Shanghai Geshe Information Technology Co., Ltd.

Geshe Debug Genius
User Manual

Version **3.2**

Contents

User Manual

1. Introduction

1.1 About

Geshe Debug Genius is a powerful communication protocol debugging and testing software. It can reduce the requirement of debugging and testing software customization in the electronic research and development process dramatically. It can quickly customize any communication protocol and cope with the fast and changeable communication test environment. It can make debugging and testing tools of enterprise product research and development be unified and standardized, and can help companies reduce development costs, learning costs and maintenance costs of testing tool!

Geshe Debug Genius's Advantage:

- Quick test drive customization - Entry-level skills demand, professional level of work results
- Intuitive test data presentation - Powerful data classification, storage, statistics and display functions
- Flexible test flow control - Support cyclic test and combined test, support active device and the slave device simulation, support simplex and duplex mode, support direct incentive and incentive protocol testing alone or mixed test

Geshe Debug Genius's Applicable Scope:

- The electronic product development, testing and production enterprises
- The electronic product development, testing engineers

1.2 Functional Characteristics

1) Basic Functions
   1.1) Support receiving and displaying data in text or hex mode;
   1.2) Support COM interface and automatically search serial ports that system supports, support 150~256000 baud rate, support custom baud rate;
   1.3) Support Network interface, support UDP, TCP Client, TCP Server protocol modes;
   1.4) Support automatic saving test data.

2) Direct Excitation Functions
   2.1) Support hex, string and file format data transfer;
   2.2) Support cyclic excitation;
   2.3) Support preservation of documents and can be easy to manage and reuse test projects.

3) Protocol Excitation Functions
   3.1) Support excitation simulation of master device and slave device;
   3.2) Support customization for any protocol frame format, the minimum resolution unit is 1bit, arbitrary frame format can be visually displayed;
   3.3) Support custom protocol and test result classification storage and display;

3.4）Support using C#、VB and Jscript language to control test, calling the third part DLL to finish complex computation and test task;
3.5）Support cyclic excitation and excitation report;
3.6）Support preservation of documents and can be easy to manage and reuse test projects.

4）Regular Tools
4.1）Support CS, BCC, LRC checksum calculator;
4.2）Support DES/3DES calculator, support 64 bits、128 bits and 196 bits keys;
4.3）Support CRC8/CRC16/CRC32 calculator, support custom parameters, support hex, string and file format and CRC standard algorithm table;
4.4）Support MD5/SHA1/SHA256/SHA384/SHA512 hash calculator，support hex, string and file format;
4.5）Support ASCII code checklist;
4.6）Support Character encoding（ASCII、UTF8、UNICODE 等）and Hex converter.

1.3 System Requirement

Supported Operation System:

- Windows 7 SP1 (x86 和 x64) and Above

Computer Hardware Requirement:

- The recommended minimum requirements: Pentium 1 GHz or faster, 1 GB RAM or larger
- The minimum disk space: 300 MB

Essential Components:

- Microsoft .NET Framework 4.8

1.4 Licensing and Purchase

1.4.1 Copyright

1.4.2 Purchase

Registration Advantage:

- Authorized to use the software in the business environment
- Get advanced functions of protocol excitation
- Get free support and help

How to purchase?

Way 1: Contact with our representatives

- Email: sales@geshe.com
- Phone: 0086-21-52194366

1.5 Support

You can contact us through our website, e-mail, etc. to get the support information when you meet with problems.

- Web Site: www.geshe.com
- Email: support@geshe.com
- QQ: 979464

2. Software Introduction

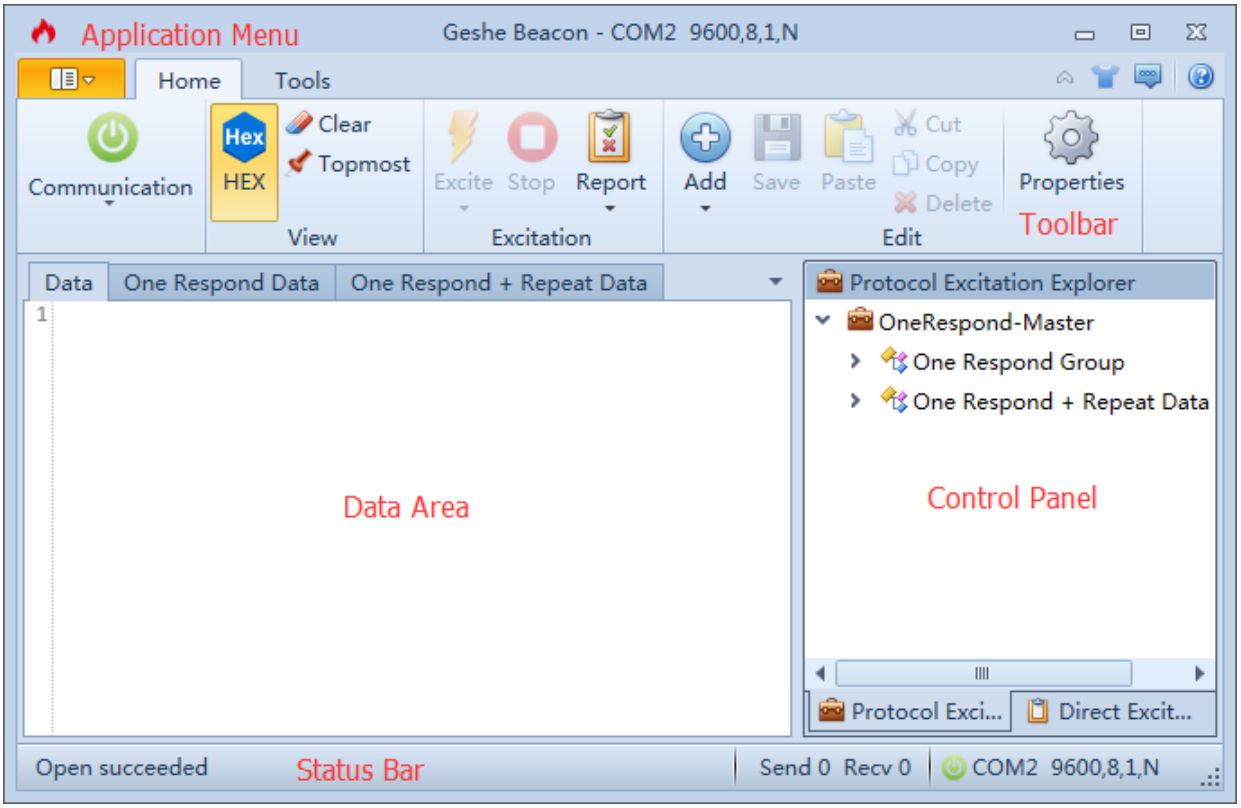
2.1 Startup Software

When installed successfully, software menu will be created in Windows Start Menu. Direct Excitation Project File with .bsp extension and Protocol Excitation Project File with .bcp extension will be attached to the windows system. There are two methods to start the software:

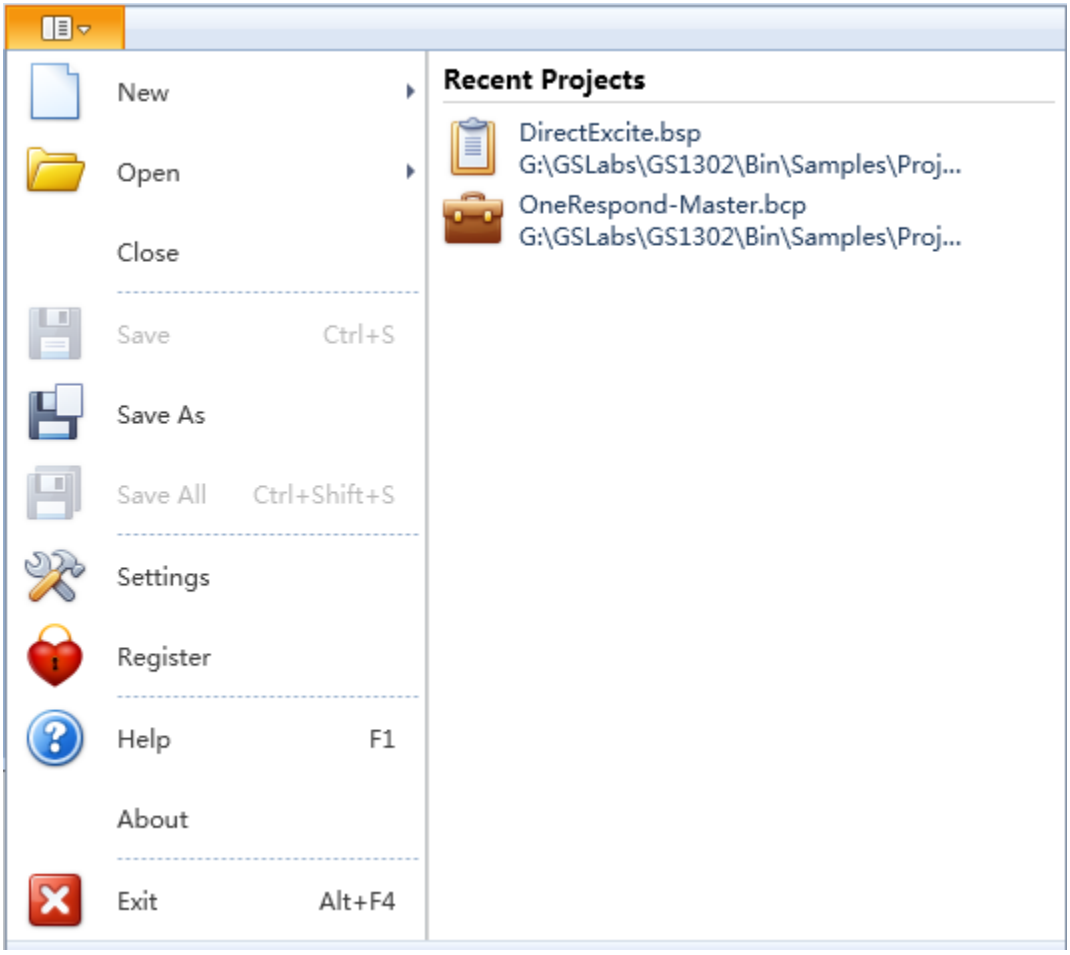
- Method 1: Windows【Start Menu】->Programs-->Geshe Debug Genius-->Geshe Debug Genius.
- Method 2: Use mouse to double click .bsp or .bcp file.

2.2 User Interface

2.2.1 Main Window



2.2.2 Application Menu

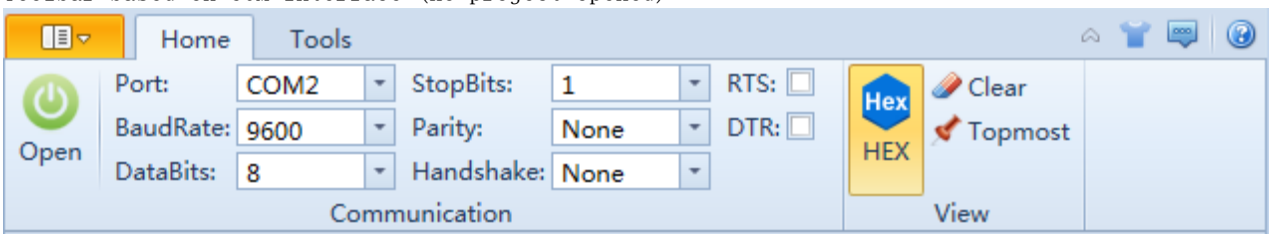


Command	Shortcut	Function
New-->Protocol Excitation Project	Ctrl+N	New a Protocol Excitation Project
New-->Direct Excitation Project	Ctrl+Shift+N	New a Direct Excitation Project
Open-->Protocol Excitation Project	Ctrl+O	Open a Protocol Excitation Project
Open-->Direct Excitation Project	Ctrl+Shift+O	Open a Direct Excitation Project
Close		Close current active project
Save	Ctrl+S	Save current active project
Save As		Save current active project to another file
Save All	Ctrl+Shift+S	Save all opened projects
Settings		Set running parameters for the software
Register		Register the software and get more functions and services.

Help	F1	Show help
About		Show copyright, version and registration information.
Exit	Alt+F4	Exit the software

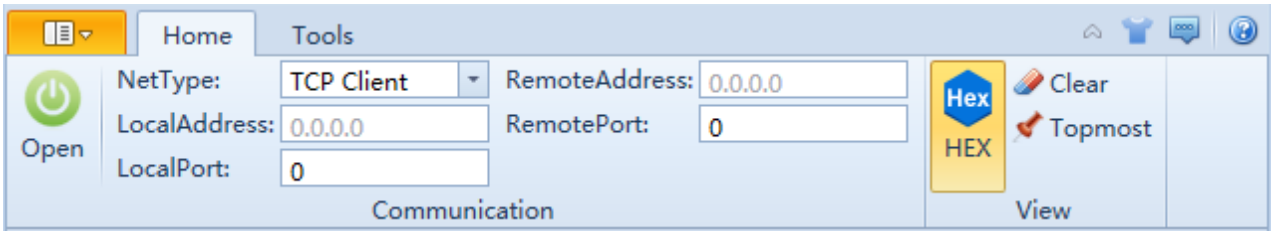
2.2.3 Toolbar

Toolbar based on COM interface (no project opened)



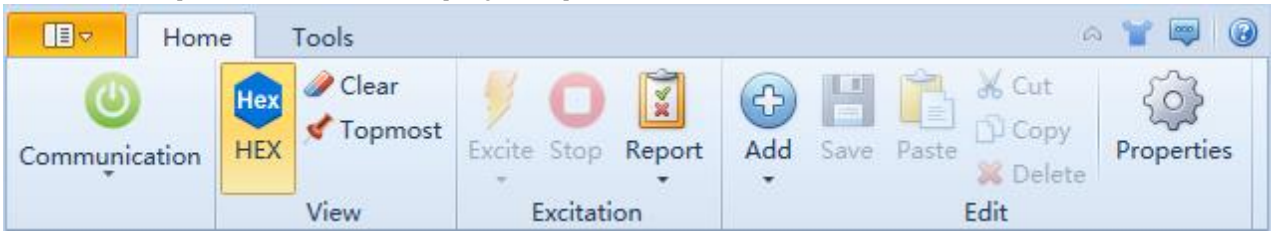
Command	Functions
Open/Close	Open or Close COM port
COM Port	Set COM Port, automatically check the system supported COM ports.
Baud Rate	Set Baud Rate, can customize baud rate by manual input.
Data Bits	Set Data Bits, support 5, 6, 7, 8.
Stop Bits	Set Stop Bits, support 1, 1.5, 2.
Parity	Set Parity, support None, Odd, Even, Mark and Space.
Flow Control	Set Flow Control, Support None, RequestToSend, XonXoff, RequestToSend /XonXoff.
RTS	Set RTS Signal
DTR	Set DTR Signal
HEX	Set Data display format to be Hex or String in Data panel.
Clear	Clear all data panels, not affect the data has been saved.
Topmost	Set the main window to be topmost.
Fold Toolbar（1st button On the right corner）	Display/Fold toolbar
Feedback（2st button On the right corner）	Show user feedback dialog.
Help （ 3st button On the right corner）	Show user manual.

