software is configured to require a confirmation of acceptance, then the following message is displayed.



Choosing the "Yes" button allows the connection, while the "No" button prevents a connection. Assuming the "Yes" button is clicked, a TCP/IP connection is then established between the demonstration and TRsSim program. The icon displayed in the lower right corner of the TRsSim software changes to reflect the connection. If the mouse is moved over the icon, the IP address of the remote computer is displayed.



4. TrsRmt .Net Assembly Reference

4.1. What's Included

All Files related to the TRSSIM remote control .Net assembly are located within the **dotNet subdirectory** of the .zip archive. Within this directory you will find the following:

- **TrsRmtdotNet** – This folder contains the project file, source code, and compiled assembly for the C# .Net TRSSIM remote control assembly
- **TrsRmt CSharp Example** – This folder contains an example program (written in C#) which demonstrates the proper usage of the .Net assembly
- **TrsRmt dotNet Code.sln** – This is a Microsoft Visual Studio 2008 solution file which contains both of the above projects.

4.2. System Requirements

This assembly requires the following:

- Microsoft .Net Framework 3.5
- Microsoft Visual Studio 2005/2008

4.3. Namespace

The features of the TRSSIM remote control can be accessed through the TrsRmt object found in the Advent.TrsRmt namespace. This document assumes the following statements:

[VB]

```
Imports Advent.TrsRmt
```

[C#]

```
using Advent.TrsRmt;
```

4.4. TrsRmt Object

The features of the TRSSIM remote control can all be accessed through the TrsRmt object. A very simplistic C# program is shown below which connects to TRSSIM executing on the same computer and then disconnects.

```
// Create a new remote control object
TrsRmt Remote = new TrsRmt();

// connect to TRSSIM using TCP/IP
Remote.Connect("127.0.0.1", 5000);

//TODO: access remote control here!

// disconnect from TRSSIM
Remote.Disconnect();
```

Once a connection is established then the TRSSIM features which can be controlled remotely are accessible through the members of the TrsRmt object which are documented in the following sections

4.4.1. Connect

Description:

This method attempts to connect to TRSSIM over TCP/IP

C# Prototype:

```
public void Connect(string ServerIP, int Port)
```

Return Value: None

Exceptions: If the connection fails, an exception will be thrown.

Parameters:

<i>Server IP</i>	Dotted decimal IP address (ie 192.168.1.102)
<i>Port</i>	TCP/IP Port number used to Connect to TRSSIM (note: TRSSIM must be configured to accept connections on this port)

4.4.2. Disconnect

Description:

This disconnects from TRSSIM if connected.

C# Prototype:

```
public void Disconnect()
```

Return Value: n/a

Exceptions: No exceptions should

Parameters: n/a

4.4.3. IsConnected

Description:

This returns true if a connection to TRSSIM has been established

C# Prototype:

```
public bool IsConnected()
```

Return Value:

True if a connection to TRSSIM has been established, False otherwise.

Exceptions: No exceptions should be raised

Parameters: n/a

4.4.4. SetTimeout

Description:

This sets the maximum time to wait for responses from TRSSIM.

C# Prototype:

```
public void SetTimeout(int TimeoutSeconds)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

TimeoutSeconds Timeout value in seconds

4.4.5. OpenFile

Description:

This instructs TRSSIM to open the specified file

C# Prototype:

```
public void OpenFile(string FileName)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

<i>Filename</i>	Complete path and filename of the file to open in TRSSIM
-----------------	--

4.4.6. Reset

Description:

This instructs TRSSIM to reset all devices and settings to their default values. This may take a long time – be sure to set a relatively long timeout.

C# Prototype:

```
public void Reset()
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters: n/a

4.4.7. Exit

Description:

This instructs TRSSIM to shutdown. Note: after this call TRSSIM will have to be re-started in order to re-connect

C# Prototype:

```
public void Exit()
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters: n/a

4.4.8. GetDevices

Description:

This returns a list of devices which are currently configured in TRSSIM

C# Prototype:

```
public System.Collections.Generic.List<DeviceInfo> GetDevices()
```

Return Value: List of DeviceInfo objects

Exceptions: Exceptions will be raised on failure.

Parameters: n/a

4.4.8.1. DeviceInfo Object

DeviceInfo objects are returned by GetDevices and represent devices currently configured in TRSSIM.

