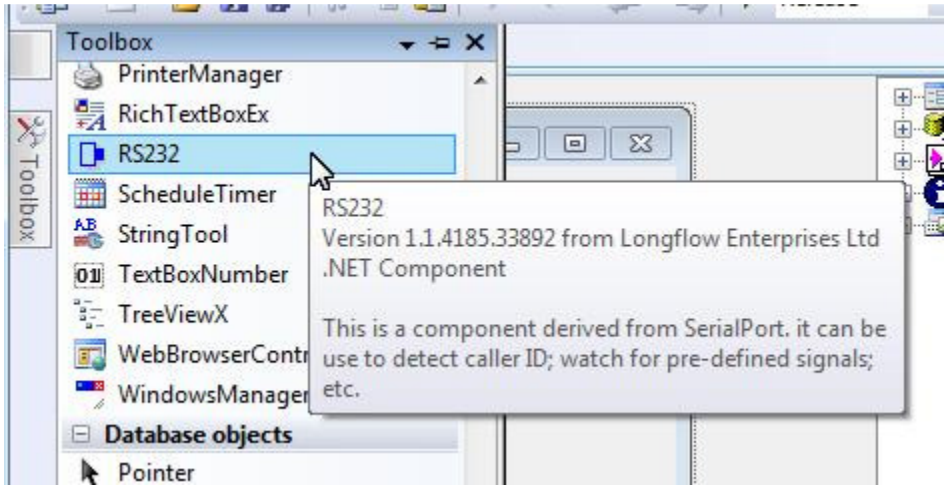
# RS232 Sample

## Contents

Use RS232 component ..... 1
Processing RS232 Component Events ..... 2
Get Caller ID ..... 14
Test without port connection ..... 16
Catch specific signals ..... 20

## Use RS232 component

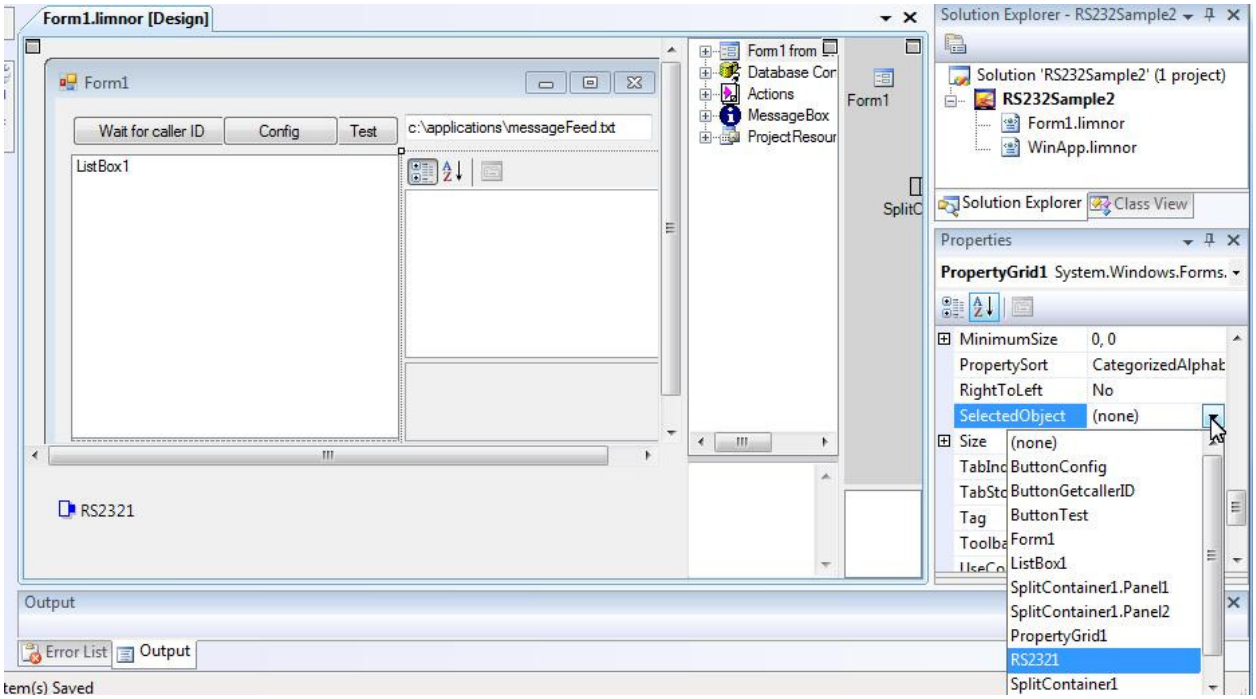
Drop a RS232 component to a form:




RS232 is a component derived from .Net SerialPort component. For a complete documentation of SerialPort component, see http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx

This sample shows some features of RS232 added to the SerialPort component.

This sample uses a list box to show events generated from the RS232 component. A PropertyGrid is used to allow the configuration of the RS232 component at runtime:




## Processing RS232 Component Events

RS232 component generates following events:

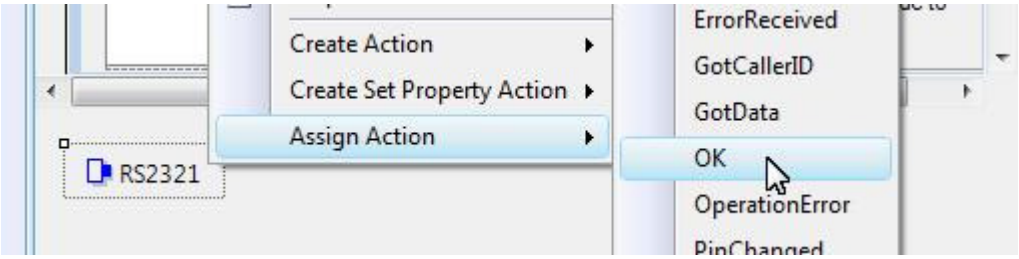
- OK – an OK signal is received after sending a command to the port.
- Ring – a telephone call is detected.
- GotCallerID – caller ID is detected after executing a WaitForCall action
- GotData – data is available. ReadDataCache action can be used to read the data.
- OperationError – it occurs when some error occurs during execution of actions. Error property is the error message.

The following events are generated from the serial port:

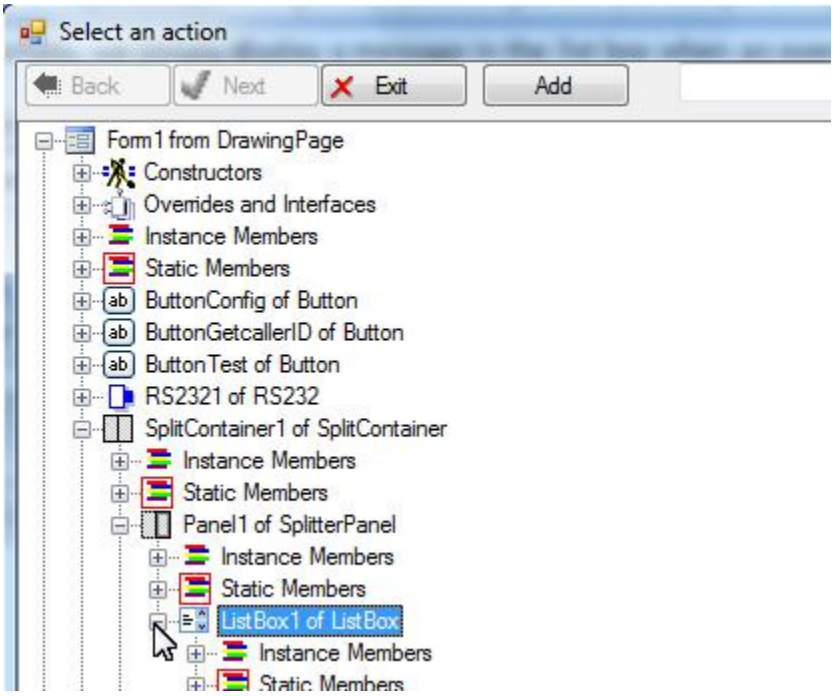
- DataReceived – it occurs when data is available from the serial port.
- ErrorReceived – it occus when an error occurs in the serial port.