Toolbar based on Network interface (no project opened)



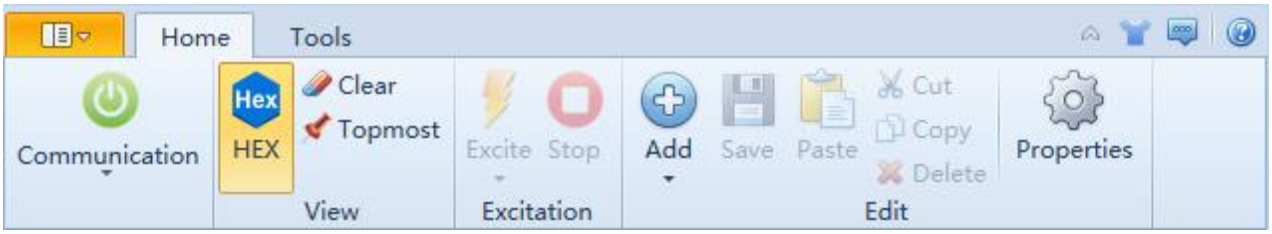
Command	Functions
Open/Close	Open or Close network interface
Protocol Type	Set Protocol Type, Support UDP、TCP Client and TCP Server.
Local Address	Set Local Address, default value is 0.0.0.0.
Local Port	Set Local Port. It is needed when use as UDP or TCP Server. When use as TCP Client, 0 indicates system default value.
Remote Address	Set Remote Address. It is needed when use as UDP or TCP Client and can't be set when use as TCP Server.
Remote Port	Set Remote Port. It is needed when use as UDP or TCP Client and can't be set when use as TCP Server.

Toolbar when protocol excitation project opened (communication interface folded)



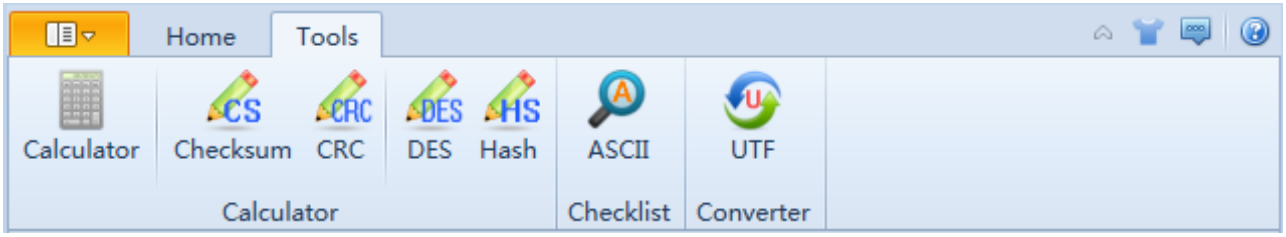
Command	Functions
Excite	Excite selected item or item suite once, and excite cyclically in drop down menu
Stop	Stop excitation
Report	Drop down menus support show/hide report and export report functions.
Add	Drop down menus support add item or item suite in current selected item.
Save	Save project.
Cut	Cut the current selected item.
Copy	Copy the current selected item.
Paste	Paste in the current selected item.
Delete	Delete the current selected item.
Properties	Show properties dialog of the current selected item.

Toolbar when direct excitation project opened (communication interface folded)



命令	功能
Excite	Excite selected item or item suite once, and excite cyclically in drop down menu
Stop	Stop excitation
Add	Drop down menus support add item in current selected item.
Save	Save project.
Delete	Delete the current selected item.
Properties	Show properties dialog of the current selected item.

Regular Tools Toolbar



命令	功能
Calculator	Run the calculator of the windows system.
Checksum	Run Checksum Calculator.
CRC	Run CRC Calculator.
DES	Run DES Calculator.
Hash	Run Hash Calculator.
ASCII Checklist	Run ASCII Checklist.
Unicode Converter	Run Unicode Converter

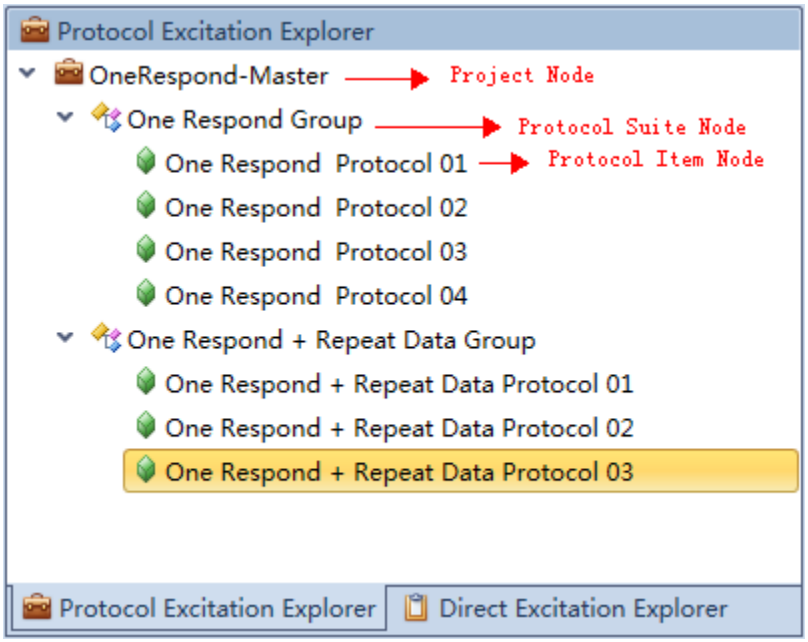
2.2.4 Status Bar

The status bar is divided into three parts, which are operating status, communication transmission byte, communication interface state.

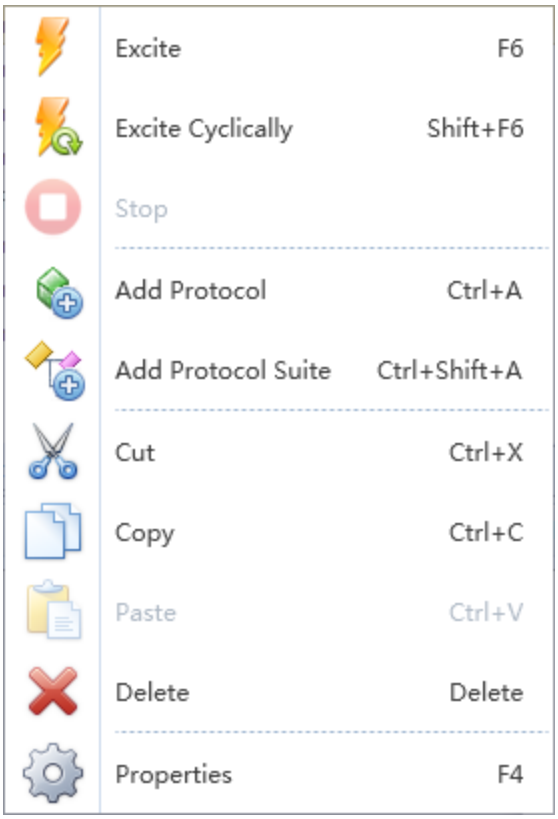
Status Bar	Functions
Operating Status	Show previous operating status.
Communication Transmission byte	Show total recv/send bytes since communication interface opened.
Communication Interface State	Show current communication interface state.

2.2.5 Control Panel

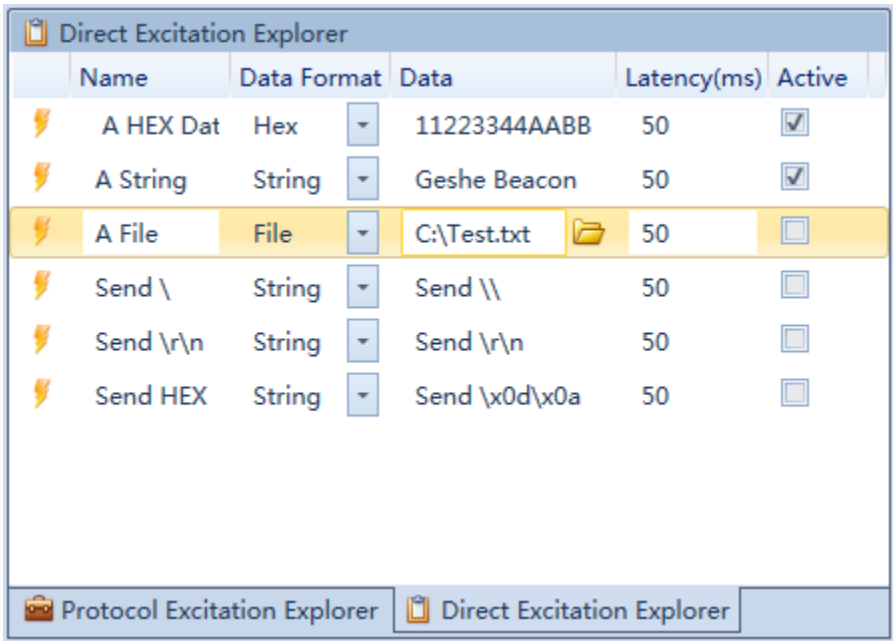
Protocol Excitation Project's control panel uses tree structure to organize items.



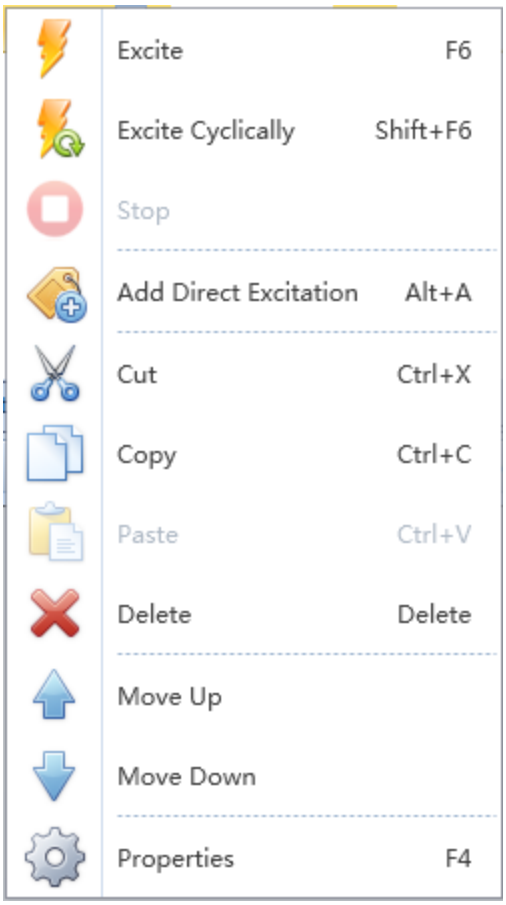
Protocol Excitation Project's shortcut menu is opened when mouse right button clicked.



Direct Excitation Project's control panel uses table structure to organize items. It supports three type of data format witch are Hex, String and File format, supports active property. Selected item can be excited immediately using the drive button on the left column.

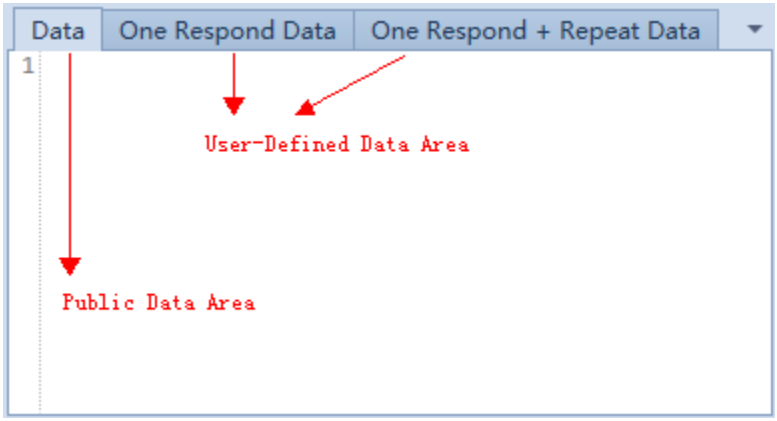


Direct Excitation Project's shortcut menu is opened when mouse right button clicked.



2.2.6 Data Area

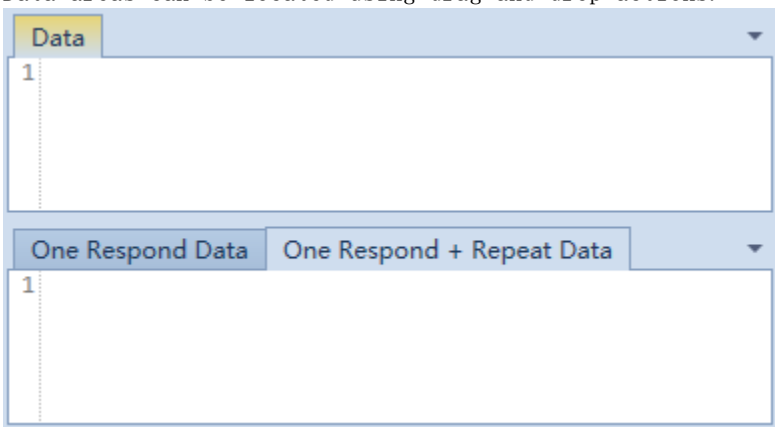
Data area can display excitation data. Data are divided into public data, custom data and report data.



Data Type	Functions
Public Data	Show the actual transfer data. Sending data can be disable using settings. Protocol transfer data is showed here if no custom data type.

Custom Data	Only protocol excitation project can define custom data type. Custom data will be showed in different data panel.
Report Data	Only protocol excitation project has running report. Report data can be showed in separate data panel.

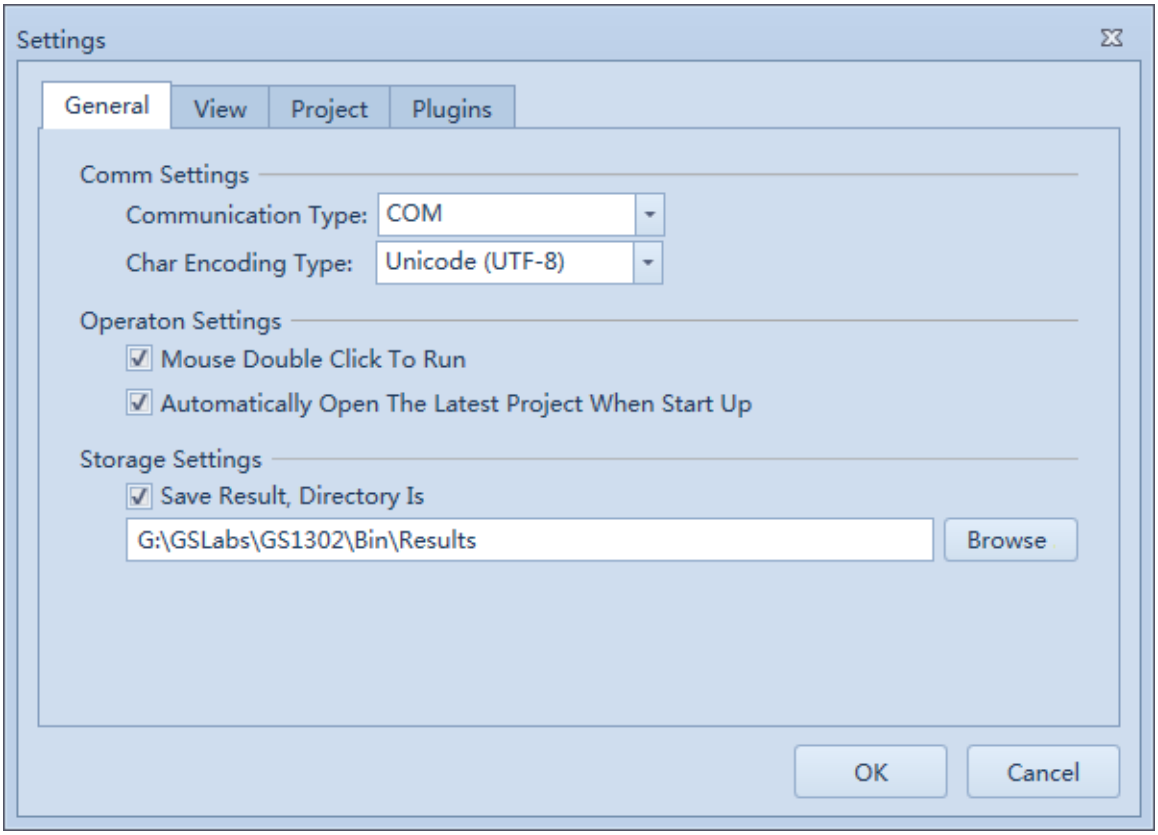
Data areas can be located using drag and drop actions.