Each object has the following members:

- **DeviceID** – this is a char which uniquely identifies this device in TRSSIM and is used in subsequent device specific calls
- **Name** –name of the device (ie “AI-5120 Telephone Line Monitor”)
- **SerialNumber** - serial number of the device (ie “SN110099”)
- **SoftwareVersion** - software version of the device (ie “4.01”)
- **HardwareVersion** - hardware version of the device (ie “2.00”)

4.4.9. GetLogs

Description:

This returns the contents of one or more script output logs from TRSSIM

C# Prototype:

```
public System.Collections.Generic.List<Log> GetLogs(string LogName)
```

Return Value: List of Log objects

Exceptions: Exceptions will be raised on failure.

Parameters:

<i>LogName</i>	If not specified (zero length or null) then all available logs will be returned, otherwise only the log with the matching name will be returned from TRSSIM
----------------	---

4.4.9.1. Log Object

The Log object is returned by the GetLogs function and contains the contents of a script output log returned by TRSSIM.

This object has the following members:

- **Name** - this returns the name of the log as a string
- **Contents**- this returns the contents of the log as a string

4.4.10. SetScriptSource**Description:**

This sets the contents of the script editor to the string value specified. TRSSIM will compile this script and return errors (which will be raised as exceptions) if the script is not valid.

C# Prototype:

```
public void SetScriptSource(string Script)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure. This may include compilation errors.

Parameters:

<i>Script</i>	This must contain a valid script program which adheres to all the TRSSIM scripting syntax. Please see the TRSSIM documentation for details.
---------------	---

4.4.11. SendScriptCommand**Description:**

This sends a command which controls the execution of the TRSSIM script.

C# Prototype:

```
public void SendScriptCommand(Advent.TrsRmt.ScriptCommand  
Command)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

Command This enumerated value which specifies which action to perform
RUN – Start execution of the current TRSSIM script
END – terminate the execution of the current TRSSIM script
PAUSE – pause the current execution of the TRSSIM script
RESUME – resume the TRSSIM script (if paused)

4.4.12. DoStatements**Description:**

This method executes a set of script statements separately from the main TRSSIM script.

C# Prototype:

```
public void DoStatements(string ScriptStatements)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure. This may include compilation errors.

Parameters:

ScriptStatements This must contain one or more TRSSIM script statements.

4.4.13. GetStatus**Description:**

This returns the status of the TRSSIM script and any DO statements.

C# Prototype:

```
public void GetStatus(ref Advent.TrsRmt.ScriptState ScriptStatus,  
ref Advent.TrsRmt.ScriptState DoStatus) Command)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

ScriptStatus This enumerated value which specifies which returns the status of the TRSSIM script. It can be one of the following values

STOP – no script is currently executing
RUN – script is executing
PAUSE – script is paused
IDLE – script is waiting for events
ERROR – script encountered an error

DoStatus This returned a value which specifies the status of DO statements executing within TRSSIM. The values are the same as the

ScriptStatus.

4.4.14. GetEvent

Description:

This returns event information from a device connected in TRSSIM

C# Prototype:

```
public Advent.TrsRmt.Event GetEvent(int DeviceId,
Advent.TrsRmt.EventSelector Selector)
```

Return Value: Event object containing valid event information or null if no such event exists. See section 4.5 for information on this object.

Exceptions: Exceptions will be raised on failure.

Parameters:

<i>DeviceID</i>	This is the DeviceID value for the TRSSIM device which supports events. This is the ASCII char value of the TRSSIM device letter (A,B,C,D). This value can also be extracted through the DeviceID property of a DeviceInfo object returned from GetDevices
<i>Selector</i>	This specifies the event to return CURRENT – the currently selected event will be returned FIRST - the first event will be selected and returned LAST - the last event will be selected and returned NEXT - the next event will be selected and returned PREV - the previous event will be selected an returned.

4.4.15. GetWaveformInfo

Description:

This returns information about a waveform which has been recorded in TRSSIM

C# Prototype:

```
public void GetWaveformInfo(int DeviceID, int WaveformRefNum, ref
Advent.TrsRmt.WaveformInfo Info)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

<i>DeviceID</i>	This is the DeviceID value for the TRSSIM device which supports
-----------------	---

	events. This is the ASCII char value of the TRSSIM device letter (A,B,C,D). This value can also be extracted through the DeviceID property of a DeviceInfo object returned from GetDevices
<i>WaveformRefnum</i>	This specifies the reference number of the waveform being requested.
<i>Info</i>	This returns a waveform information structure which reports information about the waveform

4.4.15.1. *WaveformInfo Structure*

The WaveformInfo structure describes a waveform recorded in TRSSIM. It has the following members:

- **DeviceID**- this reports the device identifier to which this waveform is associated.
- **WaveformRefNum** - this returns the waveform reference number associated with this waveform information.
- **StartTime** – this returns the device time stamp associated with the first sample in this waveform (in seconds).
- **NumSamples** – this returns the number of samples in this waveform
- **BitsPerSample** – this returns the resolution of the samples in bits/sample
- **SampleRate** – this returns the sample rate of this waveform
- **ScaleFact** – this returns the scaling factor which is needed to convert the native integer samples to “real” values (in native units)

4.4.16. GetWaveform

Description:

This downloads the specified waveform from TRSSIM

C# Prototype:

```
public void GetWaveform(int DeviceID, int WaveformRefnum, ref
    Advent.TrsRmt.Waveform Wave)
```

Return Value: n/a

Exceptions: Exceptions will be raised on failure.

Parameters:

DeviceID This is the DeviceID value for the TRSSIM device which supports events. This is the ASCII char value of the TRSSIM device letter (A,B,C,D). This value can also be extracted through the DeviceID property of a DeviceInfo object returned from GetDevices

WaveformRefnum This specifies the reference number of the waveform being

	requested.
<i>Wave</i>	Structure which will return the waveform information and the downloaded samples.

4.5. Event Object

The Event object is returned by GetEvent and contains information about an event recorded in TRSSIM for a particular device. The Event object has the following members:

- **Name** – this specifies the name of the event “ie. DTMF Digit”
- **Time** – this specifies the device timestamp of the start of the event (in seconds)
- **PCTime** – this specifies the timestamp of the event according to the clock on the computer which TRSSIM is running
- **EventType** – this returns an integer value which specifies the event type. This value is device specific. Please see TRSSIM documentation for details.
- **ACWaveRefNum** – if non-zero then this identifies an AC waveform which was recorded at a time consistent with this event.
- **DCWaveRefnum** - if non-zero then this identifies the DC waveform which was recorded at a time consistent with this event.
- **Fields** – this returns a list of fields (in Name/Value format) which details measurement values associated with the event.

4.6. Revision History

Revision 1.0

- First official release

5. TrsRmt 'C' DLL Reference

5.1. Files Included

All Files related to the TRSSIM remote control DLL are located within the **DLL subdirectory** of the .zip archive. Within this directory you will find the following:

- **Trsrmt.dll**– This is the windows dynamic link library which contains the remote control API
- **Trsrmt.lib** - this is the Microsoft object library file which is required to import the DLL definitions into development environments
- **Trsrmt.h** – this is a C header file which declares the API functions in trsrmt.dll
- **Trsrmt.bas** - this is a VB6 module which declares the API functions in trsmrmt.dll

5.2. Coding Conventions

All of the functions in the TRsSIM Remote Control DLL abstract the TRsSIM Remote Control Commands. In general, each function does one of the following:

- Manages the TCP/IP communications
- Issues a command to TRsSIM and optionally parses the response.
- Extracts information from a response to a command.

All of the functions in the DLL follow the conventions listed below:

1. Unless noted otherwise, all functions return a standard error code which is set to one of the values in the `trsrmt_err_e` enumerated type. Functions that complete normally will return `terr_None` (zero). To programmatically extract a text description of an error code, the DLL includes the function `trsrmt_GetFuncErrorMsg` which returns a NULL terminated string describing the error message.
2. Functions that issue commands to TRsSIM over a TCP/IP connection have the name of the command capitalized in the function name (ie. `trsrmt_GET_DEVICES` issues the GET DEVICES command).
3. Functions that issue commands to TRsSIM hold the response in memory after the function completes so other related functions may extract this information (ie. `trsrmt_Get_DeviceInfo` can be called after `trsrmt_GET_DEVICES` to get specific device information).
4. Only one command response is held in memory at any given time.
5. In general, functions that extract information from the response to a command contain a phrase (not in all upper case) that relates to the information it extracts (ie. `trsrmt_Get_LogContents` gets the contents of a log that was returned from `trsrmt_GET_LOG`).
6. All parameters named `MaxLen` must be set to length of a character buffer parameter (in bytes). This is used to insure that memory copies don't accidentally overflow buffers.
7. All function parameters named `CharsCopied` generally return the number of characters copied into a character buffer parameter. The value returned does not include the NULL terminator character. (ie if "Hello" was copied then `CharsCopied` would be set to 5. The data copied would be `{'H','e','l','l','o',NULL}`)

5.3. System and Error Handling Functions

5.3.1. trsrmt_Get_Version

Description:

This function returns the version information of the software.

Function Prototype:

```
void trsrmt_Get_Version(long *Major, long *Minor)
```

Return Value: None

Function Parameters:

Major This will return the major revision number of the DLL

Minor This will return the minor revision number of the DLL

5.3.2. trsrmt_Get_FuncErrorMsg

Description:

This function returns a text description of an error code returned from a function call.

Function Prototype:

```
void trsrmt_Get_FuncErrorMsg(long ErrCode, char *ErrMsg, long MaxLen,  
long *CharsCopied)
```

Return Value: None

Function Parameters:

ErrCode The should be set to the error code value returned by a DLL function

ErrMsg This should point to a character buffer where the error message can be stored

MaxLen This should be set to the length of the ErrMsg buffer in bytes

CharsCopied When the function completes this will be set to the number of characters (not including NULL) that were copied into ErrMsg

5.3.3. trsrmt_Get_ERROR_Msg

Description:

If TRsSIM returns an ERROR response to a command, the function that issued the command will return a value of `terr_ERROR_Resp`. When this occurs, the contents of the error message can be read using the `trsmt_Get_ERROR_Msg` function. If an error message is not present then this function copies zero bytes.

Function Prototype:

```
long trsmt_Get_ERROR_Msg(char *ErrMsg, long MaxLen, long
*CharsCopied);
```

Return Value: None**Function Parameters:**

<i>ErrMsg</i>	This parameter should point to a buffer where the error message will be stored.
<i>MaxLen</i>	This must be set to the size of the buffer pointed to by ErrMsg
<i>CharsCopied</i>	When the function completes this will be set to the number of characters (not including NULL) that were copied into ErrMSg

5.3.4. trsmt_Get_WARNING_Msg

Description:

If TRsSIM returns an WARNING response to a command, the function that issued the command will return a value of `terr_WARNING_Resp`. When this occurs, the contents of the error message can be read using the `trsmt_Get_WARNING_Msg` function. If a warning message is not present then this function copies zero bytes.

Function Prototype:

```
long trsmt_Get_WARNING_Msg(char *WarnMsg, long MaxLen, long
*CharsCopied);
```

Return Value: None**Function Parameters:**

<i>WarnMsg</i>	This parameter should point to a buffer where the warning message will be stored.
<i>MaxLen</i>	This must be set to the size of the buffer pointed to by ErrMsg
<i>CharsCopied</i>	When the function completes this will be set to the number of characters (not including NULL) that were copied into ErrMSg

5.4. TCP/IP Connection Functions

5.4.1. trsrmt_Connect

Description:

This function attempts to establish a connection to a TRsSIM server at a given IP address and port number.

Function Prototype:

```
long trsrmt_Connect(char *ServerIP, long Port)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

ServerIP This should point to a NULL terminated string which contains the IP address of the PC running the TRsSIM software. The string should be formatted in the “dotted-decimal” notation. (ie. “192.168.1.1”)

Port The port number specifies which port to connect to on the server. This number must match the opened port number in the TRsSIM software in order to connect.

5.4.2. trsrmt_Disconnect

Description:

This function closes the current TCP/IP connection. If no connection is open, this function does nothing.

Return Value: None

Function Prototype:

```
void trsrmt_Disconnect()
```

5.4.3. trsrmt_Set_Timeout

Description:

This function sets the communication timeout (in seconds) for all TCP/IP communications. This sets the amount of time the software will wait for a response from TRsSIM.

Function Prototype:

```
long trsrmt_Set_Timeout(long Timeout)
```


Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Timeout</i>	This should be set to the number of seconds to wait for a response from the TRsSIM software. Since TCP/IP communications can sometimes involve long time delays, a modestly large timeout (such as 10 seconds) is recommended.
----------------	--

5.5. Low-Level Communication Functions

To allow the most flexibility for the user, the DLL exposes the following functions that allow the user to send TRsSIM remote control commands directly. **It is not recommended to mix these low-level function calls with other high-level functions.**

5.5.1. trsrmt_SendCmd

Description:

This function sends a command to TRsSIM over the TCP/IP connection. This function automatically encloses the data in CmdData with the necessary STX, and ETX characters before sending the data. This function doesn't wait for a response.