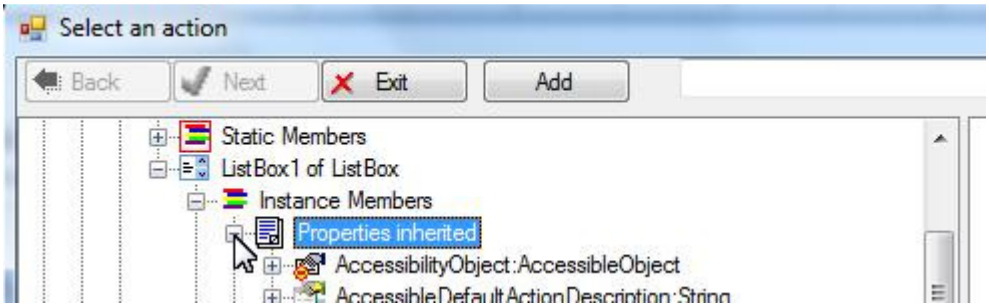
For this sample, we simply display a message in the list box when an event occurs.

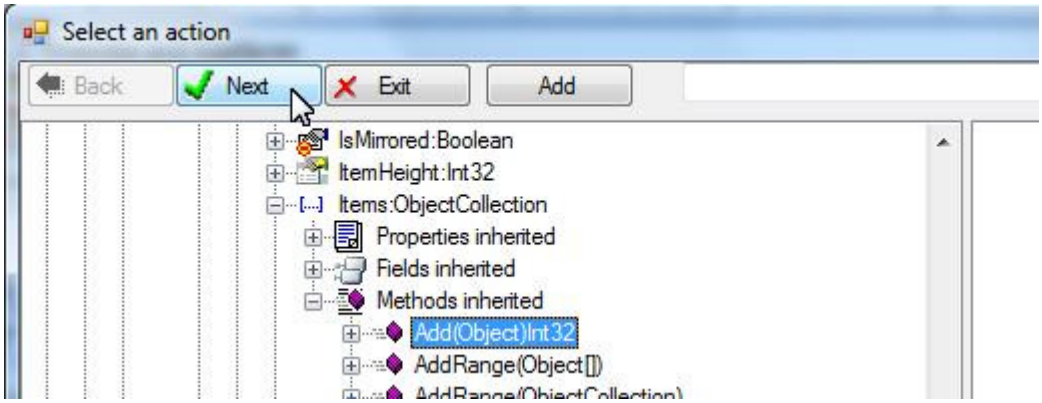
Right-click the RS232 component; choose “Assign Action”; choose “OK” event:



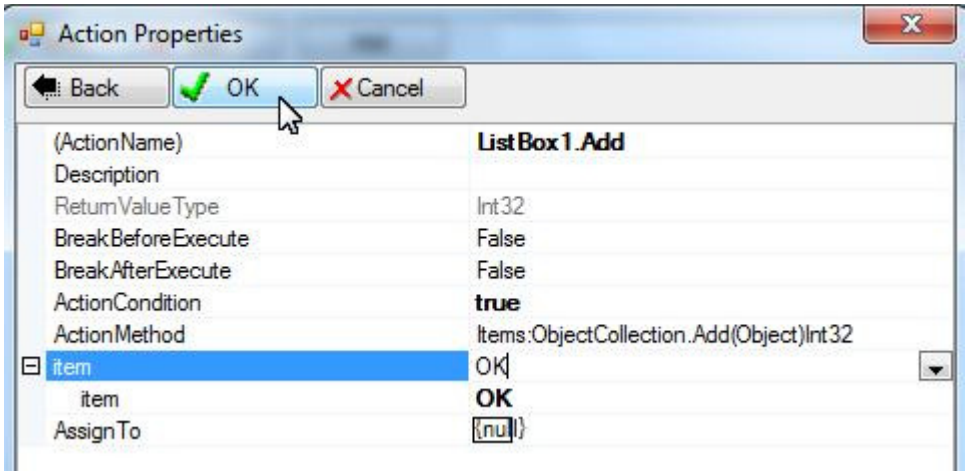

Select Add method of the Items property of the list box; click Next.



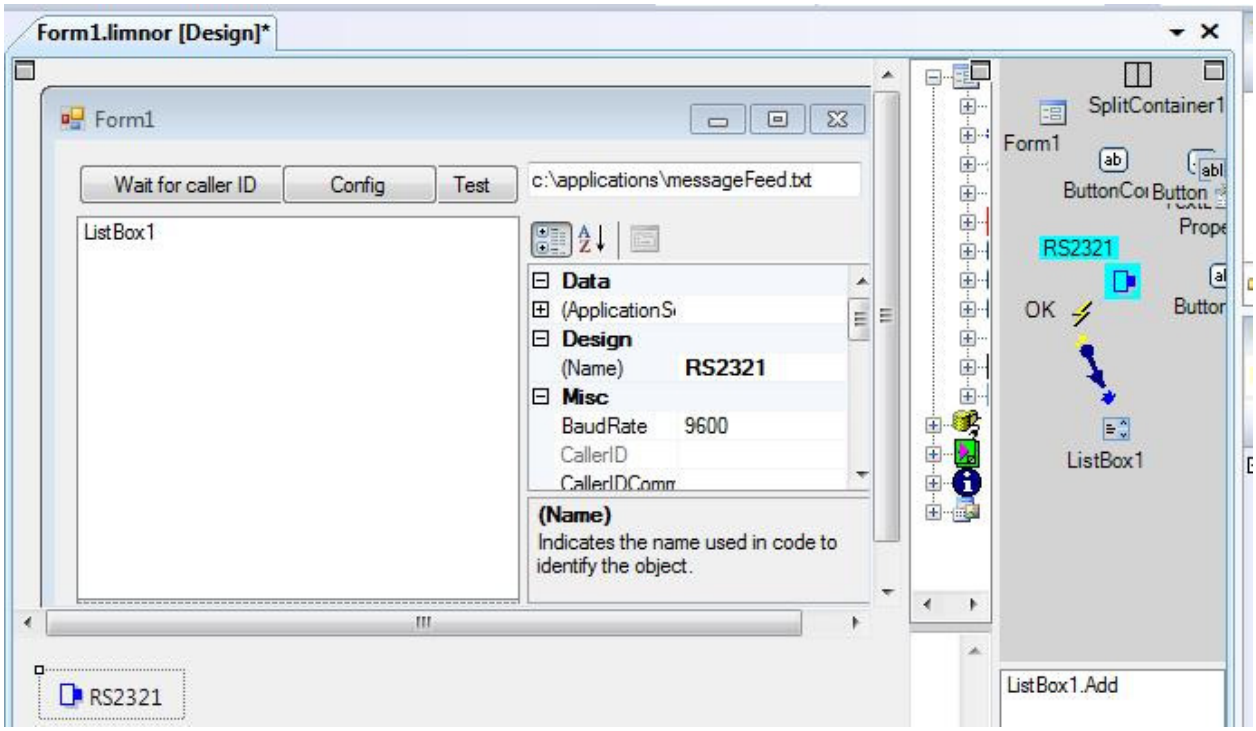





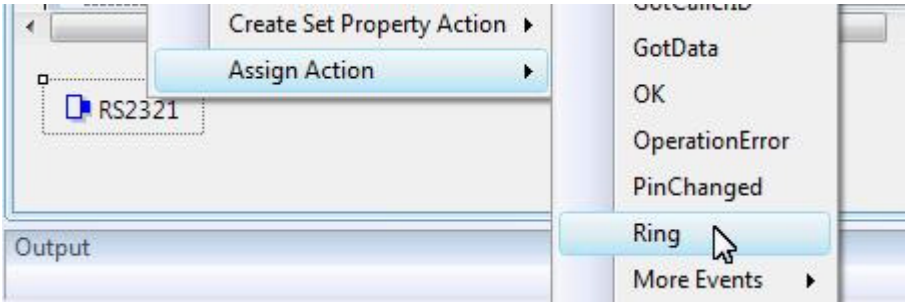

Enter OK for the "item" so that an "OK" text will be added to the list box; click OK button.