2.3 Software Settings

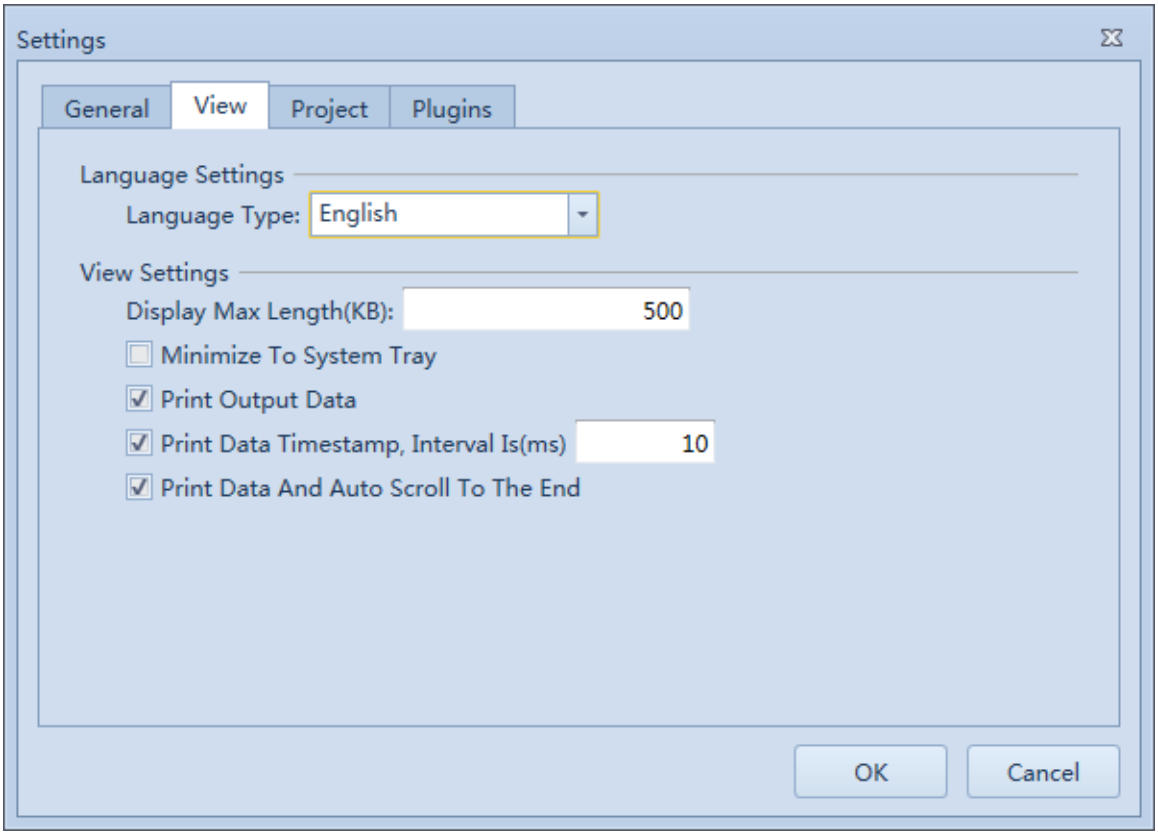
Open settings dialog:【Application Menu】->Settings.

2.3.1 General Settings



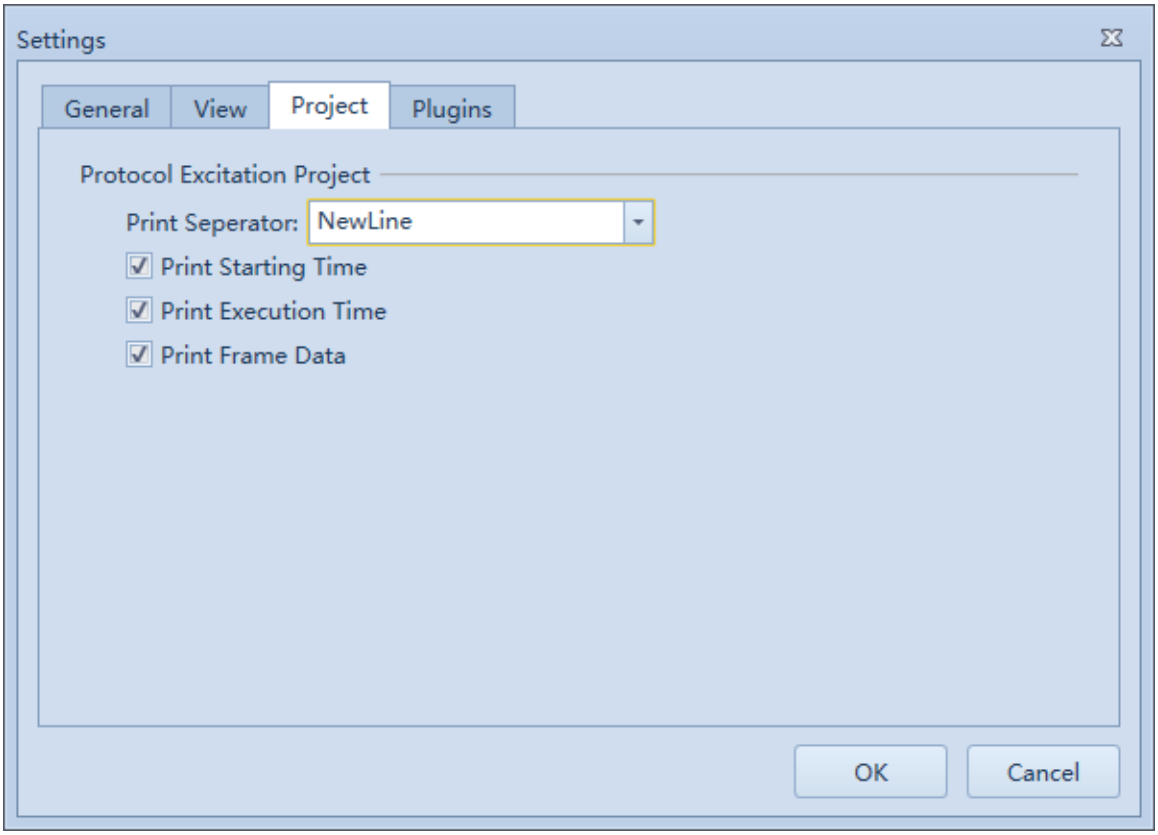
Parameters	Functions
Communication Type	Set communication type. Support COM and Network interface.
Char Encoding Type	Set character encoding type for all string.
Mouse Double Click To Run	Set up the incentive item can be a double click to run when the communication interface opened.
Automatically Open The Latest Project When Start Up	Automatically open the latest project when the software starts.
Save Result	Enable save result function and set the output directory.

2.3.2 View Settings



Parameters	Functions
Language Type	Select the UI language for the software.
Display Max Length(KB)	Set the maximum number of bytes in the data area. If the number is more than the maximum, the data panel automatically clears the oldest data.
Minimize To System Tray	Main window will be minimized to the system tray when enabled.
Print Output Data	Sent data will be displayed in the “Data” panel when enabled.
Print Data Timestamp	Data timestamp will be displayed in the “Data” panel when enabled. Timestamp is printed after every transfer if the interval parameter is 0. Timestamp is printed in a certain time when the interval is greater than 0.
Print Data and Auto Scroll to The End	The “Data” panel will automatically scroll to the end when data come.

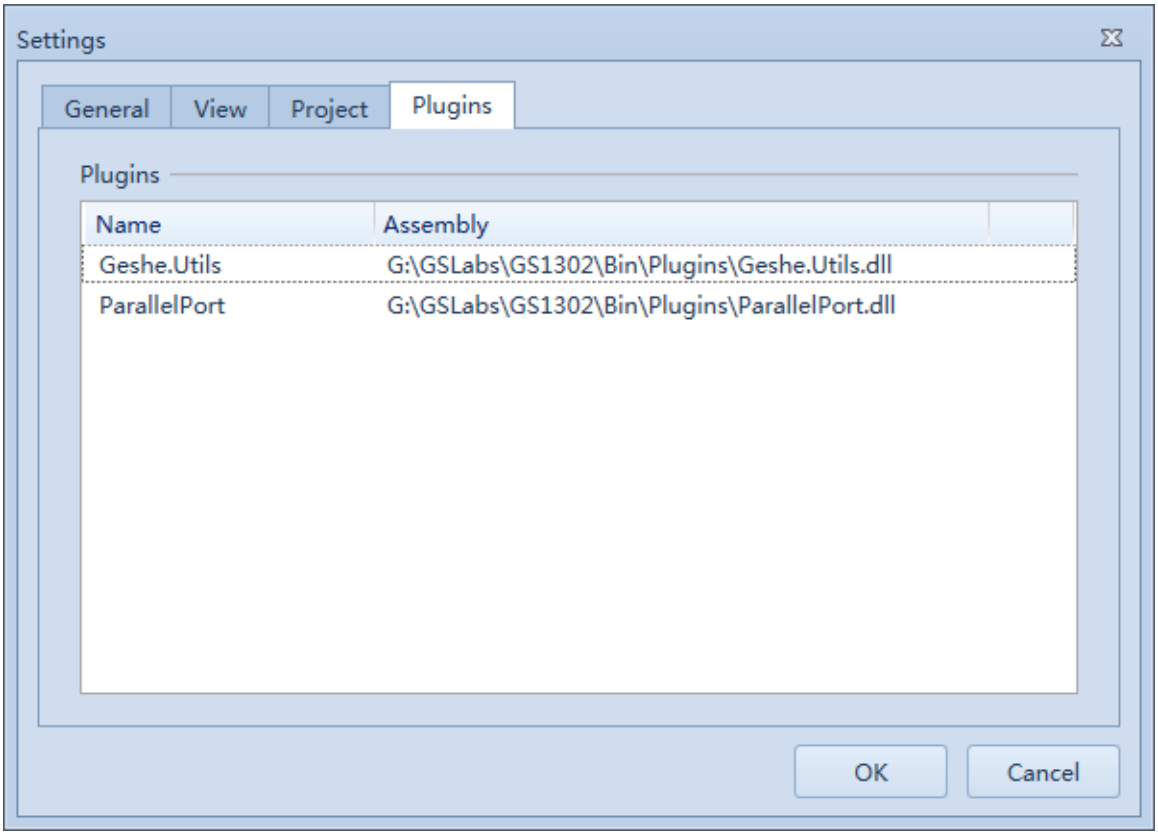
2.3.3 Project Settings



Protocol Excitation Project Parameters	Functions
Print Separator	Set up separator for the data segments of the excitation result. Support “New Line” and “Space” separators.
Print Starting Time	Set up whether to display the “Time” data segment
Print Execution Time	Set up whether to display the “Time Consumed” data segment
Print Frame Data	Set up whether to display the “Request” and “Respond “ data segments.

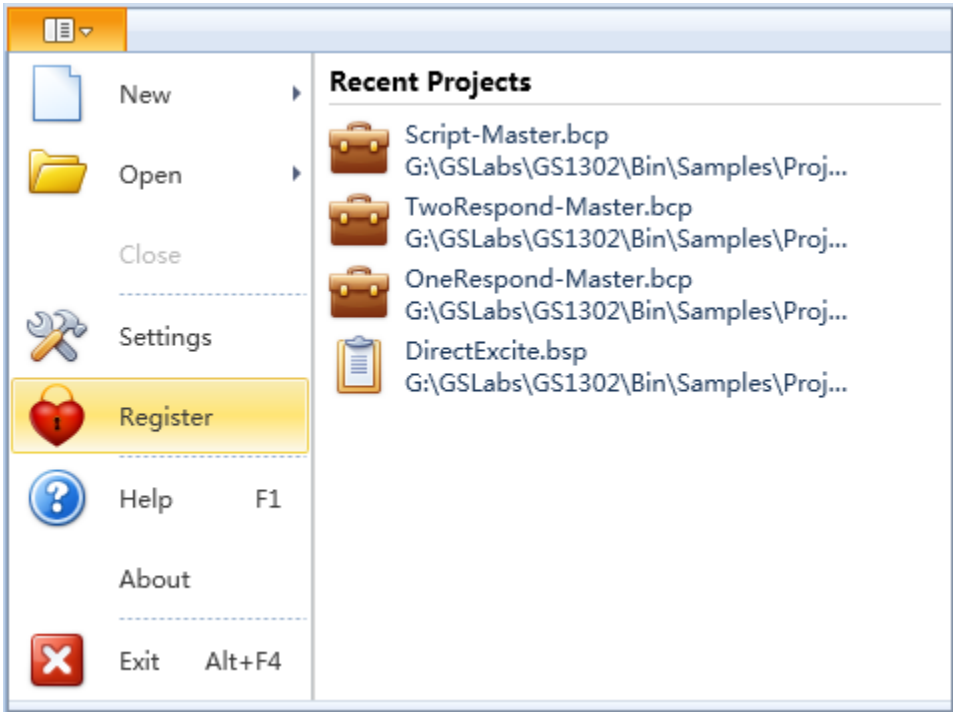
2.3.4 Plugins Information

List all plugins in the Plugins directory of the software install directory which can be known as managed components by the software based on Microsoft .NET Framework.



2.4 Software Registration

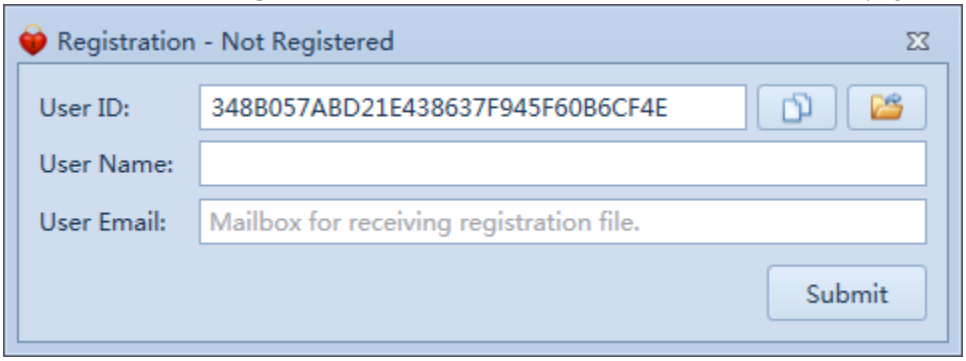
Software registration is for professional version. No need to register when using free version.



The software can be registered step by step:

Step 1: 【Application Menu】-> Register.

User ID is user computer's character code and can be automatically generated.



Step 2: Input User Name and User Email. Or use "Import" button to import a license file.
Please input the true email address because the email is for the license file.

Step 3: Click "Summit" button.
Registration information will be sent to the registration server. If the action fails or the software has been licensed in this computer before, the software will start email to send some information to us.

Step 4: Purchase the software.
Please refer to 1.4.3 section to know how to purchase.