Function Prototype:

```
long trsrmt_SendCmd(char *CmdData, long CmdDataLen)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>CmdData</i>	This parameter should point to a buffer containing the command to send to the TRsSIM software. Important: This buffer MUST NOT contain the STX, ETX, or NULL characters.
<i>CmdDataLen</i>	This parameter should be set to the length (in bytes) of the command in CmdData.

5.5.2. trsrmt_WaitCmdResp

Description:

This function waits for a response from TRsSIM. A valid response from TRsSIM must be enclosed in STX/ETX characters and must arrive within the specified timeout. If a valid message is received then this function returns the length of the message in bytes.

Function Prototype:

```
long trsrmt_WaitCmdResp(long *ResponseLen)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>ResponseLen</i>	If a valid message is received from TRsSIM then this will be set to the length of the message received.
--------------------	---

5.5.3. trsrmt_Get_RespLen

Description:

This function returns the length of the last response from TRsSIM in bytes. If no response has been received, then this function returns 0.

Function Prototype:

```
long WINAPI trsrmt_Get_RespLen()
```

Return Value: Length of last response in bytes

Function Parameters: None

5.5.4. trsrmt_Get_Response

Description:

This function returns the contents of the last response from TRsSIM. If there is no response in memory then CharsCopied will return zero.

Function Prototype:

```
long trsrmt_Get_Response(char *Response, long MaxLen, long  
*CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Response</i>	This parameter should point to a buffer where the response can be stored. This buffer should be at least as long as the response (which can be read using trsrmt_Get_RespLen)
<i>MaxLen</i>	This parameter should be passed as the length of the Response buffer.
<i>CharsCopied</i>	This returns the number of characters copied into Response (not including NULL character)

5.6. General Remote Control Functions

5.6.1. trsrmt_GET_DEVICES

Description:

This function issues the GET DEVICES command to TRsSIM, then receives and parses the response, and returns the number of devices in the response.

Related Functions:

trsrmt_Get_DeviceInfo

Function Prototype:

```
long trsrmt_GET_DEVICES(long *NumDevices)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>NumDevices</i>	This parameter returns the number of devices detected in the response from TRsSIM.
-------------------	--

5.6.2. trsrmt_Get_DeviceInfo

Description:

This function extracts specific device information from the response to trsrmt_GET_DEVICES.

Function Prototype:

```
long trsrmt_Get_DeviceInfo(long Index, long *DeviceID, char *Info, long MaxLen, long *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Index</i>	This parameter specifies which set of device information to extract from the response. This value should range from 1 to NumDevices (as returned from trsrmt_GET_DEVICES)
<i>DeviceID</i>	This parameter returns the ASCII code of the device identifier in the Info string. This value can be used to reference this device in other device specific function calls.
<i>Info</i>	This parameter should point to a buffer where the information

	string for the device will be copied. This string follows the formatting specified in the TRsSIM Remote Control Command Reference.
<i>MaxLen</i>	This parameter should be set to the length of the Info buffer in bytes.
<i>CharsCopied</i>	This parameter will return the number of characters copied into the Info buffer (not including NULL character)

5.6.3. trsrmt_GET_LOG

Description:

This function issues the GET LOG command to TRsSIM (with an optional logname to get) and returns the number of logs detected in the response.

Related Functions:

trsrmt_Get_LogInfo
trsrmt_Get_LogContents

Function Prototype:

```
long trsrmt_GET_LOG(char *LogName, long *NumLogs)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>LogName</i>	This parameter specifies the name of a specific log to be returned. If you wish to return all logs then set this parameter to NULL or an empty string.
<i>NumLogs</i>	This parameter returns the number of logs detected in the response

5.6.4. trsrmt_Get_LogInfo

Description:

This function returns the name and content length for a specific log detected in the response to trsrmt_GET_LOG.

Function Prototype:

```
long trsrmt_Get_LogInfo(long Index, char *LogName, long MaxLen, long *LogSize, long *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Index</i>	This parameter specifies which log to extract the information from. This parameter should range from 1 to NumLogs (as returned from trsrmt_GET_LOG)
<i>LogName</i>	This parameter should point to a character buffer where the name of the log will be stored.
<i>MaxLen</i>	This parameter should be set to the length of the LogName buffer
<i>LogSize</i>	This parameter should will return the length of the log contents in bytes.
<i>CharsCopied</i>	This parameter will return the number of characters copied into the LogName buffer (not including NULL terminator)

5.6.5. trsrmt_Get_LogContents**Description:**

This function extract the contents of a specific log found in the response to trsrmt_GET_LOG.

Function Prototype:

```
long trsrmt_Get_LogContents(long Index, char *LogContents, long MaxLen,
long *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Index</i>	This parameter specifies which log to extract the information from. This parameter should range from 1 to NumLogs (as returned from trsrmt_GET_LOG)
<i>LogContents</i>	This parameter should point to a character buffer where the log contents will be stored. This buffer should be allocated to be at least LogSize bytes as returned from trsrmt_Get_LogInfo.
<i>MaxLen</i>	This parameter should be set to the length of the LogContents buffer.
<i>CharsCopied</i>	This parameter will return the number of characters copied into the LogContents buffer (not including NULL terminator)

5.6.6. trsrmt_GET_STATUS

Description:

This function sends the GET STATUS command and returns the status of the script and the status of script command issued with the DO command.