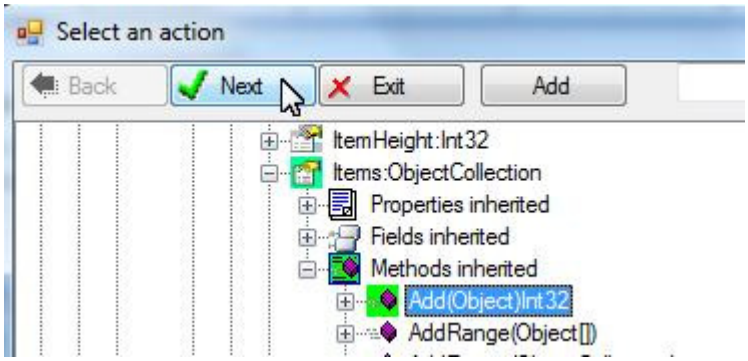


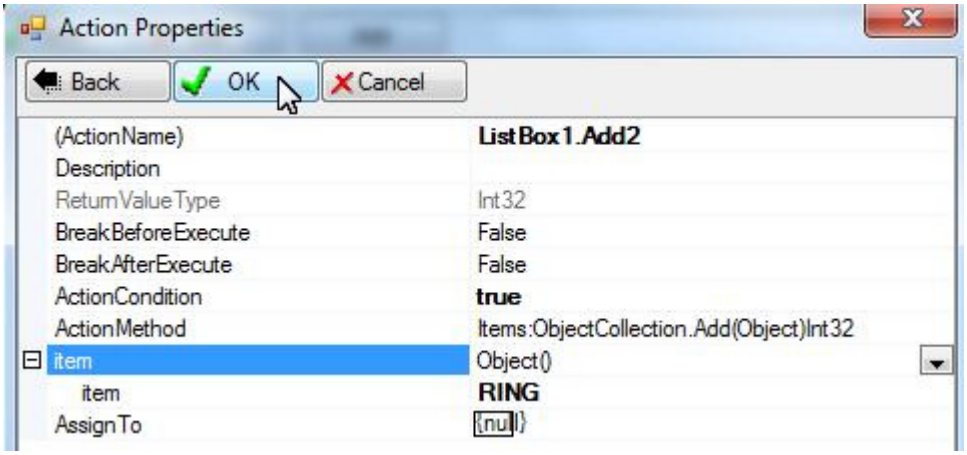

This action is created and assigned to the OK event:



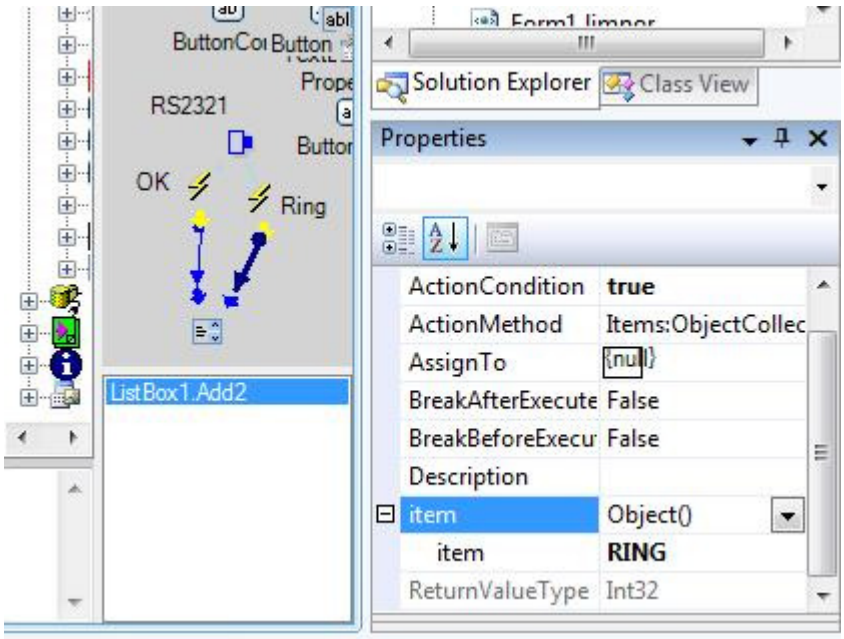

Also create an action to show "RING" in the list box when Ring event occurs:



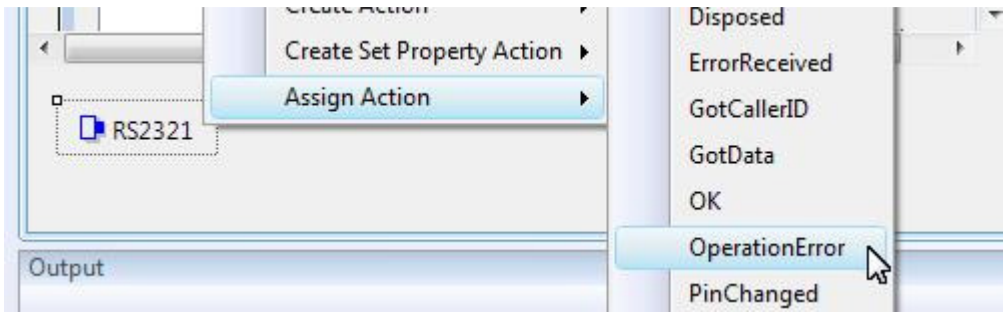



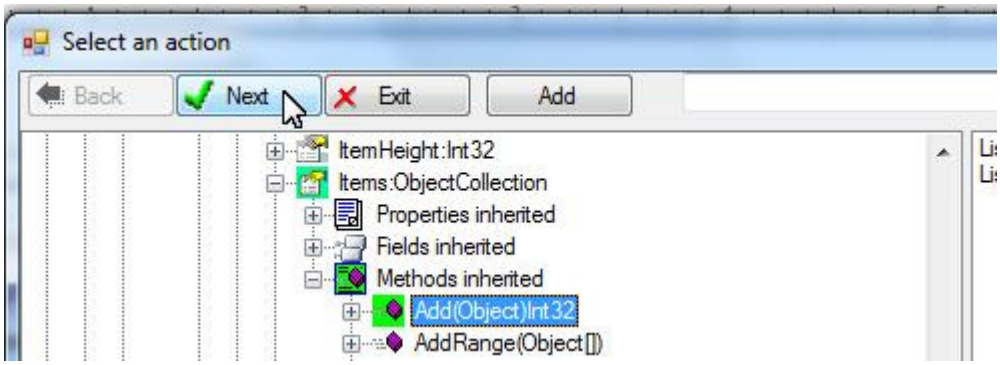


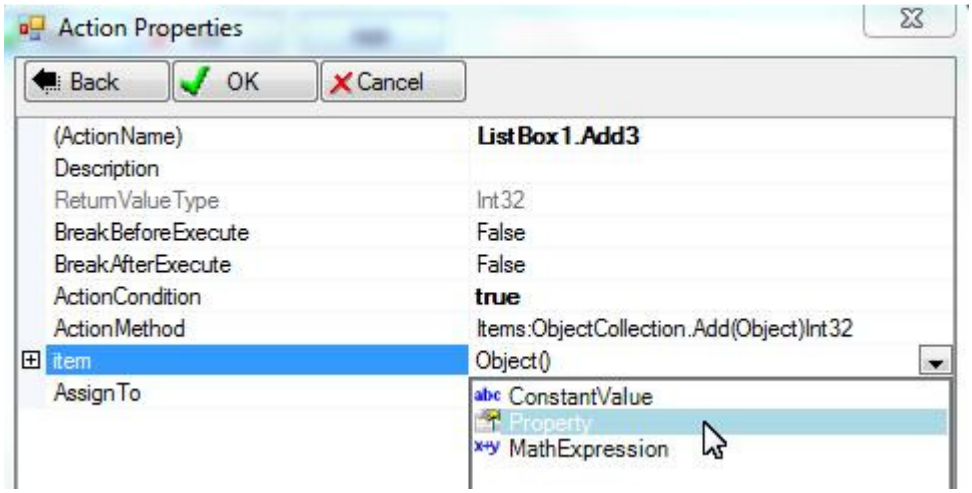

The action is created and assigned to Ring:

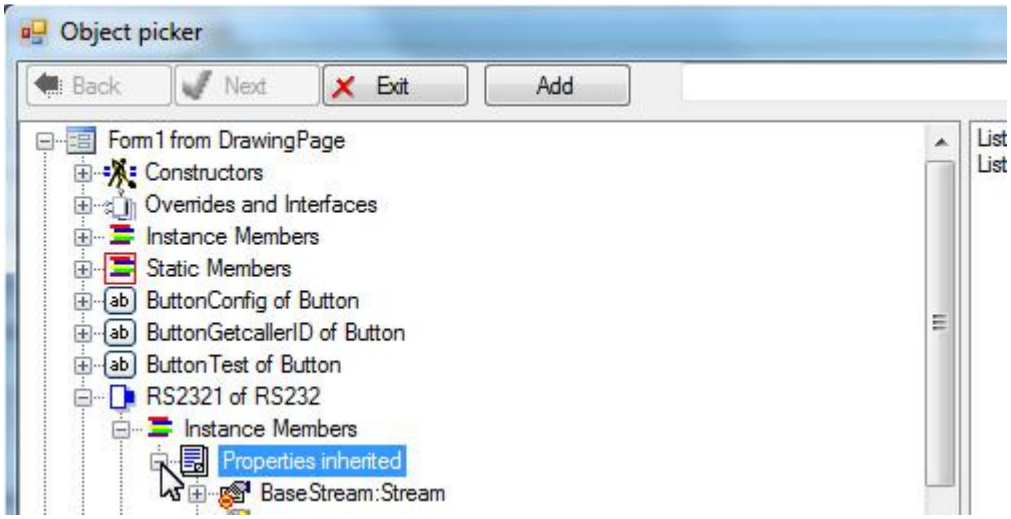


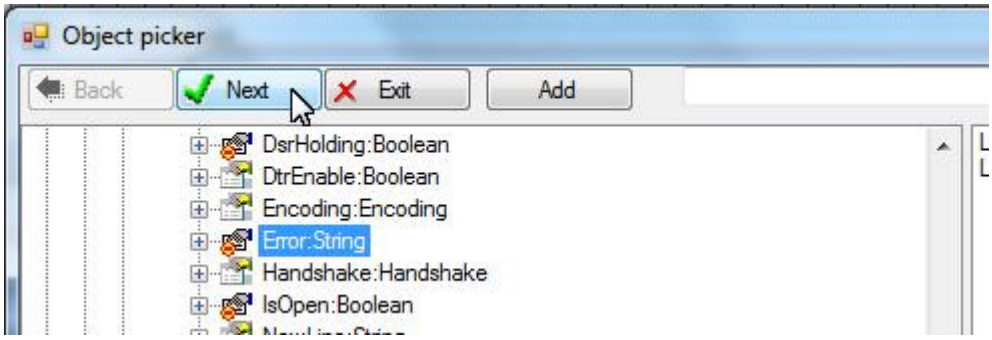

We may show Error property of the RS232 component when OperationError occurs:

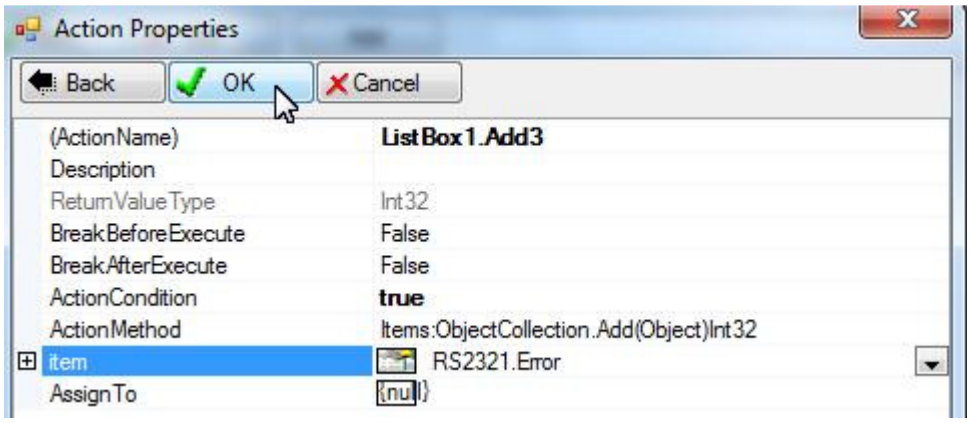



Select an action
Back
Next
Exit
Add
ItemHeight:Int32
Items:ObjectCollection
Properties inherited
Fields inherited
Methods inherited
Add(Object)Int32
AddRange(Object[])

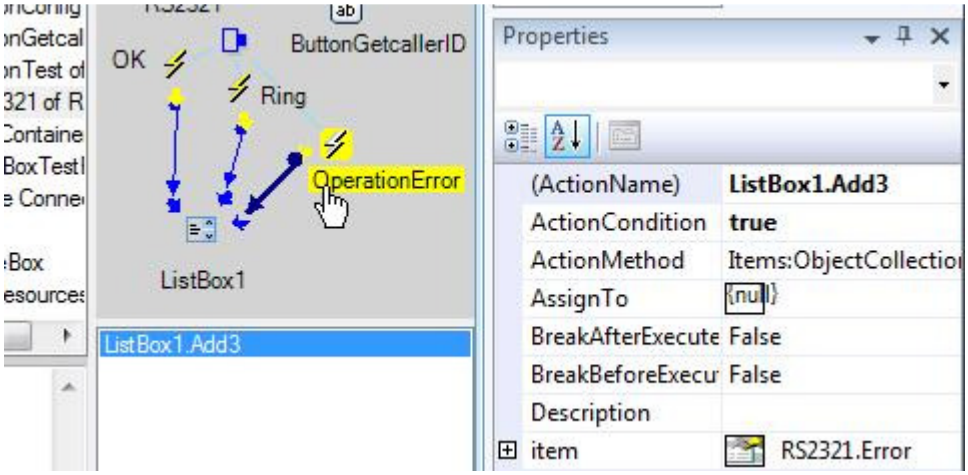

Action Properties
Back
OK
Cancel
(ActionName)
ListBox1.Add3
Description
ReturnValueType
Int32
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Items:ObjectCollection.Add(Object)Int32
item
Object()
AssignTo
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
ButtonConfig of Button
ButtonGetcallerID of Button
ButtonTest of Button
RS2321 of RS232
Instance Members
Properties inherited
BaseStream:Stream