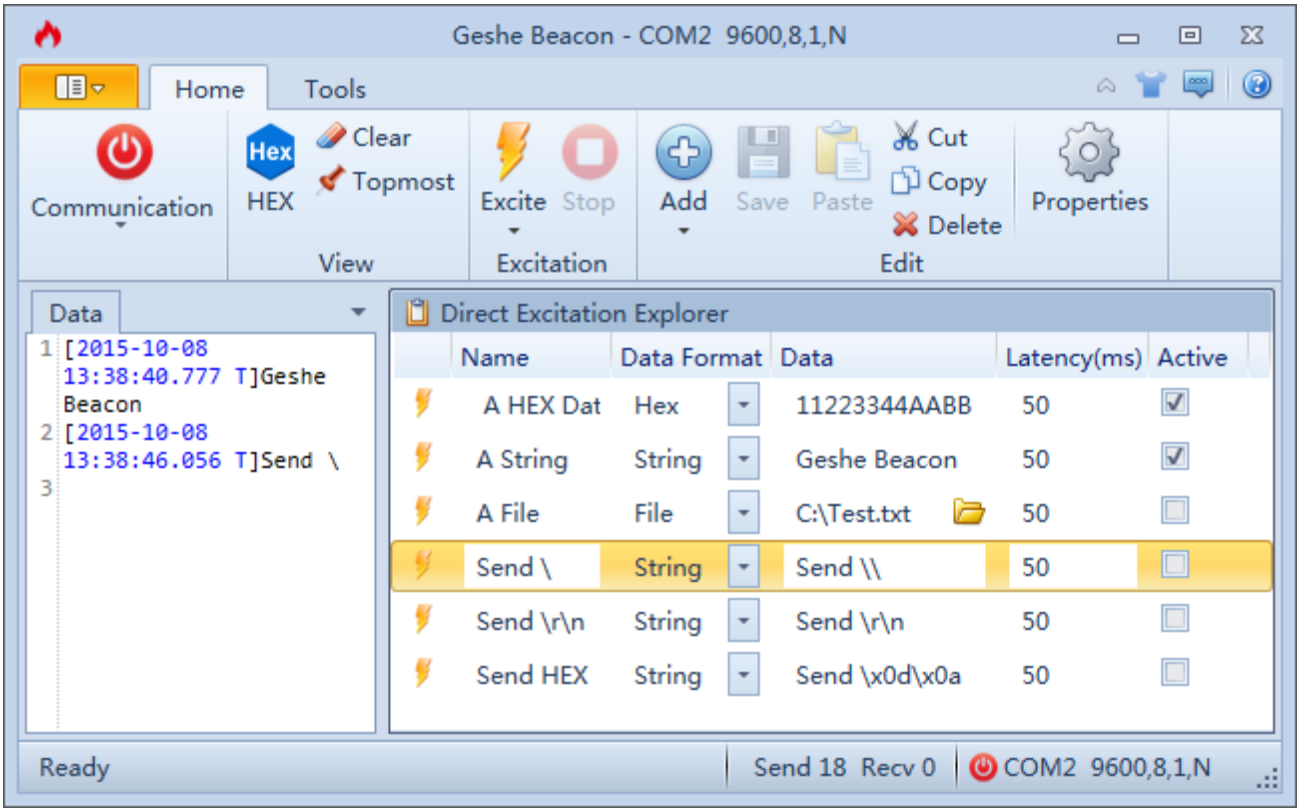
Step 5: When purchased successfully, we will send you license file using email or other transfer methods within 24 hours.

Step 6: Use the received license file to overlap the license.xml file in the install directory.

3. Direct Excitation Project

3.1 Brief

Direct Excitation Project supports send data function and uses list structure to organize items. It supports Hex, String and File format, supports item delay, supports disabling item, and supports immediate execution of an incentive.



3.2 Basic Operation

3.2.1 New Direct Excitation Project

Step 1: 【Application Menu】->“New”->“Direct Excitation Project”.
Step 2: Select project path, input project name and click “Save”.

3.2.2 Open Direct Excitation Project

Method 1:
Step 1: 【Application Menu】->“Open”->“Direct Excitation Project”.
Step 2: Select the project file and click “Open”.

Method 2:
Step 1: 【Application Menu】->Use “Recent Projects” panel to open a project.

3.2.3 Close Direct Excitation Project

Step 1: 【Control Panel】->Select “Direct Excitation Project Explorer”.
Step 2: 【Application Menu】->“Close”.

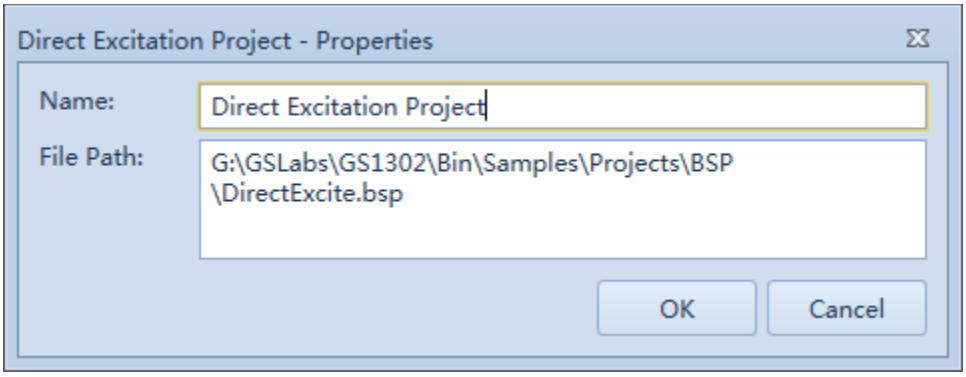
3.2.4 Save Direct Excitation Project

Step 1: 【Control Panel】->Select "Direct Excitation Project Explorer".
Step 2: 【Application Menu】->"Save" Or "Save As".

3.2.5 Modify Direct Excitation Project Properties

Step 1: 【Control Panel】->Select "Direct Excitation Project Explorer".
Step 2: 【Toolbar】->"Properties"; Or【Control Panel】->Click right mouse button to open the shortcut menu ->"Properties".

Direct Excitation Project Properties



Properties	Description
Name	Show or modify project name.
File Path	Show project file path and can't be modified.

3.2.6 Add Direct Excitation Item

Step 1: 【Control Panel】->Select "Direct Excitation Project Explorer".
Step 2: Select an item in "Direct Excitation Project Explorer" , new item will be inserted before the selected item.
Step 3: 【Toolbar】->"Add"->"Add Direct Excitation"; Or【Control Panel】-> Click right mouse button to open the shortcut menu ->"Add Direct Excitation".
Step 4: Edit the new item's properties.

3.2.7 Delete Direct Excitation Item

Step 1: 【Control Panel】->Select "Direct Excitation Project Explorer".
Step 2: Select the item to be deleted in the "Direct Excitation Project Explorer".
Step 3: 【Toolbar】->"Delete"; Or 【Control Panel】-> Click right mouse button to open the shortcut menu ->"Delete".

3.2.8 Excite

Incentive premise: communication interface has been opened.
Step 1: 【Control Panel】->Select “Direct Excitation Project Explorer”.
Step 2: 【Toolbar】->“Excite”; Or 【Toolbar】->“Excite” Menu->“Excite”/ “Excite Cyclically”; Or 【Control Panel】-> Click right mouse button to open the shortcut menu ->“Excite”/ “Excite Cyclically”.

3.2.9 Stop Excite

Stop Incentive premise: Exciting.
Step 1: 【Control Panel】->Select “Direct Excitation Project Explorer”.
Step 2: 【Toolbar】->“Stop”; Or 【Control Panel】-> Click right mouse button to open the shortcut menu ->“Stop”.

3.3 Usability

Direct Excitation Project is for the following environments.

- As a command control panel to communicate with a device
- Add interference signal to the excited device and test the excited device.

4. Protocol Excitation Project

4.1 Brief

Protocol is an essential part of the communication system. How to organize and conduct effective protocol testing, to organize a low cost testing to meet the rapidly changing communication environment are concerned by more and more enterprises and R&D personnel. In the past, we need to customize a test tool for each of the communication protocol. Numerous custom testing tools greatly increase the test development, maintenance and learning costs. And these will decrease the competitiveness in the modern electronic Industry.

Geshe Debug Genius’s protocol excitation function can separate thoroughly the changeable communication protocol from the communication software and makes communication software able to cope with the rapidly changing communication environment.

Advantage of Geshe Debug Genius’s protocol excitation function:

- A single test tool to replace many custom testing tools, greatly reduce the cost of maintenance, test development and learning.
- The fast excitation source management greatly reduces development costs.
- Organize projects by documents, advantageous to the management of a large number of test items and test standard.
- Support cyclic test and test report, improve the level of test automation.

4.2 Basic Operation

4.2.1 New Protocol Excitation Project

Step 1: 【Application Menu】->“New”->“Protocol Excitation Project”.
Step 2: Select project path, input project name and click “Save”.

4.2.2 Open Protocol Excitation Project

Method 1:
Step 1: 【Application Menu】->“Open”->“Protocol Excitation Project”.
Step 2: Select the project file and click “Open”.

Method 2:
Step 1: 【Application Menu】-> Use “Recent Projects” panel to open a project.

4.2.3 Close Protocol Excitation Project

Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: 【Application Menu】->“Close”.

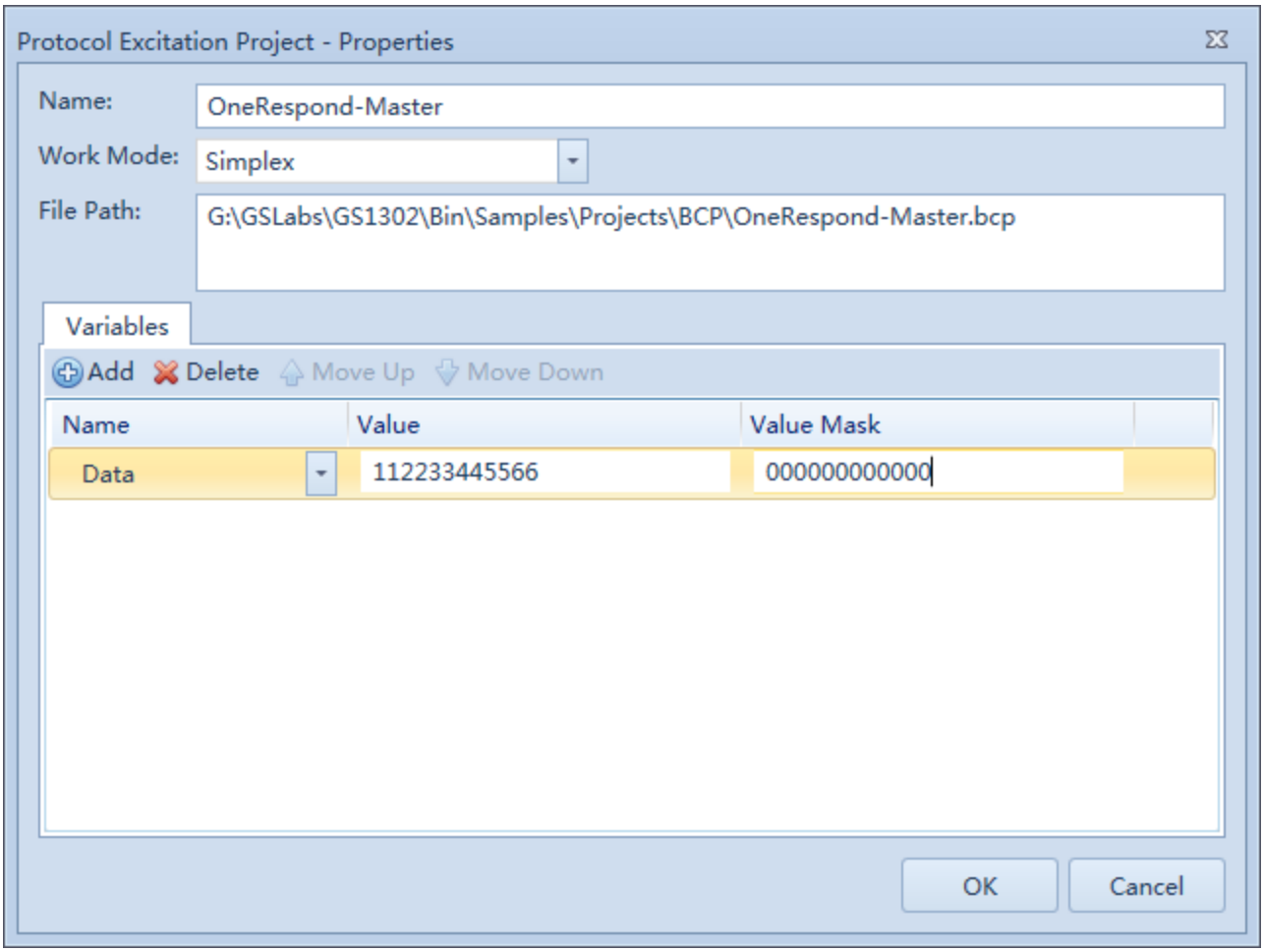
4.2.4 Save Protocol Excitation Project

Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: 【Application Menu】->“Save” Or “Save As”.

4.2.5 Modify Protocol Excitation Project Properties

Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: Select project item in “Protocol Excitation Project Explorer”.
Step 3: 【Toolbar】->“Properties”; Or 【Control Panel】->Click right mouse button to open the shortcut menu->“Properties”.

Protocol Excitation Project Properties

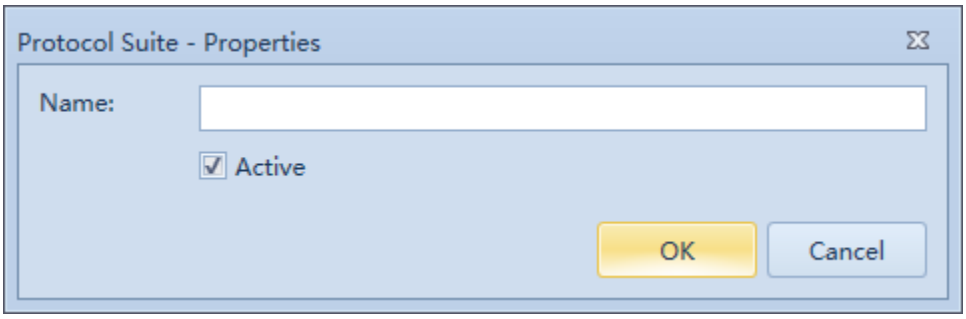


Properties	Description
Name	Show or modify project name.
Work Mode	Protocol Excitation Project supports Simplex and Duplex work modes. In ″Simplex″ mode excitation is in accordance with the selected order execution. In “Duplex” mode master items and slave items are executed simultaneously. Master items are executed in sequence while slave items are executed simultaneously. For example, if simulating master device, use “Simplex” or “Duplex” according to the demand. If simulating slave device, use “Duplex” in general.
File Path	Show project file path and can’t be modified.
Protocol Variables	Support global variables for the project. A record is composed of ″name″, ″value″ and ″value mask″. If the name of the corresponding field is consistent with the name of the variable name, and the default value is equal to the value mask, the value of the corresponding field is replaced by the value of the variable value during the execution. Generally used in frequently changing variable fields such as the similar device

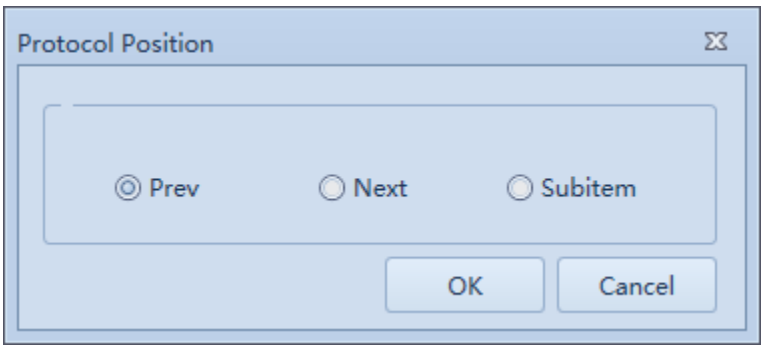
	address.