Function Prototype:

```
long trsrmt_GET_STATUS(long *ScriptStatus, long *DoStatus)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>ScriptStatus</i>	This parameter returns the status of the script as an enumerated value defined by trsrmt_scriptstate_e. (see definition below)
<i>DoStatus</i>	This parameter returns the status of script statements executed with the DO command as an enumerated value defined by trsrmt_scriptstate_e. (see definition below)

The script status will be returned as a value defined in the following enumerated type:

```
enum trsrmt_scriptstate_e {
    script_NOSTATUS= -1, /* no valid status found*/
    script_STOP=0,      /* no script program executing */
    script_RUN=1,        /* script is running */
    script_PAUSE  =2,    /* script is paused */
    script_IDLE=3,       /* script is waiting for events */
    script_ERROR=4,      /* error detected in script program */
};
```

5.6.7. trsrmt_RESET

Description:

This function resets TRsSIM to default settings.

Function Prototype:

```
long trsrmt_RESET(void)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters: None

5.6.8. trsrmt_EXIT

Description:

This function exits the TRsSIM software. After this function call completes successfully, TRsSIM will shutdown the TCP/IP link and any further communications will fail. In most programs, this function call should be immediately followed by trsrmt_Disconnect.

Function Prototype:

```
long trsrmt_EXIT(void)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters: None

5.6.9. trsrmt_DO

Description:

This function executes script statements in TRsSIM by issuing the DO command.

Function Prototype:

```
long trsrmt_DO(char *Script)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Script</i>	This parameter should point to a NULL terminated string containing the script statements to execute in TRsSIM.
---------------	--

5.6.10. trsrmt_FILE_OPEN

Description:

This function issues the FILE OPEN command, which causes TRsSIM to open the specified file.

Function Prototype:

```
long trsrmt_FILE_OPEN(char *Filename)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Filename</i>	This parameter should point to a NULL terminated string containing the name of the file to open.
-----------------	--

5.6.11. trsrmt_SCRIPT_Ctrl

Description:

This function issues the SCRIPT command to control the execution of the script in TRsSIM.

Function Prototype:

```
long trsrmt_SCRIPT_Ctrl(long Command)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

Command This parameter specifies the script action to perform. The parameter should be set to one of the values in the trsrmt_scriptcmd_e enumerated type.

The command parameter should be passed with a value contained in the enumerated type shown below:

```
enum trsrmt_scriptcmd_e {
    script_cmd_END=0,           /* Stop the script*/
    script_cmd_PAUSE=1,        /* Pause the script */
    script_cmd_RESUME =2,      /* Resume script execution*/
    script_cmd_RUN=3,          /* Run the script*/
};
```

5.6.12. trsrmt_SCRIPT_SOURCE

Description:

This function issues the SCRIPT SOURCE command, which loads a script program into the current script project; overwriting any existing script project.

Function Prototype:

```
long trsrmt_SCRIPT_SOURCE(char *Script)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

Script This parameter should point to a NULL terminated string containing a script program to load into TRsSIM.

5.6.13. trsrmt_DEV_GET_EVENT

Description:

This function issues the DEV <ident> GET EVENT <selector> command to TRsSIM to extract an event recorded for the specified device.

Related Functions:

trsrmt_Get_EventTime

trsrmt_Get_EventWaveInfo

trsrmt_Get_EventNumFields

trsrmt_Get_EventFieldName

trsrmt_Get_EventFieldValue

Function Prototype:

```
long trsrmt_DEV_GET_EVENT(long DeviceID, long EventSelector, long
    *EventIndex, long *EventType, char *EventName, long MaxLen, long
    *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>DeviceID</i>	This parameter is used to uniquely identify a device in TRsSIM. The value passed should be returned from trsrmt_Get_DeviceInfo.
<i>EventSelector</i>	This parameter is used to identify which event to select. The value of this parameter should be one of the values in the trsrmt_eventsel_e enumerated type. (see below)
<i>EventIndex</i>	This parameter will return an index value for the current event (as returned from TRsSIM). If a value of zero is returned this means that there is no event at the specified location.
<i>EventType</i>	This parameter returns a numerical value representing the event type. This possible values for the event type are documented in the TRsSIM documentation. "TRsSim - Script Properties & Functions" under Event.