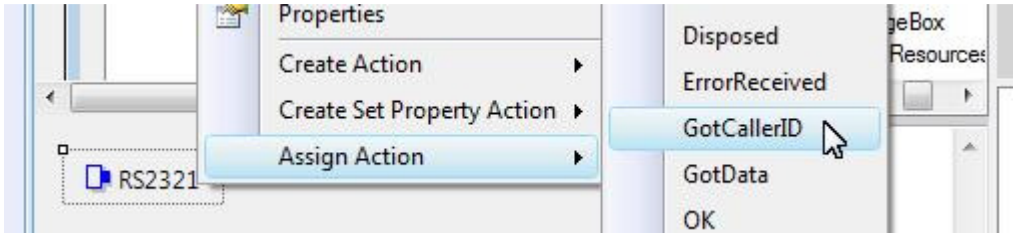


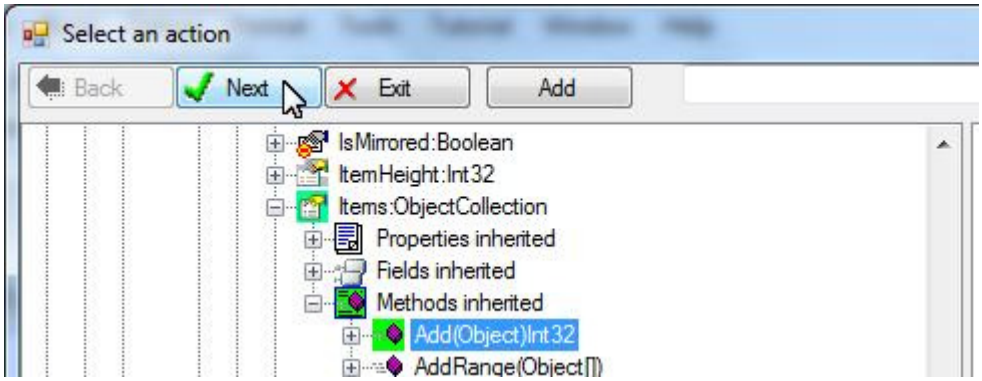



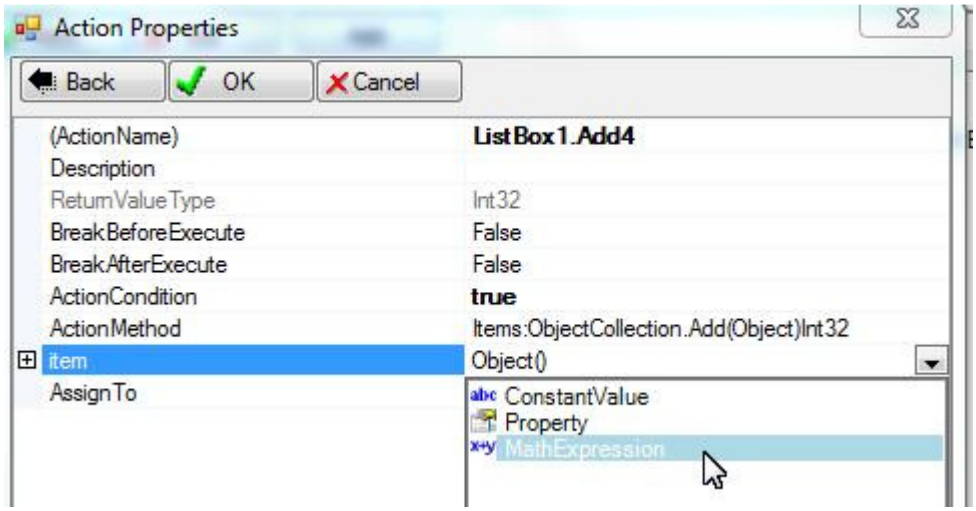

The action is created and assigned to the event:

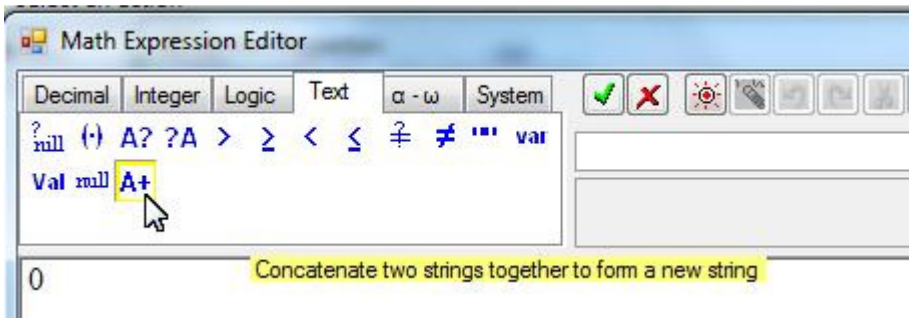


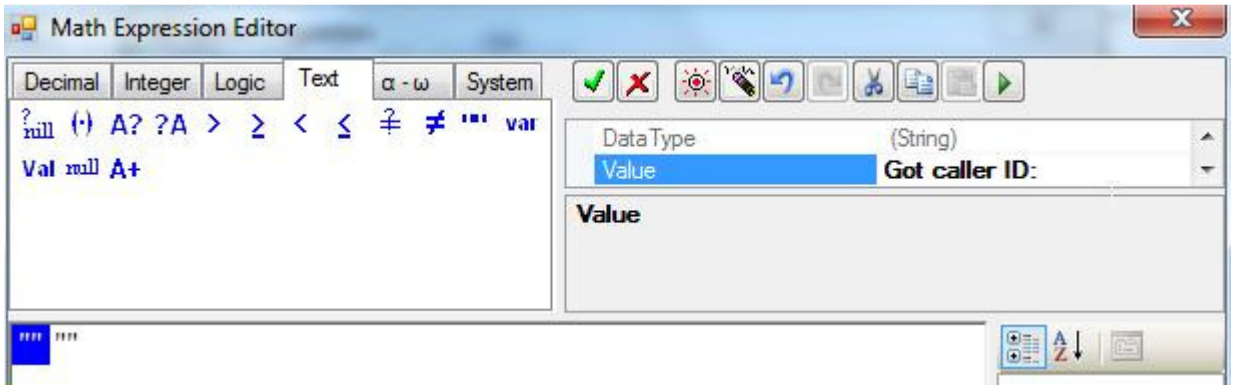

We may show Caller ID when GotCallerID event occurs:

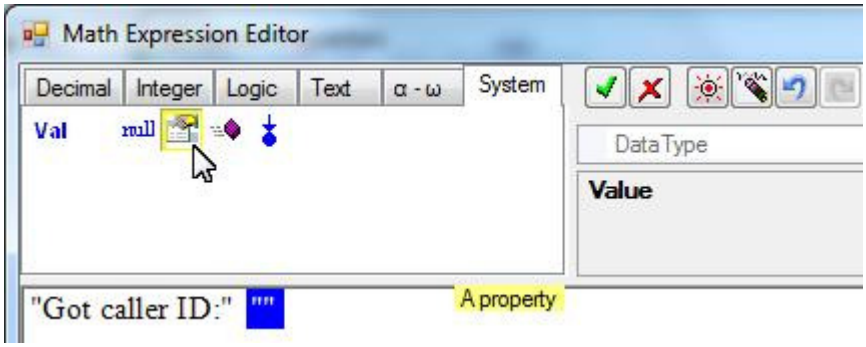


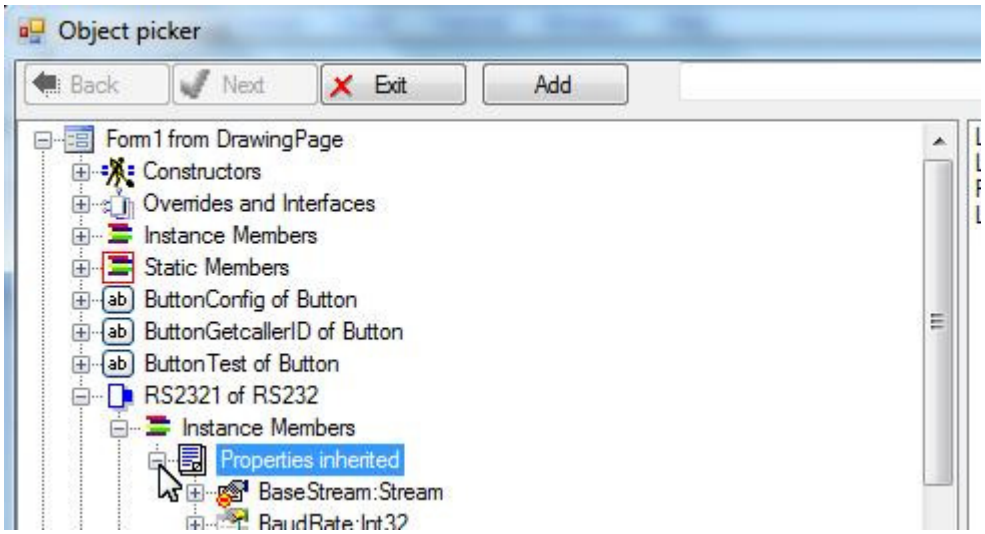

Select an action
Back
Next
Exit
Add
IsMirrored:Boolean
ItemHeight:Int32
Items:ObjectCollection
Properties inherited
Fields inherited
Methods inherited
Add(Object)Int32
AddRange(Object[])