4.2.6 Add Protocol Suite

Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: Select a protocol item or protocol suite in “Protocol Excitation Project Explorer” as a location for the new protocol suite.
Step 3: 【Toolbar】->“Add”->“Add Protocol Suite”; Or 【Control Panel】->Click right mouse button to open the shortcut menu->“Add Protocol Suite”.
Step 4: Input suite name in the protocol suite properties dialog and click “OK”.

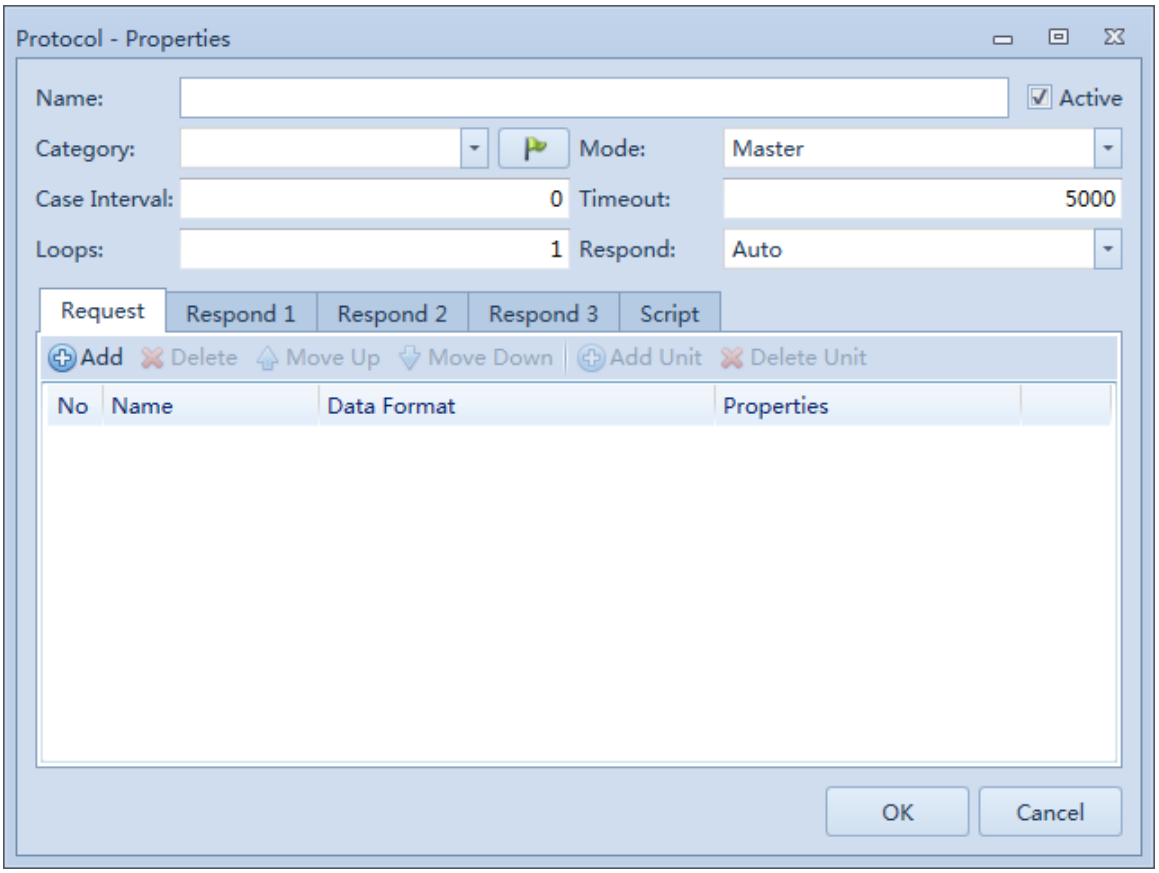


Step 5: Select the relative position for the new item and click “OK”.



4.2.7 Add Protocol

Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: Select a protocol item or protocol suite in “Protocol Excitation Project Explorer” as a location for the new protocol.
Step 3: 【Toolbar】->“Add”->“Add Protocol”; Or 【Control Panel】->Click right mouse button to open the shortcut menu->“Add Protocol”.
Step 4: Input parameters in the protocol properties dialog and click “OK”.



Properties	Description
Name	Protocol name
Active	Enable the item.
Category	In order to achieve the classification of the result data, Protocol Excitation Project can define custom data category. The result data can be sent to the corresponding custom data area in the excitation according to the category. Protocol Category can be managed by Protocol Category Dialog.
Mode	There are two kinds of modes "master mode" and "slave mode" for a protocol. Master mode means that the protocol is an active command, which is to send a request and check response frame. Slave mode indicates that the protocol is a passive command, which is waiting for a request frame, and then transmits the response frame according to the request frame.
Case Interval	Indicates how long after the execution of the protocol is to enter the next protocol.
Timeout	Indicate how long the protocol has to match the correct data before failure.
Loops	Times for cyclic excitation.
Respond	In master mode, indicate the expected response frame after the request is transmitted. The default means automatic matching.

	In slave mode, indicate which response will be sent when the request frame is received. The default value means response 1 will be sent.
Request	Show and edit the request frame. Refer to Section 4.3.
Respond 1 Respond 2 Respond 3	Show and edit the response frame. Refer to Section 4.3.
Script	Show and edit the protocol script. Refer to Section 4.4.

Step 5: Select the relative position for the new item and click “OK”.

4.2.8 Excite

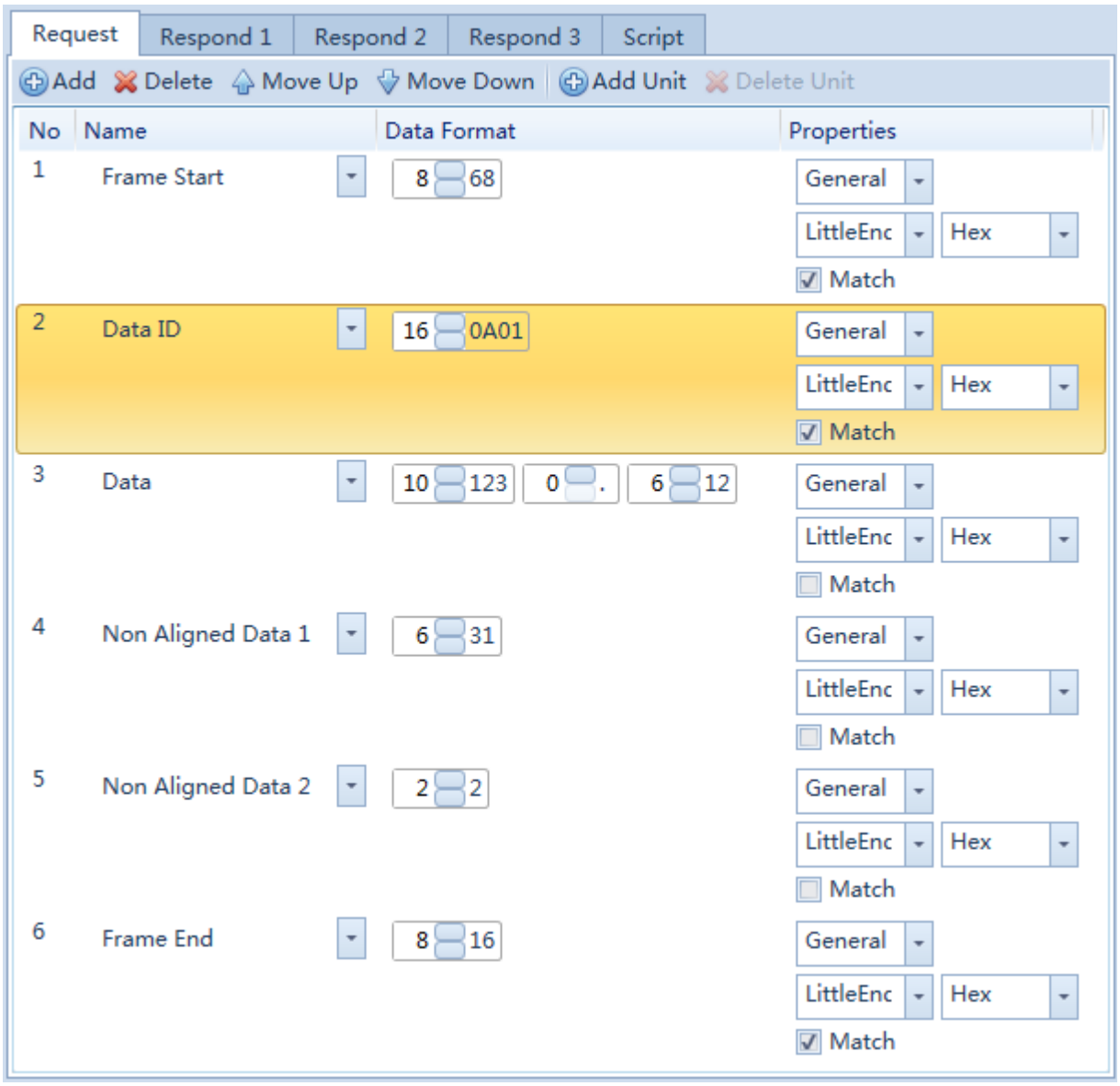
Incentive premise: communication interface has been opened.
Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: 【Toolbar】->“Excite”; Or 【Toolbar】->“Excite” Menu->“Excite”/ “Excite Cyclically”; Or 【Control Panel】->Click right mouse button to open the shortcut menu->“Excite”/ “Excite Cyclically”.

4.2.9 Stop Excite

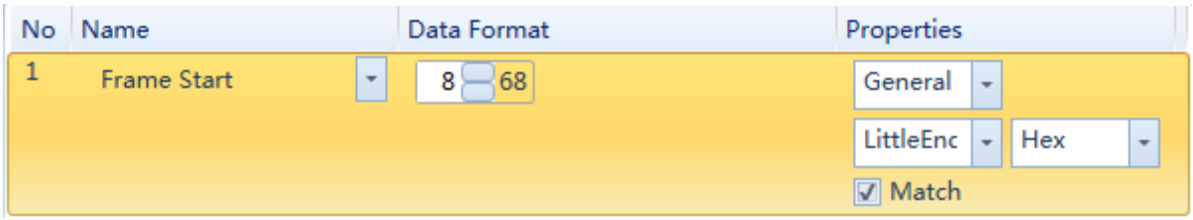
Stop Incentive premise: Exciting.
Step 1: 【Control Panel】-> Select “Protocol Excitation Project Explorer”.
Step 2: 【Toolbar】->“Stop”; Or 【Control Panel】->Click right mouse button to open the shortcut menu->“Stop”.

4.3 Protocol Frame

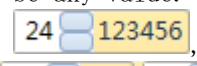
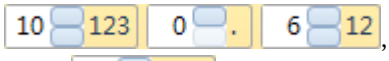
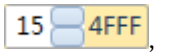
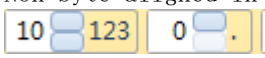
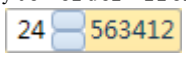
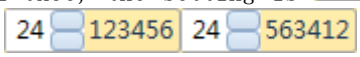
The protocol frame is composed of one or more frame format units. Frame format unit can be non-byte aligned. But total frame must be byte aligned.



Frame Format Unit:



Properties	说明
No	Sequence for the frame format unit.
Name	Name for the frame format unit.
Data Format	Represent the data structure of the frame unit, and can be composed of multiple data format. Data format, The left side of

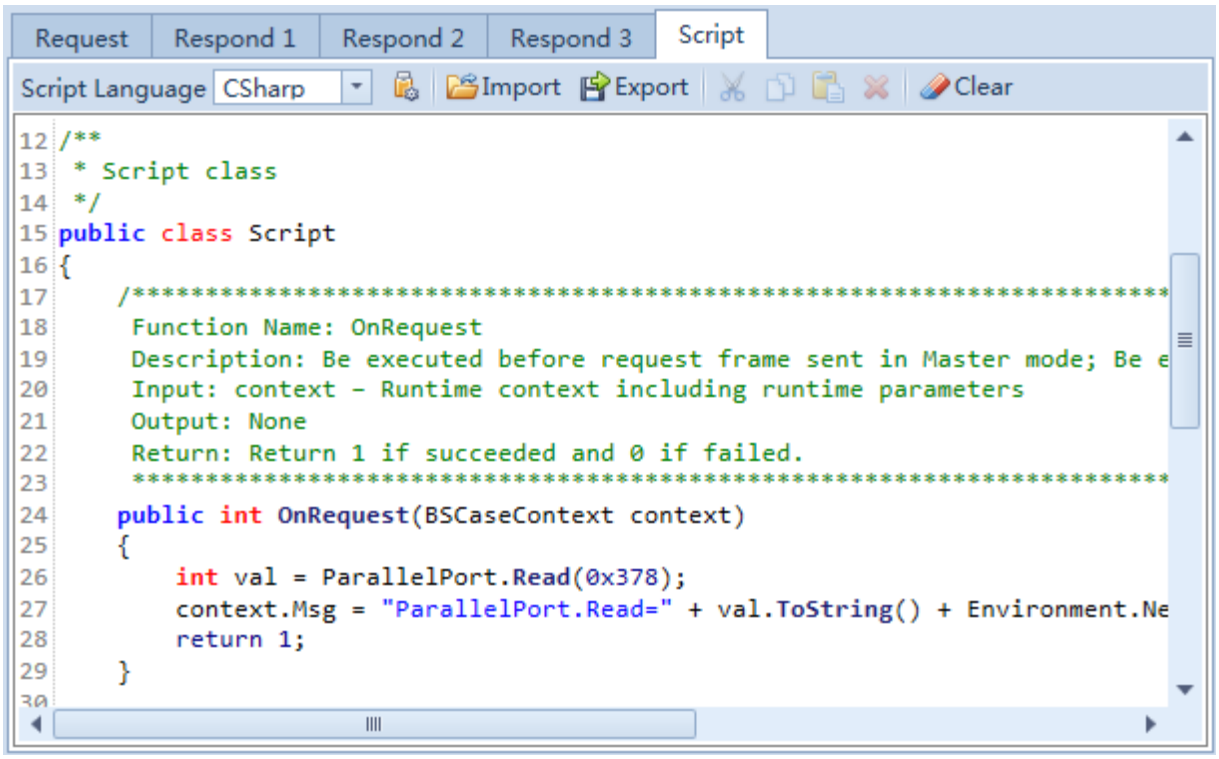
	the figure represent bit count, the right side of the figure indicates value. Bit count can be any value. For example: Byte aligned in single section  , Byte aligned in multi section  , Non-byte aligned in single section  , Non-byte aligned in multi section  .
Properties (Variable Type)	Frame format unit of the variable type supports three kinds of variables such as General variable, Calc variable and Repeat variable. General variable is a constant value. Calc variable is a variable which is computed by previous data, such as a checksum. Repeat variable indicates the variable is properly repeated any times in the frame. 0 represents 0~n repeated times and >0 represents an exact repeated times.
Properties (Storage Mode)	Frame format unit data support Little Endian mode and Big Endian mode. For example, if byte order from low to high is 0x12 0x34 0x56, the setting is  in Little Endian mode and  in Big Endian mode.
Properties (Display Mode)	Support Hex and String display modes. The character encoding type for String display mode is determined by the global settings.
Properties (Match)	Used to determine the integrity of the protocol frame.