<i>EventName</i>	This parameter should point to a buffer where the name of the event will be stored.
<i>MaxLen</i>	This parameter should be set to the length of the EventName buffer
<i>CharsCopied</i>	This parameter returns the number of characters copied into the EventName buffer (not including NULL terminator)

The EventSelector parameter should be passed with a value from the following enumerated type:

```
enum trsrmt_eventsel_e {
    ventssel_FIRST=0,           /* get the first event */
    ventssel_LAST  =1,         /* get the last event */
    ventssel_NEXT  =2,         /* get the next event */
    ventssel_PREV  =3,         /* get the previous event */
    ventssel_CURRENT=4,        /* get the current event */
};
```

5.6.14. trsrmt_Get_EventTime

Description:

This function extracts the event timing from the response to the trsrmt_DEV_GET_EVENT function.

Function Prototype:

```
long trsrmt_Get_EventTime(long *Year, long *Month, long *Day, long
    *Hour, long *Min, long *Sec, double *TimeFromStart)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Year</i>	This parameter returns the year when the PC recorded the event.
<i>Month</i>	This parameter returns the month when the PC recorded the event.
<i>Day</i>	This parameter returns the day when the PC recorded the event.
<i>Hour</i>	This parameter returns the hour when the PC recorded the event.
<i>Min</i>	This parameter returns the minute when the PC recorded the event.
<i>Sec</i>	This parameter returns the second when the PC recorded the event.

<i>TimeFromStart</i>	This parameter returns the timing of the event relative to the start of the TRsSIM software (in seconds)
----------------------	--

5.6.15. trsrmt_Get_EventWaveInfo

Description:

This function extracts the waveform reference numbers from the response to the trsrmt_DEV_GET_EVENT function. These waveform reference numbers can then be used to download the waveform.

Function Prototype:

```
long trsrmt_Get_EventWaveInfo(long *ACWaveRefNum, long
*DCWaveRefNum)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

ACWaveRefNum This parameter returns a reference number corresponding to the AC capture which contains this event. If this value is zero, this means there is no waveform available.

DCWaveRefNum This parameter returns a reference number corresponding to a DC capture, which contains this event. If this value is zero, this means that there is no waveform available.

5.6.16. trsrmt_Get_EventNumFields

Description:

This function returns the number of event information fields in the response from a call to trsrmt_DEV_GET_EVENT. These event information fields are specific to each event type and are documented in the TRsSIM documentation “TRsSim - Script Properties & Functions” under “Event”.

Function Prototype:

```
long trsrmt_Get_EventNumFields(long *NumFields)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>NumFields</i>	Returns the number of event information fields in the response.
------------------	---

5.6.17. trsrmt_Get_EventFieldName**Description:**

This function returns the field name portion of an event information field from the response to the trsrmt_DEV_GET_EVENT. These event information fields are specific to each event type and are documented in the TRsSIM documentation "TRsSim - Script Properties & Functions" under "Event".

Each event information field is formatted:

<field name>=<field value><CRLF>

where field name is generally all upper case and the field value is a specially formatted string. For example if the event type is OSI (6) then there may be a field
DURATION=(time):(units)

Function Prototype:

```
long trsrmt_Get_EventFieldName(long Index, char *FieldName, long
MaxLen, long *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>Index</i>	This parameter specifies which event information field name to return. This value should range from 1 to NumFields (as returned by trsrmt_Get_EventNumFields).
<i>FieldName</i>	This parameter should point to a character buffer where the field name will be stored
<i>MaxLen</i>	This parameter should be set to the length (in bytes) of the FieldName buffer.
<i>CharsCopied</i>	This parameter returns the total number of characters copied into the FieldName buffer (not including NULL)

5.6.18. trsrmt_Get_EventFieldValue

Description:

This function is used to extract event information fields from the response to the trsrmt_DEV_GET_EVENT function. These event information fields are specific to each event type and are documented in the TRsSIM documentation “TRsSim - Script Properties & Functions” under “Event”.