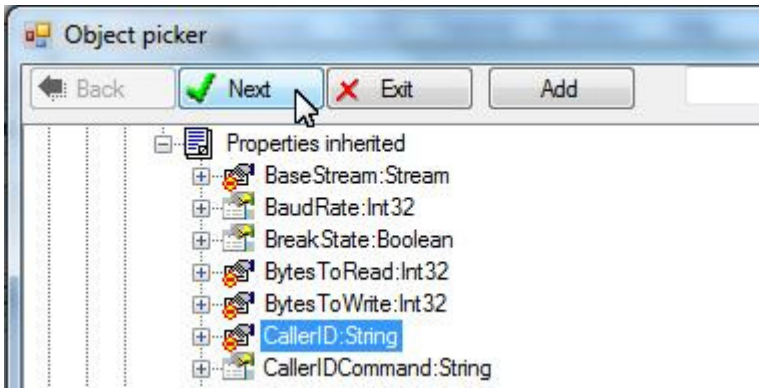

Action Properties
Back
OK
Cancel
(ActionName)
ListBox1.Add4
Description
ReturnValueType
Int32
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Items:ObjectCollection.Add(Object)Int32
item
Object()
AssignTo
ConstantValue
Property
MathExpression

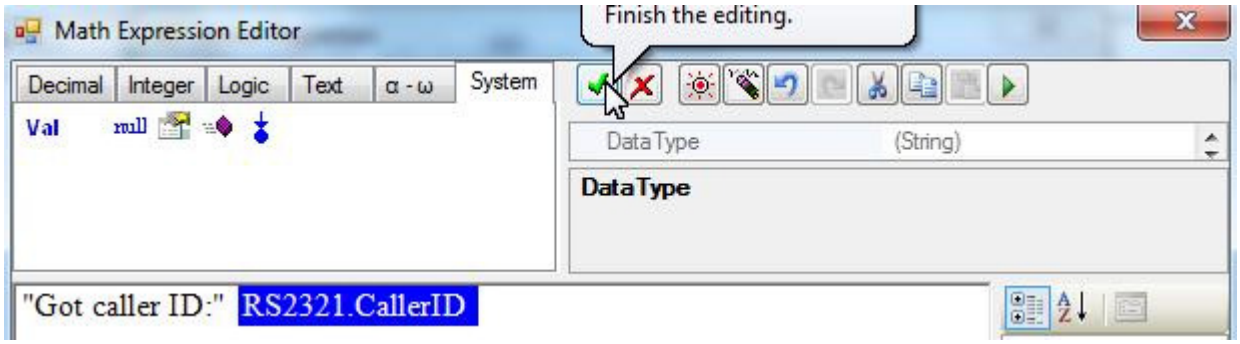

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
A+
0
Concatenate two strings together to form a new string

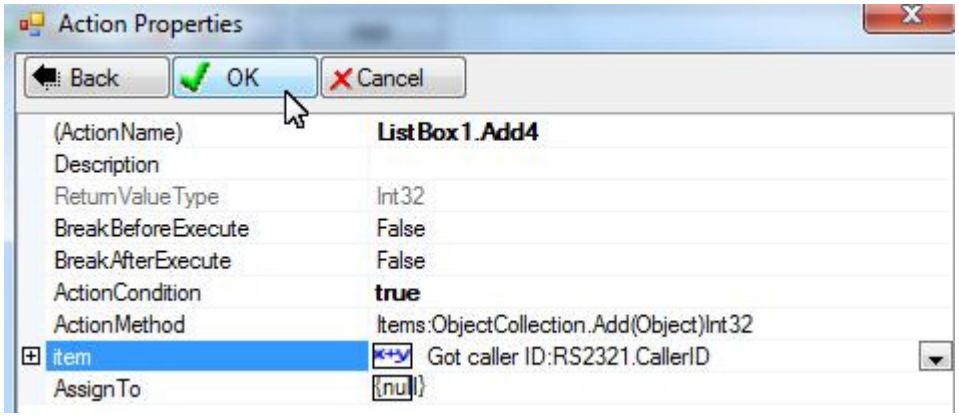

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
A+
DataType
(String)
Value
Got caller ID:
Value


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
Value
"Got caller ID:"
""
A property

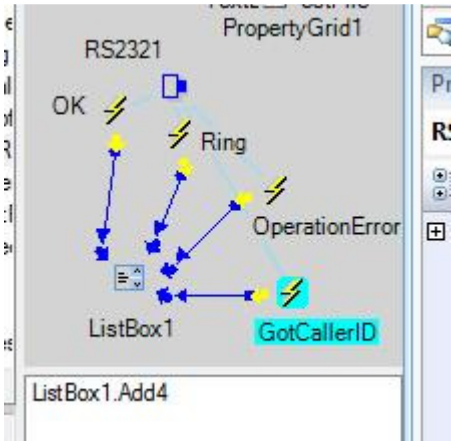

Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
ButtonConfig of Button
ButtonGetcallerID of Button
ButtonTest of Button
RS2321 of RS232
Instance Members
Properties inherited
BaseStream:Stream


Object picker
Back
Next
Exit
Add
Properties inherited
BaseStream:Stream
BaudRate:Int32
BreakState:Boolean
BytesToRead:Int32
BytesToWrite:Int32
CallerID:String
CallerIDCommand:String

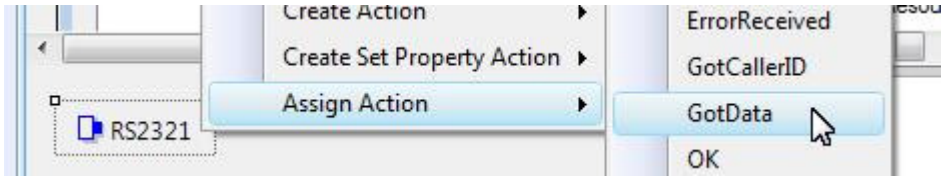

Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
DataType
"Got caller ID:"
RS2321.CallerID