Toolbar for Frame Format Unit:

Command	Function
Add	Insert a new frame unit in the currently selected frame format unit.
Delete	Delete the selected frame format unit.
Move Up	Move up a line for the selected frame format unit.
Move Down	Move down a line for the selected frame format unit.
Add Unit	Add a data format unit in the selected frame unit.
Delete Unit	Delete the selected data format unit.

4.4 Protocol Script

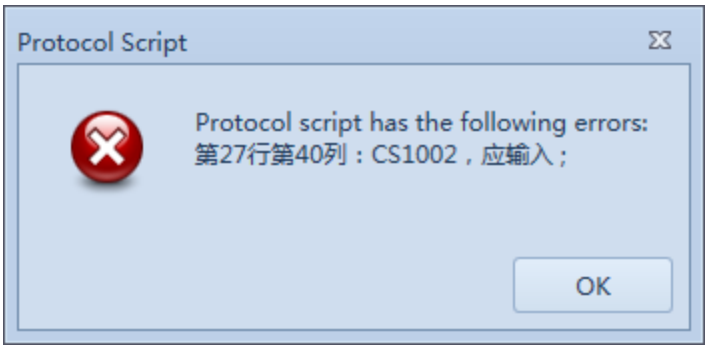
4.4.1 Script Editor



Toolbar for Script Editor

命令	功能
Script Language	Set script language for this item. Support C#, VB, Jscript.
Compile Check	Check the current script.
Import	Import a script from a file.
Export	Export the current script to a file.
Cut	Cut the selected script text.
Copy	Copy the selected script text.
Paste	Paste a text in the current cursor from the system clipboard.
Delete	Delete the selected script text.
Clear	Clear all script text.

If the script has errors, compile check function will show the location and error code of the errors.



4.4.2 Script Structure

Protocol script language supports C#, VB and Jscript. There are some templates of scripts in the directory Scripts in the software installed directory. You can use "Import" command to import a template to the script editor.

Below is the C# version of the script template. It is a .NET class and contains three methods, namely OnRequest method, OnProcess method, OnRespond method.

```
/*****************************************************************************
 Copyright (c) 2014, Shanghai Geshe Information Technology Co., Ltd.
 Filename: Script.cs
 Description: C# Script Template
*****************************************************************************/

/**
 * Namespace definition
 */
//using System;

/**
 * Script class
 */
public class Script
{
  /*****************************************************************************
   Function Name: OnRequest
   Description: Be executed before request frame sent in Master mode; Be executed after request frame received in Slave mode.
   Input: context – Runtime context including runtime parameters
   Output: None
   Return: Return 1 if succeeded and 0 if failed.
   *****************************************************************************/
  public int OnRequest(BSCaseContext context)
  {
    return 1;
  }

  /*****************************************************************************
```

```
   Function Name: OnProcess
   Description: Be executed once at about every 5ms while waiting to receive or sending
response.
   Input: context –Runtime context including runtime parameters
   Output: None
   Return: Return 1 if succeeded and 0 if failed.
   ***************************************************************************/
  public int OnProcess(BSCaseContext context)
  {
    return 1;
  }

  /****************************************************************************
   Function Name: OnRespond
   Description: Be executed after response frame sent in Slave mode; Be executed after
response frame received in Master mode.
  Input: context –Runtime context including runtime parameters
   Output: None
   Return: Return 1 if succeeded and 0 if failed.
   ***************************************************************************/
  public int OnRespond(BSCaseContext context)
  {
    return 1;
  }
}
```

4.4.3 Script Parameter BSCaseContext Class

The only input parameter for the script functions is a type of BSCaseContext. This type instance carries the script execution context for the entire process. This class provides properties and methods for script and can control and manage excitation, protocol frame and communication interface.

4.4.3.1 .Power

Get or set the operating state of the excitation, which indicates whether the excitation is running.

Syntax

C#	public bool Power { get; set; }
	Property Type: System.Boolean
VB	Public Property Power As Boolean Get Set
	Property Type: System.Boolean

Note

When you need to stop the drive, you can set the Power value to false in the script.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    context.Power = false;
	    context.Msg = "Power=false.\r\n";
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.Power = false
	    context.Msg = "Power=false.\r\n"
	    OnRequest = 1
	End Function

4.4.3.2 .CommParameters

Get the object to describe the communication interface. The object is a type of BSComStreamParameters in COM. Refer to Section 4.4.4. And it is a type of BSNetStreamParameters in Network. Refer to Section 4.4.5.

Syntax

C#	public object CommParameters { get; }
	Property Type: System.Object
VB	Public ReadOnly Property CommParameters As Object Get
	Property Type: System.Object

Note

When you need to get or modify the communication interface parameters in the script, you can switch the CommParameters to the communication interface parameter type, and then operate it.

Example

C#	public int OnRespond(BSCaseContext context)
	{
	    BSComStreamParameters comParams = context.CommParameters
	        as BSComStreamParameters;
	    comParams.BaudRate = 115200; // Change Baud Rate to 115200
	    comParams.Parity = Parity.Odd; // Set Parity to Odd
	
	    StringBuilder sb = new StringBuilder();
	    sb.Append(string.Format("COM parameters: BaudRate={0}, Parity={1}\r\n",
	       comParams.BaudRate, comParams.Parity));

	// Output info
	context.Msg = sb.ToString();
	
	return 1;
	}

VB	Public Function OnRespond (ByRef context As BSCaseContext) As Integer Dim comParams As BSComStreamParameters comParams = DirectCast(context.CommParameters, BSComStreamParameters) comParams.BaudRate = 115200 ' Change Baud Rate to 115200 comParams.Parity = Parity.Odd ' Set Parity to Odd Dim sb As StringBuilder sb = new StringBuilder() sb.Append(String.Format("COM parameters: BaudRate={0}, Parity={1}\r\n", comParams.BaudRate, comParams.Parity)) ' Output info context.Msg = sb.ToString() OnRequest = 1 End Function

4.4.3.3 .Msg

Get or set an information string that is displayed to the data area after the script function is finished.

Syntax

C#	public string Msg { get; set; }
	Property Type: System.String
VB	Public Property Msg As String Get Set
	Property Type: System.String

Note

When you need to display information to the data area after the script function executed, you can set the Msg value in the script function. If you need to set several message, you can use StringBuilder to combine all of the messages and set the Msg value or Use AppendMsg method.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	context.Msg = "Hello World.\r\n";
	return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.Msg = “Hello World.\r\n”
	    OnRequest = 1
	End Function

4.4.3.4 .MatchResult

Get the matched frame index. It starts from 0. If the matching frame is a request frame, the valid value is 0. If the matching frame is a response, the valid value is 0 or 1 or 2.

Syntax

C#	public int MatchResult { get; }
	Property Type: System.Integer
VB	Public Property MatchResult As Integer Get
	Property Type: System. Integer

Note

None

Example

C#	public int OnRespond (BSCaseContext context)
	{
	    if (context.MatchResult == 0)
	    {
	        // Expected to receive response frame 1
	        string msg = “Frame format unit 3 of the response frame 1:” + context.GetRespondValue(0,2) + “\r\n”;
	        context.Msg = msg;
	        return 1; // Return Success
	    }
	    return 0; // Return failure
	}

VB	Public Function OnRespond (ByRef context As BSCaseContext) As Integer
	    Dim msg As String
	    If context.MatchResult = 0 Then
	        ’ Expected to receive response frame 1
	        msg = “Frame format unit 3 of the response frame 1:” + context.GetRespondValue(0,2) + “\r\n”
	        context.Msg = msg
	        OnRespond = 1 ’Return Success
	    End If
	
	    OnRespond = 0 ’Return failure
	 End Function

4.4.3.5 .ExpectedRespond

Get or set the index of the Respond property. The valid value starts from 0. It indicates expected respond index in Master mode and the ready-to-send response index after the request frame received in Slave mode.

Syntax

C#	public int ExpectedRespond{ get; set; }
	Property Type: System.Integer
VB	Public Property ExpectedRespond As Integer Get Set
	Property Type: System. Integer

Note

None

Example

C#	public int OnRequest(BSCaseContext context)
	{
	// Expected response
	string msg = "Expected response:" + context.ExpectedRespond + "\r\n";
	context.Msg = msg;
	return 1; // Return Success
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer Dim msg As String msg = "Expected response:" + context.ExpectedRespond + "\r\n" context.Msg = msg OnRequest = 1 'Return Success End Function

4.4.3.6 .AppendMsg

Add an information string, which is attached to the original Msg. And the Msg will be displayed to the data area after the script function is executed.

Syntax

C#	public void AppendMsg(string msg)
	Parameters msg Type: System.String The new information string, which is attached to the original Msg

VB	Public Sub AppendMsg(msg As String) Parameters msg Type: System.String The new information string, which is attached to the original Msg

Note

When you need to set multiple messages, use the AppendMsg method.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    context.AppendMsg("Hello A.\r\n");
	    context.AppendMsg("Hello B.\r\n");
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.AppendMsg("Hello A.\r\n")
	    context.AppendMsg("Hello B.\r\n")
	    OnRequest = 1
	End Function

4.4.3.7 .GetVariant

Get the value of the user defined variable.

Syntax

C#	public object GetVariant(object key) Parameters key Type: System.Object Name of the user defined variable. It is a key and must be unique. Return Type: System.Object Value of the user defined variable. You can get the actual value by type conversion.
VB	Public Function GetVariant (key As Object) As Object Parameters key Type: System.Object Name of the user defined variable. It is a key and must be unique. Return Type: System.Object Value of the user defined variable. You can get the actual value by type

	conversion.

Note

When you need to set up a user defined variable in the excitation operation process, use the SetVariant method, and use the GetVariant method when you need to get it. The custom variables are valid for the entire drive period.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    context.SetVariant ("Str1", "Hello A.\r\n");
	    context.Msg = context.GetVariant("Str1").ToString();
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.SetVariant ("Str1", "Hello A.\r\n")
	    context.Msg = context.GetVariant("Str1").ToString()
	    OnRequest = 1
	End Function

4.4.3.8 .SetVariant

Set the value of the user defined variable.

Syntax

C#	public void SetVariant(object key, object val)
	Parameters key Type: System.Object Name of the user variable. If it exists, modify the old value. Or create a new user variable. val Type: System.Object Value of the user variable
VB	Public Sub SetVariant (key As Object, val As Object)
	Parameters key Type: System.Object Name of the user variable. If it exists, modify the old value. Or create a new user variable. val Type: System.Object Value of the user variable

Note

When you need to set user defined variables in the process of excitation, use the SetVariant

method.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    context.SetVariant (“Str1”, “Hello A.\r\n”);
	    context.Msg = context.GetVariant(“Str1”).ToString();
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.SetVariant (“Str1”, “Hello A.\r\n”)
	    context.Msg = context.GetVariant(“Str1”).ToString()
	    OnRequest = 1
	End Function

4.4.3.9 .GetRequestValue

Get the value of the frame format unit of the request frame.

Syntax

C#	public string GetRequestValue(int unitIndex)
	Parameters unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. Return Type: System.String The value of the frame format unit
VB	Public Function GetRequestValue (unitIndex As Integer) As String
	Parameters unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. Return Type: System.String The value of the frame format unit

Note

None

Example

C#	public int OnRespond (BSCaseContext context)
	{
	    string val = context.GetRequestValue (0);
	    context.Msg = val;

	return 1; }
VB	Public Function OnRespond (ByRef context As BSCaseContext) As Integer Dim val As String val = context.GetRequestValue (0) context.Msg = val OnRequest = 1 End Function

4.4.3.10 .SetRequestValue

Set the value of the frame format unit for the request frame.

Syntax

C#	public int SetRequestValue(int unitIndex, string unitValue)
	Parameters unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. unitValue Type: System.String The new value Return Type: System.Integer Return 0 if succeeded and return -1 if failed.
VB	Public Function SetRequestValue (unitIndex As Integer, unitValue As String) As Integer
	Parameters unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. unitValue Type: System.String The new value Return Type: System.Integer Return 0 if succeeded and return -1 if failed.

Note

When you need to change the value of the frame format unit of the protocol frame, use the SetRequestValue method.

Example

C#	public int OnRequest(BSCaseContext context) {

	context.SetRequestValue (1, “02”); return 1; }
VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer context.SetRequestValue (1, “02”) OnRequest = 1 End Function

4.4.3.11 .GetRespondValue

Get the value of the frame format unit of the response frame.

Syntax

C#	public string GetRespondValue (int respondIndex, int unitIndex)
	Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. Return Type: System.String The value of the frame format unit
VB	Public Function GetRespondValue (respondIndex As Integer, unitIndex As Integer) As String
	Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. Return Type: System.String The value of the frame format unit

Note

Get the value of the frame format unit of the response frame, and judge whether the received response is correct.

Example

C#	public int OnRespond (BSCaseContext context)
	{
	string val = context.GetRespondValue(0, 1);
	context.Msg = val;

	return 1; }
VB	Public Function OnRespond (ByRef context As BSCaseContext) As Integer Dim val As String val = context.GetRespondValue(0, 1) context.Msg = val OnRequest = 1 End Function

4.4.3.12 .SetRespondValue

Set the value of the frame format unit of the response frame.

Syntax

C#	public int SetRepondValue(int respondIndex, int unitIndex, string unitValue)
	Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. unitValue Type: System.String The new value Return Type: System.Integer Return 0 if succeeded and return -1 if failed.
VB	Public Function SetRepondValue (respondIndex As Integer, unitIndex As Integer, unitValue As String) As Integer
	Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. unitIndex Type: System.Integer The index of the frame format unit. It starts from 0. unitValue Type: System.String The new value Return Type: System.Integer Return 0 if succeeded and return -1 if failed.

Note

The method is generally used in the slave mode protocol.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    context.SetRepondValue (0, 1, "02");
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    context.SetRepondValue (0, 1, "02")
	    OnRequest = 1
	End Function

4.4.3.13 .GetRequestData

Get the byte array of the request data.

Syntax

C#	public byte[] GetRequestData() Parameters None Return Type: System.Byte[] The byte array of the request data
VB	Public Function GetRequestData () As Byte() Parameters None Return Type: System.Byte() The byte array of the request data

Note

None

4.4.3.14 .GetRespondData

Get the byte array of the respond data.

Syntax

C#	public byte[] GetRespondData(int respondIndex) Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. Return

	Type: System.Byte[] The byte array of the respond data
VB	Public Function GetRespondData (respondIndex As Integer) As Byte() Parameters respondIndex Type: System.Integer The index of the respond frame. 0, 1 and 2 are the valid value. Return Type: System.Byte() The byte array of the respond data

Note
None

4.4.3.15 .GetMatchData

Get the byte array of the matched result.