Each event information field is formatted:

<field name>=<field value><CRLF>

where field name is generally all upper case and the field value is a specially formatted string. For example if the event type is OSI (6) then there may be a field

Function Prototype:

```
long trsrmt_Get_EventFieldValue(char *FieldName, char *FieldValue, long
MaxLen, long *CharsCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>FieldName</i>	This is the field name to search for in the event information.
<i>FieldValue</i>	This parameter should point to a character buffer used to store the field value if found.
<i>MaxLen</i>	This parameter should be set to the length of the FieldValue buffer (in bytes)
<i>CharsCopied</i>	This parameter will return the number of characters copied into the FieldValue buffer (not including the NULL terminator)

5.6.19. trsrmt_DEV_GET_WAVEINFO

Description:

This function issues the DEV <deviceID> GET WAVEINFO command and returns the waveform information found in the response.

Function Prototype:

```
long trsrmt_DEV_GET_WAVEINFO(long DeviceID, long RefNum, double
*Starttime, long *NumSamples, double *SampleRate, double *ScaleFact, long
*MaxBlockSize, long *BitsPerSample, long *NumBlocks)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>DeviceID</i>	This parameter is used to identify which device this command
-----------------	--

	applies to. This value should be returned from the trsrmt_Get_DeviceInfo function.
<i>RefNum</i>	This parameter is used to identify the waveform to extract information from. Values for the waveform reference number should be returned from trsrmt_Get_EventWaveInfo.
<i>StartTime</i>	This parameter returns the time reference for the start of the waveform relative to the start of the TRsSIM program (in seconds).
<i>NumSamples</i>	This parameter returns the number of samples in the waveform
<i>SampleRate</i>	This parameter returns the sample rate of the waveform (in Hz)
<i>ScaleFact</i>	This parameter returns a value which can be used to convert the waveform from integer samples to units of volts.
<i>MaxBlockSize</i>	This parameter returns the maximum size of a waveform block (useful if using the trsrmt_DEV_GET_WAVEBLOCK function)
<i>BitsPerSample</i>	This specifies the resolution of each sample in bits.
<i>NumBlocks</i>	This specifies the number of blocks required to transfer the entire waveform.

5.6.20. trsrmt_DEV_GET_WAVEBLOCK

Description:

This function issues the DEV <deviceID> GET WAVEBLOCK <ref> <blocknum> and returns the waveform information as an array of signed 16 bit integer samples.

NOTE: This function should only be used if the application requires it! We recommend using trsrmt_Get_WaveformData which downloads an entire waveform.

Function Prototype:

```
long trsrmt_DEV_GET_WAVEBLOCK(long DeviceID, long RefNum, long
BlockNum, __int16 *SampleData, long MaxSamples, long *SamplesCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>DeviceID</i>	This parameter is used to specify which device this command applies to. This value should be returned from the trsrmt_Get_DeviceInfo function.
<i>RefNum</i>	This parameter specifies the waveform reference number of the waveform.
<i>BlockNum</i>	This parameter specifies which block to download from the waveform. This value should range from 1 to NumBlocks (as

	returned from trsrmt_DEV_GET_WAVEINFO)
<i>SampleData</i>	This parameter should point to an array of 16 bit signed integers where the block of samples will be stored. This buffer should be at least MaxBlockSize samples long (as returned from trsrmt_DEV_GET_WAVEINFO)
<i>MaxSamples</i>	This parameter should be set to the number of 16 bit integer samples in the SampleData array.
<i>SamplesCopied</i>	This parameters returns the number of 16 bit samples copied into the SampleData array.

5.6.21. trsrmt_Get_WaveformData

Description:

This function downloads an entire waveform from TRsSIM and returns the data as an array of 16 bit signed integer samples. To accomplish this, this function makes calls to trsrmt_DEV_GET_WAVEINFO and trsrmt_DEV_GET_WAVEBLOCK

Related Functions:

trsrmt_DEV_GET_WAVEINFO

Function Prototype:

```
long trsrmt_Get_WaveformData(long DeviceID, long RefNum, __int16
*SampleData, long MaxSamples, long *SamplesCopied)
```

Return Value: DLL Error Code (see section on error codes)

Function Parameters:

<i>DeviceID</i>	This parameter specifies which device this command applies to. This value should be returned from the trsrmt_Get_DeviceInfo function.
<i>RefNum</i>	This parameter should be set to the reference number of the waveform to download.
<i>SampleData</i>	This parameter should point to an array of signed 16 bit integers where the waveform will be stored. This array should be at least NumSamples long (as returned from trsrmt_DEV_GET_WAVEINFO)
<i>MaxSamples</i>	This parameter should be set to the number of 16 bit samples in the SampleData array.
<i>SamplesCopied</i>	This parameter returns the number of samples copied into the SampleData array

5.7. Revision History

Version 1.0

- Initial Release

Version 1.2

- Removed compiler optimization which caused problems in certain development environments

Version 1.3

- Fixed error in trsrmt_Get_EventFieldName and underlying functions which caused GPF faults.