Syntax

C#	public byte[] GetMatchData() Parameters None Return Type: System.Byte[] the byte array of the matched result
VB	Public Function GetMatchData() As Byte() Parameters None Return Type: System.Byte() the byte array of the matched result

Note
None

4.4.4 COM Parameter BSComStreamParameters Class

4.4.4.1 .Port

Get or set the serial port

Syntax

C#	public string Port { get; set; }

	Property Type: System.String
VB	Public Property Port As String Get Set
	Property Type: System.String

Note

Example

C#	public int OnRequest(BSCaseContext context)
	{
	BSComStreamParameters comParams = context.CommParameters
	as BSComStreamParameters;
	StringBuilder sb = new StringBuilder();
	sb.Append(string.Format("COM Port: Port={0}\r\n",
	comParams.Port));
	// Output info
	context.Msg = sb.ToString();
	
	return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer Dim comParams As BSComStreamParameters comParams = DirectCast(context.CommParameters, BSComStreamParameters) Dim sb As StringBuilder sb = new StringBuilder() sb.Append(String.Format("COM Port: Port={0}\r\n", comParams.Port)) ' Output info context.Msg = sb.ToString() OnRequest = 1 End Function

4.4.4.2 .BaudRate

Get or set the baud rate of serial communication port.

Syntax

C#	public int BaudRate { get; set; }
	Property Type: System.Integer
VB	Public Property BaudRate As Integer Get Set

	Property Type: System.Integer

Note

The system must support the baud rate.

Example

C#	public int OnRequest(BSCaseContext context)
	{
	    BSComStreamParameters comParams = context.CommParameters
	        as BSComStreamParameters;
	    comParams.BaudRate = 115200; // Set baud rate to 115200
	
	    return 1;
	}

VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer
	    Dim comParams As BSComStreamParameters
	    comParams = DirectCast(context.CommParameters, BSComStreamParameters)
	
	    comParams.BaudRate = 115200 'Set baud rate to 115200
	    OnRequest = 1
	End Function

4.4.4.3 .DataBits

Get or set the standard data bit length of each byte.

Syntax

C#	public int DataBits{ get; set; }
	Property Type: System.Integer

VB	Public Property DataBits As Integer
	    Get
	    Set

	Property Type: System.Integer

Note

The value range for this property is 5 to 8

Example

None

4.4.4.4 .Parity

Get or set the parity check protocol.

Syntax

C#	public Parity Parity{ get; set; }
	Property Type: System.IO.Ports .Parity Support None, Odd, Even, Mark and Space. Default value is None.
VB	Public Property Parity As Parity Get Set
	Property Type: System.IO.Ports .Parity Support None, Odd, Even, Mark and Space. Default value is None.

Note

If a parity error occurs on the end of the stream, an additional byte of a value of 126 is added to the input buffer.

Example

None

4.4.4.5 .StopBits

Get or set the standard stop bits for each byte.

Syntax

C#	public StopBits StopBits{ get; set; }
	Property Type: System.IO.Ports .StopBits Support 1, 1.5, 2.
VB	Public Property StopBits As StopBits Get Set
	Property Type: System.IO.Ports .StopBits Support 1, 1.5, 2.

Note

Default value is 1.

Example

None

4.4.4.6 .Handshake

Get or set the handshake protocol for serial port data transfer.

Syntax

C#	public Handshake Handshake{ get; set; }

	Property Type: System.IO.Ports .Handshake Support None, RequestToSend, XonXoff, RequestToSend /XonXoff.
VB	Public Property Handshake As Handshake Get Set
	Property Type: System.IO.Ports .Handshake Support None, RequestToSend, XonXoff, RequestToSend /XonXoff.

Note

Example
None

4.4.4.7 .DtrEnable

Enable or disable DTR signal in serial communication.

Syntax

C#	public bool DtrEnable{ get; set; }
	Property Type: System .Boolean True to enable DTR. Default value is false.
VB	Public Property DtrEnable As Boolean Get Set
	Property Type: System .Boolean True to enable DTR. Default value is false.

Note
Data terminal ready (DTR) is typically enabled in the process of XON/XOFF software handshake, request sending / (RTS/CTS) hardware handshake and modem communications.

Example
None

4.4.4.8 .RtsEnable

Enable or disable RTS signal in serial communication.

Syntax

C#	public bool RtsEnable{ get; set; }
	Property Type: System .Boolean

	True to enable RTS. Default value is false.
VB	Public Property RtsEnable As Boolean Get Set
	Property Type: System.Boolean True to enable RTS. Default value is false.

Note
A request for sending (RTS) signal is usually used in the request to send / transmit (RTS/CTS) hardware handshake.

Example
None

4.4.5 Network Parameter BSNetStreamParameters Class

4.4.5.1 .Socket

Get network communication interface.

Syntax

C#	public Socket Socket { get; }
	Property Type: System.Net.Sockets.Socket
VB	Public Property Port As Socket Get Set
	Property Type: System.Net.Sockets.Socket

Note

4.4.6 Use Plugins in Script

This software supports the function of the managed code component based on.NET Framework Microsoft as a function of the plug-in extension script. Refer to Section 5.

In the script, it is convenient to use the plug-in, which directly calls the public service provided by the plug-in.

Example
The following example calls the ParallelPort.dll plug-in in the script OnRequest method. (Source code is located in the software installation directory - Samples\Plugins\ParallelPort).

C#	public int OnRequest(BSCaseContext context)
	{
	int val = Geshe.Utils.ParallelPort.Read(0x378);

	context.Msg = “Read ParallelPort 0x378. Value=“ + val.ToString(); return 1; }
VB	Public Function OnRequest(ByRef context As BSCaseContext) As Integer Dim val As Integer val = Geshe.Utils.ParallelPort.Read(&H378) context.Msg = “Read ParallelPort 0x378. Value=“ + val.ToString() OnRequest = 1 End Function

Note: The first access to plugin ParallelPort.dll will install parallel port driver to system installation directory System32. This operation needs administrator privileges to perform.

5. Plugins

The software dynamically identifies all managed code components based on.NET Framework Microsoft in the Plugins directory, and takes them as a plugin in the script to call their public services.

5.1 Managed Code and Unmanaged Code

Microsoft .NET Framework is a set of windows components that supports the generation and operation of the next generation of applications and network services. Microsoft.NET Framework has two main components which are CLR (Common Language Runtime) and .NET Framework Class Library. The common language runtime is the base of Framework.NET, which can be viewed as a proxy for managing code in execution time. It provides core services such as memory management, thread management, and remote processing, and also enforces strict type safety and other forms of code accuracy that can improve security and reliability. As a matter of fact, the concept of code management is the basic principle of the runtime. Code that runs the library as the target is called managed code, and the code that does not run the library as the target is called the unmanaged code. The other component of Microsoft.NET Framework is Class Library. It is an integrated object-oriented reusable type collection that can be used to develop a variety of applications and network services.

The difference between managed and unmanaged code:

- The managed code is an Interlingua, based on the runtime, run in CLR; Unmanaged code not based on the runtime, is compiled into machine code and run on the machine.
- The managed code is independent of platform and language, can realize the compatibility between different language platform; Unmanaged code depends on the platform and language.
- The managed code can use CLR services, such as security detection, garbage collection, do not need to complete the operation of their own; Unmanaged code, you need to provide their own security detection, garbage collection and other operations.

Managed code can be written in a high level language with more than 20 supports for Microsoft.NET Framework including C#, J#, Microsoft Visual Basic .NET, Microsoft JScript .NET, Visual C++ .NET and etc.

5.2 Write Plugins

All managed code components based on.NET Framework Microsoft can be identified by the software, and as a plugin in the script to use them to provide functionality, interface protocol without any requirements.

The process of writing a plug-in and the preparation of ordinary Microsoft.NET Framework based managed code libraries and applications, as long as the function is to ensure that the function is open (public).

5.3 Use the 3rd Party's Managed Code

Directly copy managed code of the third party libraries and its dependences on the library based on Microsoft.NET Framework to the Plugins directory of the software.

5.4 Use the 3rd Party's Unmanaged Code

Unmanaged code, such as driver library written in C/C++, cannot be identified and used directly by the software. It is used in an indirect way through a managed code library package.

First, write a managed code class library based on Microsoft.NET Framework. In this class library, through the platform invoke services provided in the Interop System.Runtime.InteropServices namespace, you can interoperate with the unmanaged code.

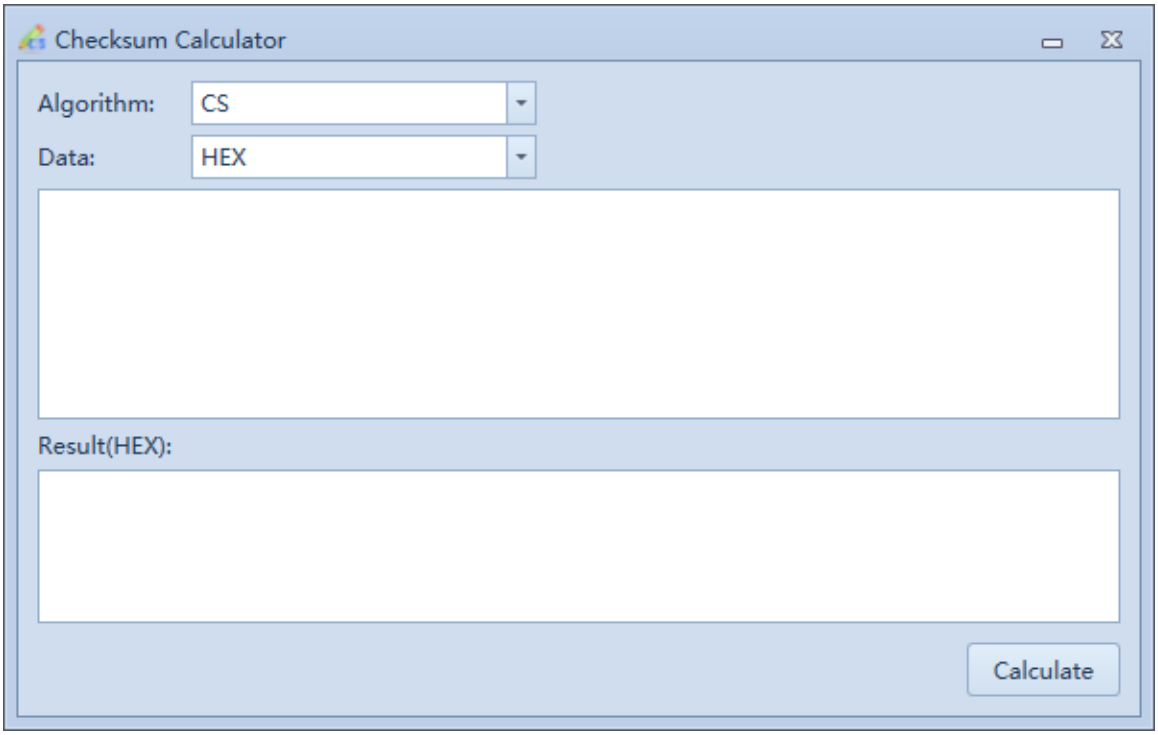
Finally, copy the packaged library and the references to the unmanaged code libraries to the Plugins directory, the script can execute unmanaged code library by visiting the library package.

6. Toolbox

6.1 Checksum Calculator

Checksum calculator is a calculator for BCC, CS, LRC calculation, the data source can be HEX data string, string (character encoding type is decided by the global encoding settings) and file.

Main UI:

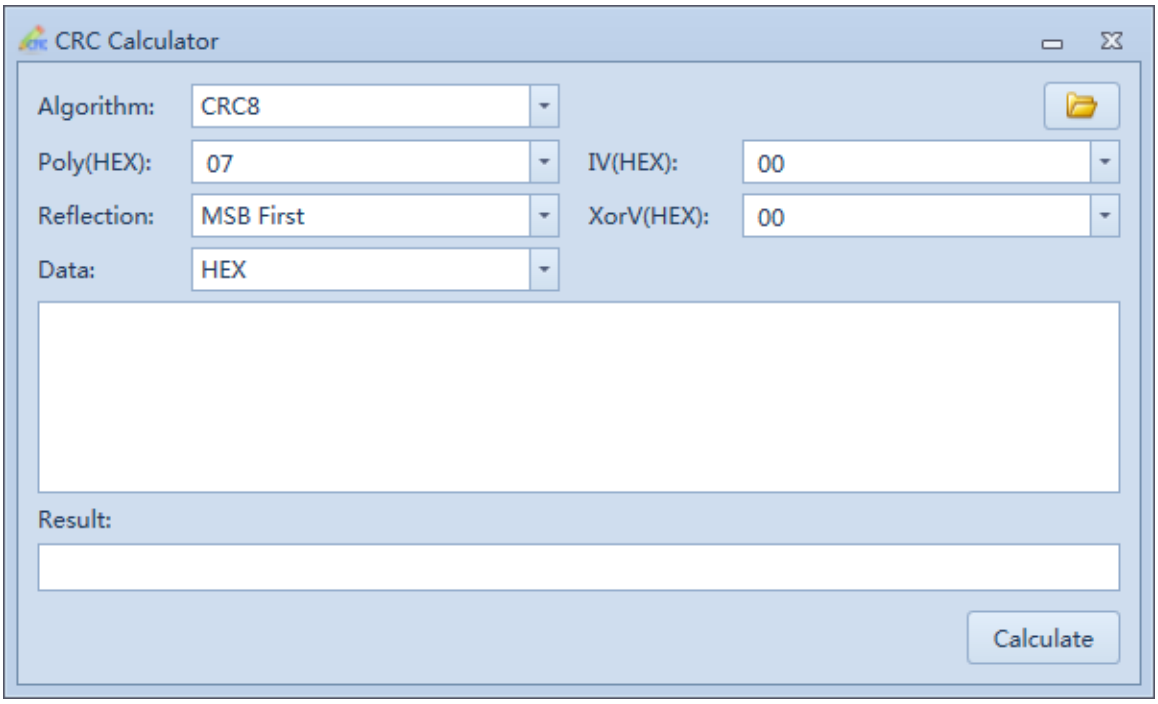


6.2 CRC Calculator

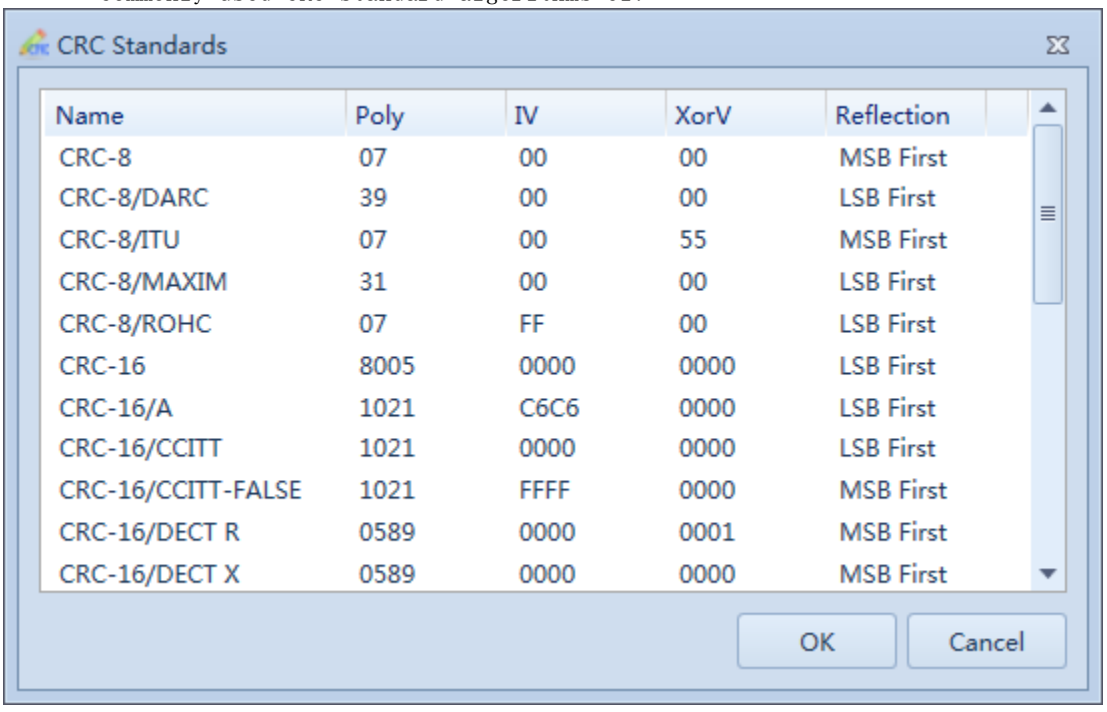
CRC calculator is a general Redundancy Check Cyclic (cyclic redundancy check code) computing tool. It supports CRC8, CRC16 and CRC32 algorithms. Polynomial, initial value, data inversion and XOR result can be defined by user. The data source can be HEX data string, string (character encoding type is decided by the global encoding settings) and file.

It supports many commonly used CRC standard algorithms.

Main UI:



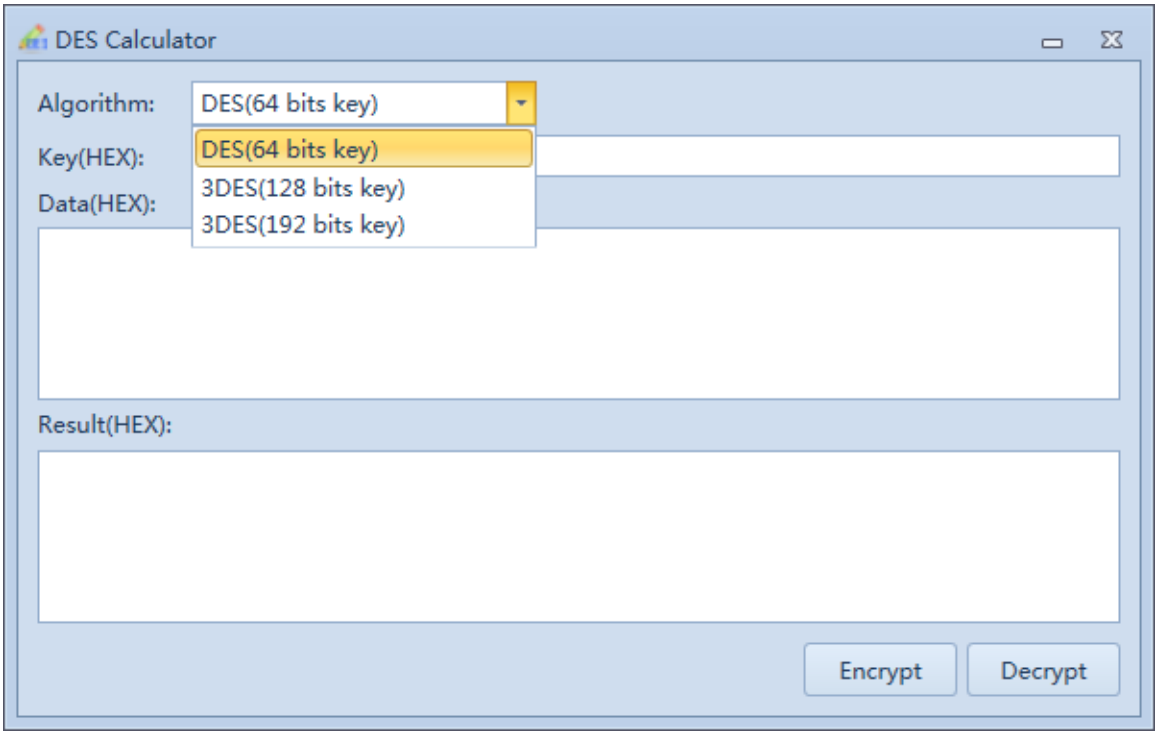
Commonly used CRC standard algorithms UI:



6.3 DES Calculator

DES calculator is a DES (64 bit key), 3DES (128 bit key), 3DES (192 bit key) DES computing tools. The data source is HEX data string.

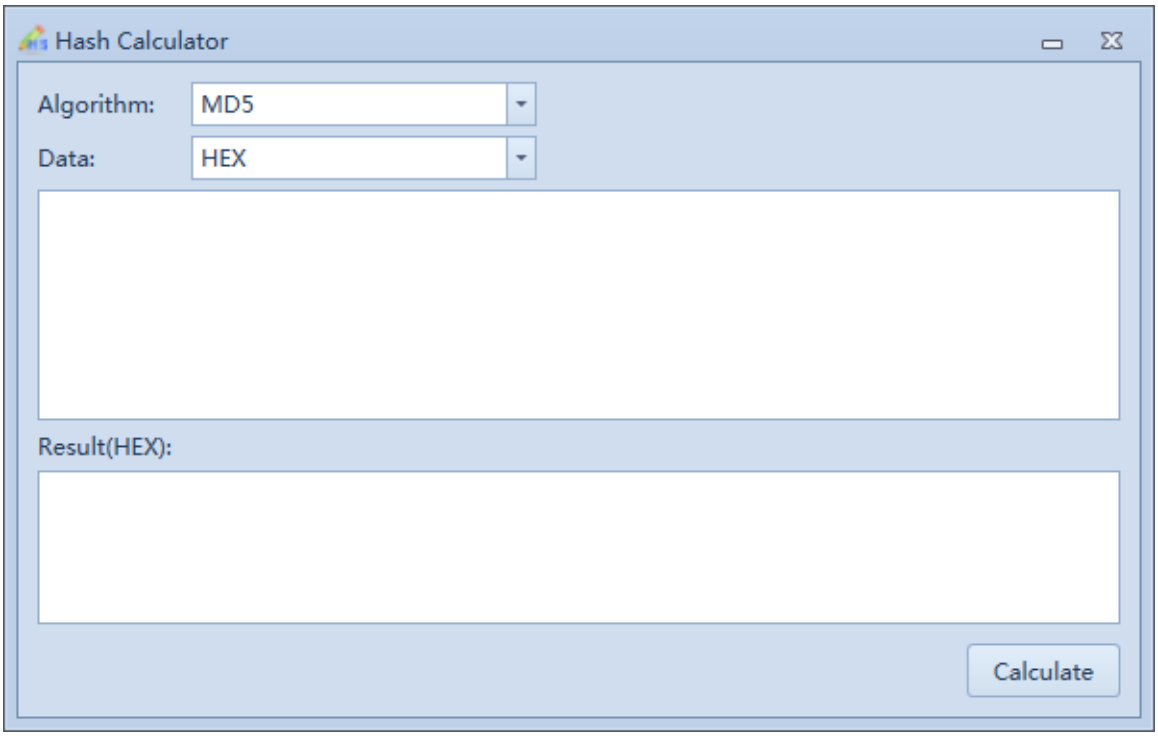
Main UI:



6.4 Hash Calculator

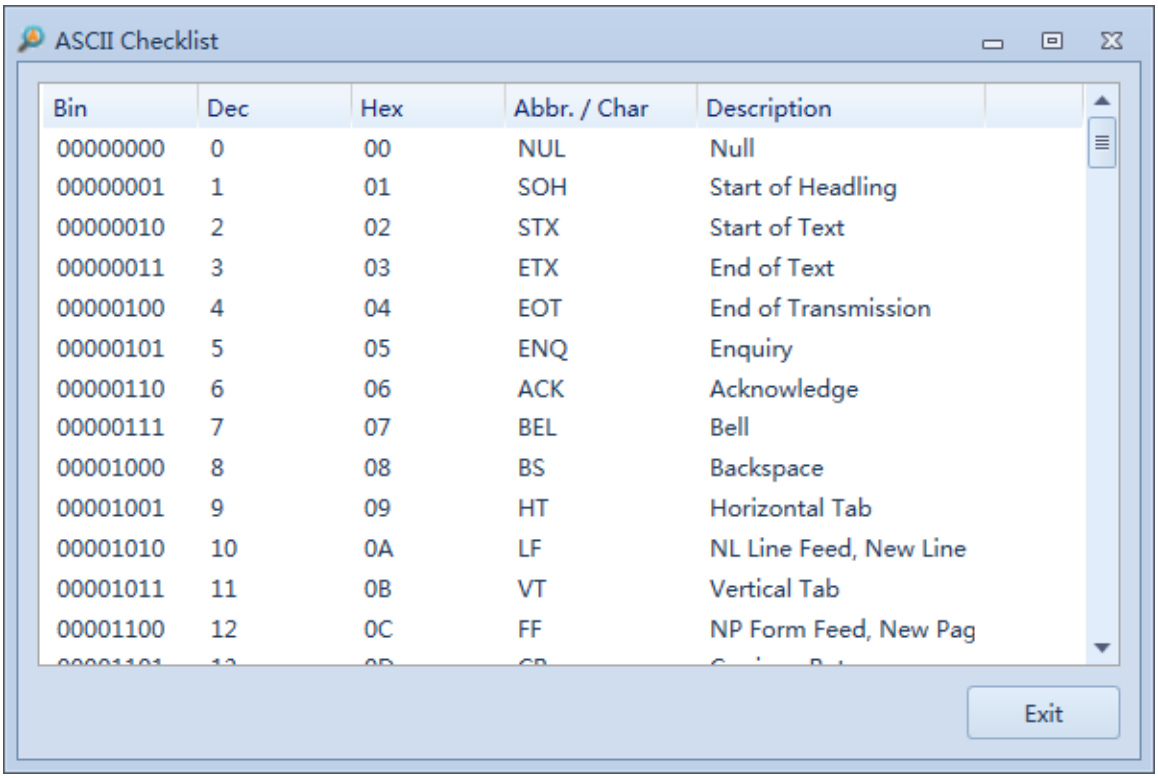
Hash calculator is a calculator for SHA1, MD5, SHA256, SHA384, SHA512 hash algorithm computing tools. Data source can be HEX data string, string (character encoding type is decided by the global encoding settings) and file.

Main UI:



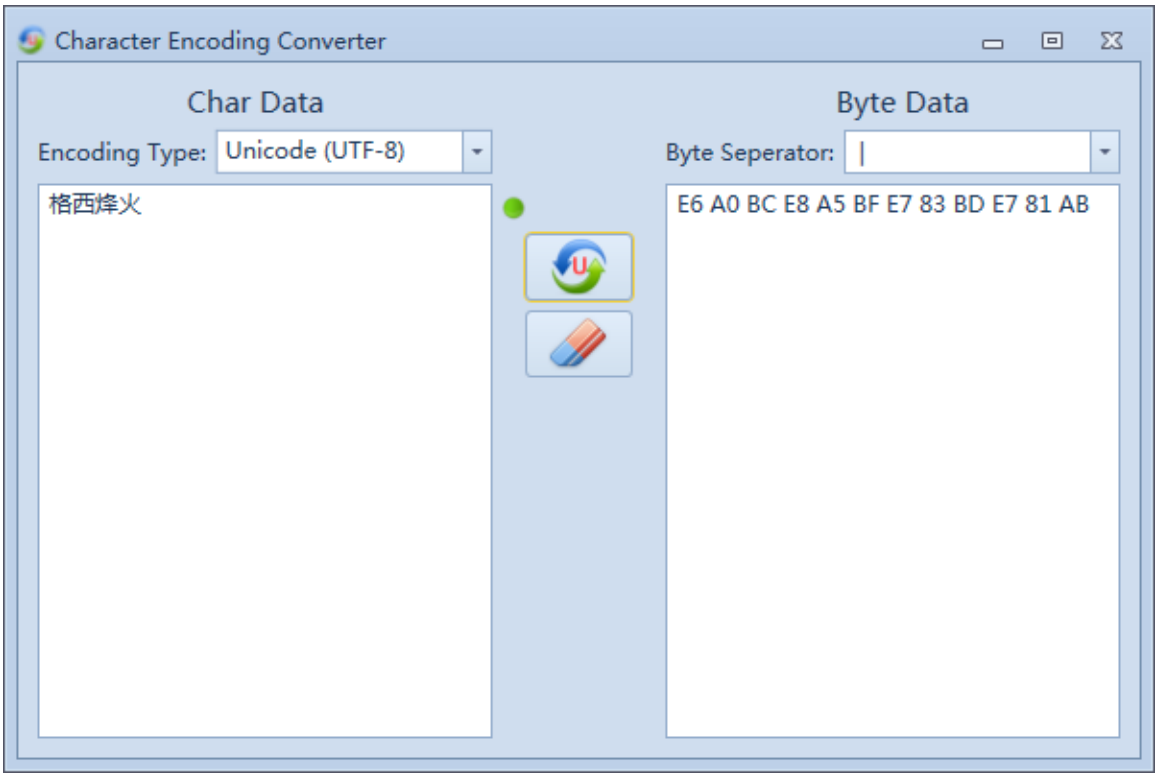
6.5 ASCII Checklist

Main UI:



6.6 Character Encoding Converter

Main UI:

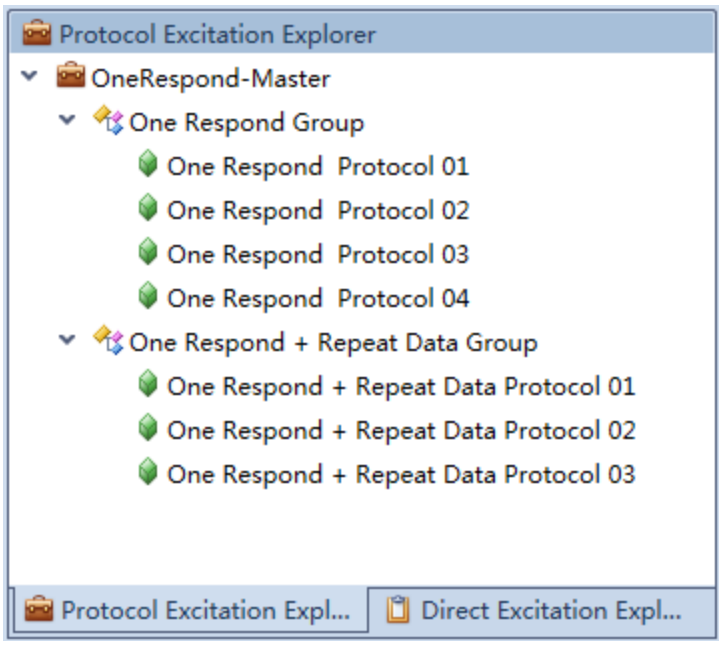


7. Application Skills

7.1 Classify Protocol Excitation Project Items

For a test project which contains many protocols, it can be classified into a tree structure, and it is more convenient to manage.

For example, in a communication protocol test, the protocol is usually organized as a basic protocol set, as a common protocol library, and then to create a specific set of test cases for different test cases.



7.2 Run Multiple Software Instances

The software supports multiple instances of software and can meet the needs of different environment and users.

Main application scenarios:

- Parallel test. As long as enough of the configuration and performance of computer, you can run multiple instances of software at the same time, each instance of software testing a communication interface, while detecting a plurality of external devices. This can reduce test cost.
- Copy and paste excitation item. This software not only supports the function of copying and pasting, but also supports the copy and paste function from one software to another.

8. FAQ

8.1 While doing “Feedback” or “Register”, Why the Unknown error (0x80041002) occurs?

Answer: In the case of unable to contact the external network server to carry out feedback and registration operation, the software will try to start the Email program to send Email, if the machine does not have Email client program, then this error occurs.

8.2 When the protocol is excited, the slave party has issued the right frame, why does the master party return failure?

Answer: One reason is that the timeout of the active incentive protocol is not reasonable. The failure time of the protocol can be appropriately extended.