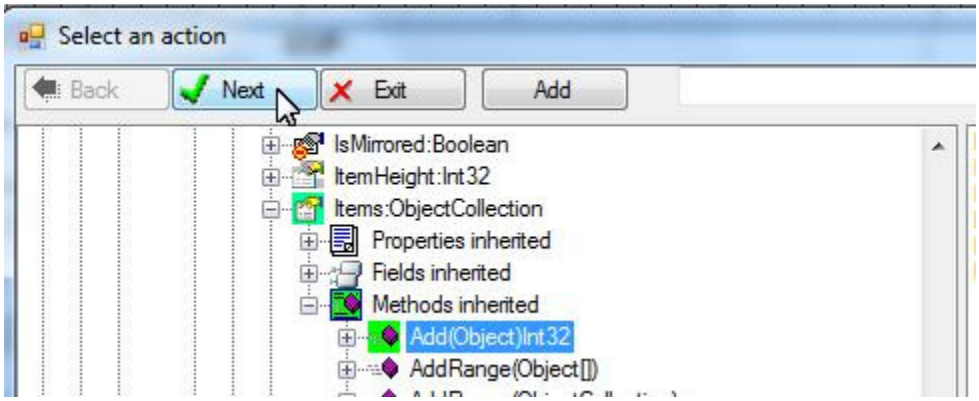


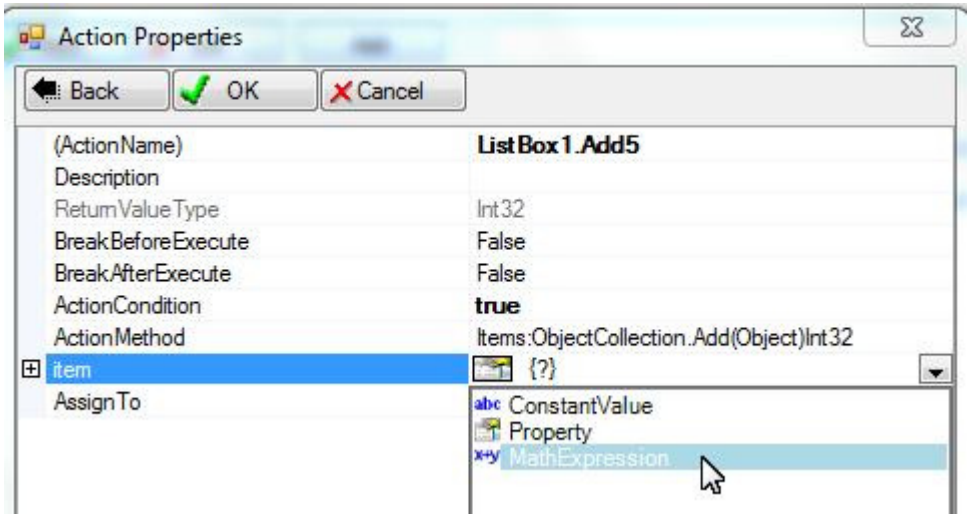

The action is created and assigned to the event:

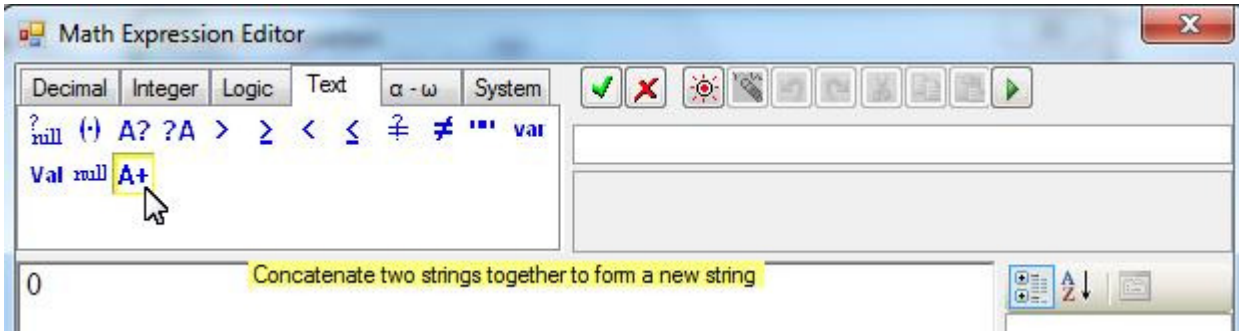


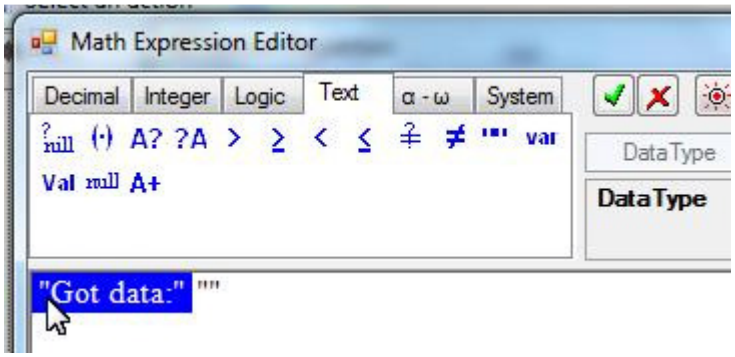

We may display the data received when GotData event occurs:

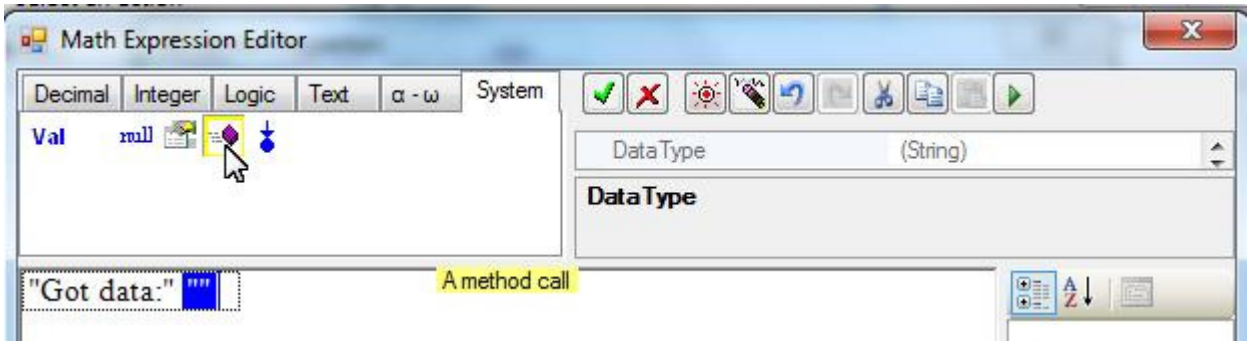


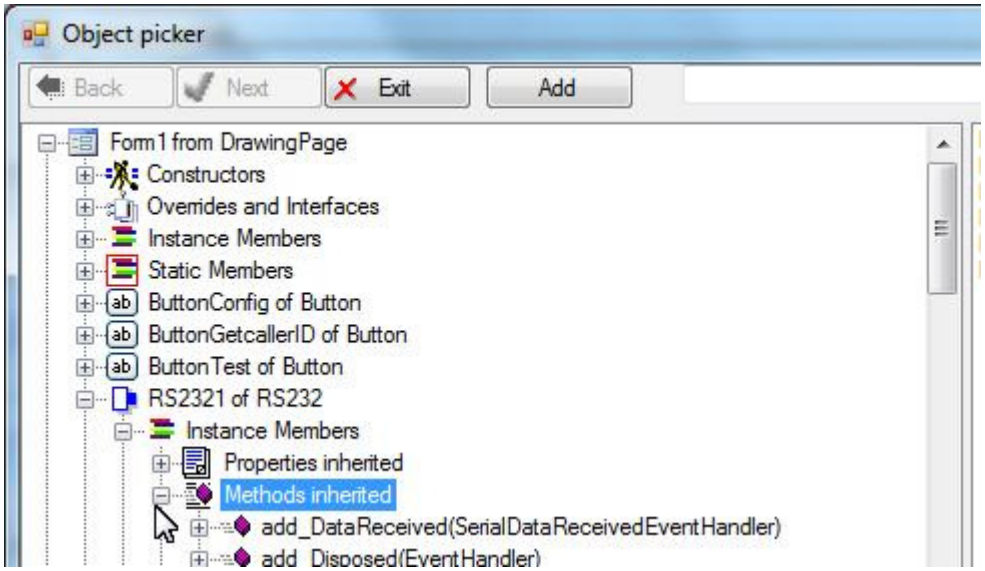


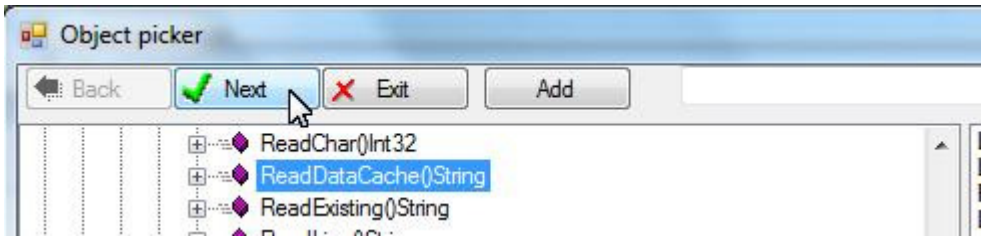



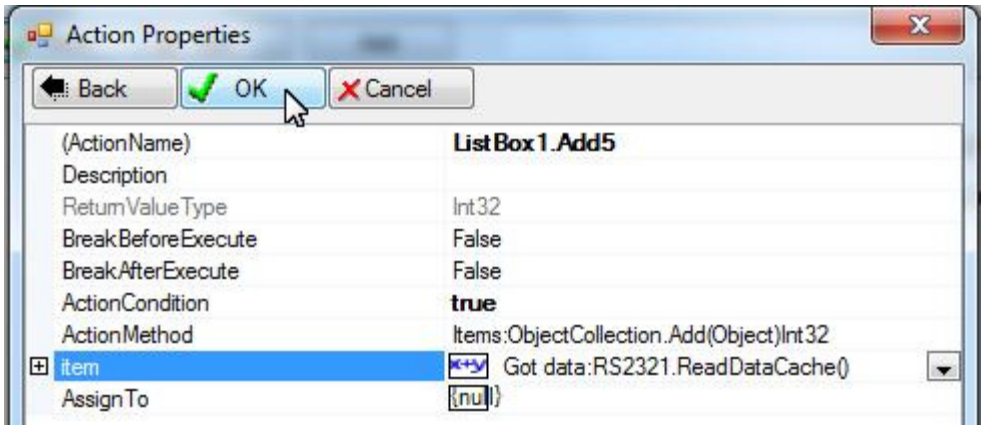




Execute ReadDataCache:

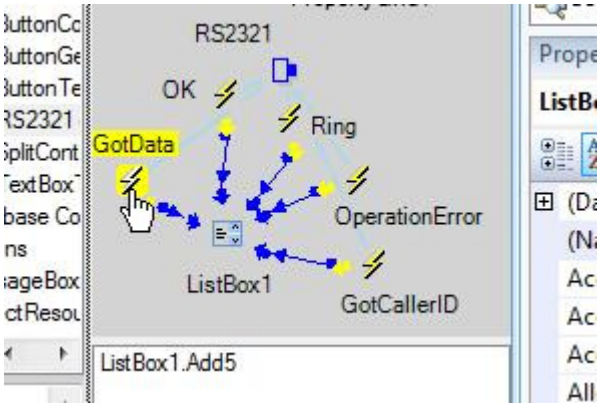







The action is created and assigned to the event:




## Get Caller ID

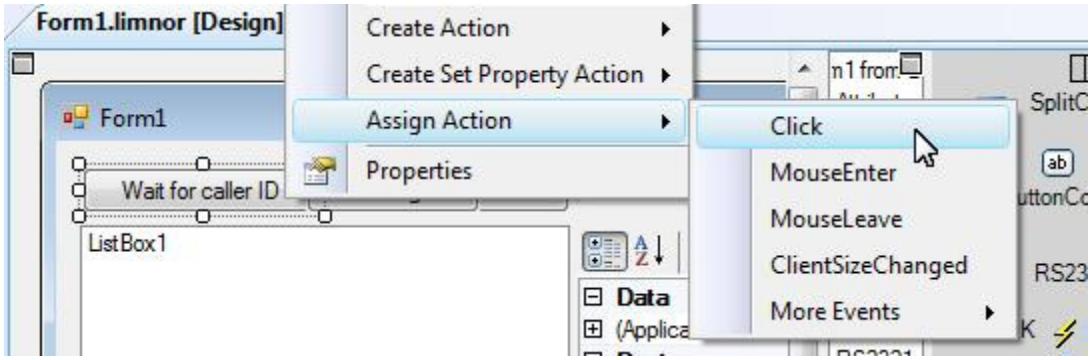
The CallerIDCommand property must match your modem's command. If you lost your modem's manual, you may try some common commands such as

AT#CID=2
AT%CCID=1
AT+VCID=1
AT%CCID=2
AT#CC1
AT*ID1

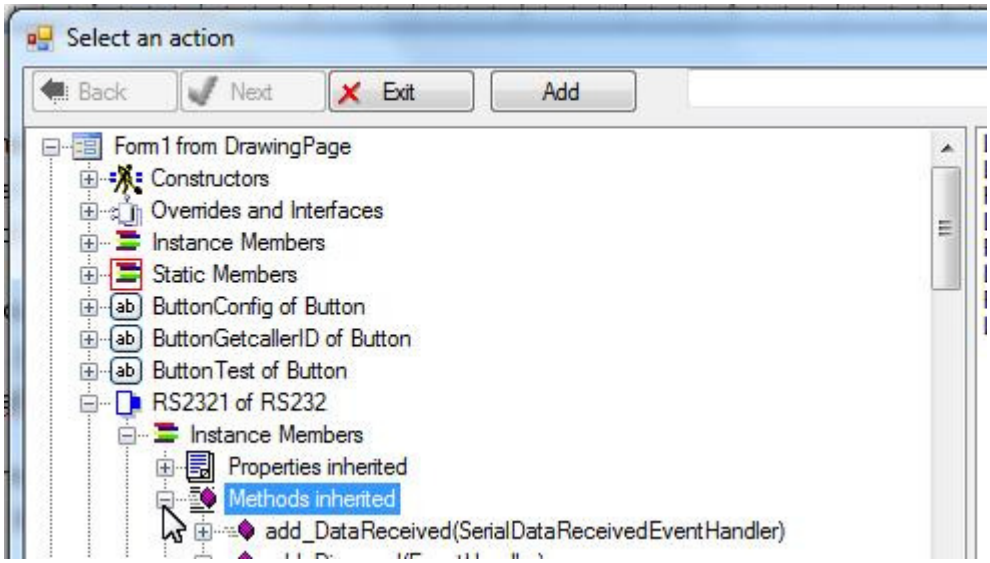
Try them one by one and execute WaitForCall action and watch for OK and ErrorReceived events. If you get two OK events then the command is correct. If you get one OK event and one ErrorReceived event then the command is not for your modem.

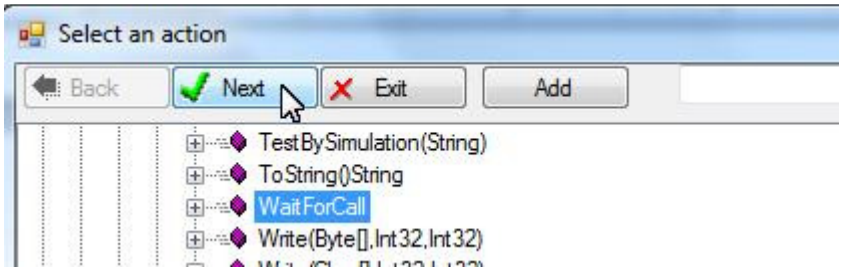
After executing WaitForCall, when a phone call is detected, Ring event occurs. When a caller ID is detected, GotCallerID event occurs. The CallerID property is the caller ID detected. Your application may use GotCallerID event and CallerID property to do things according to your business logic.

In this sample we display the caller ID in the list box when it is available. We want to execute WaitForCall when the button is clicked. Right-click the button; choose “Assign Action”; choose “Click” event:

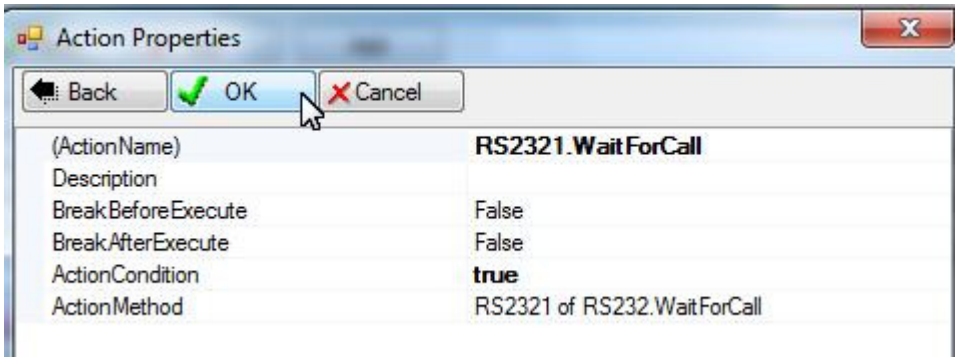



Select WaitForCall and Click Next:

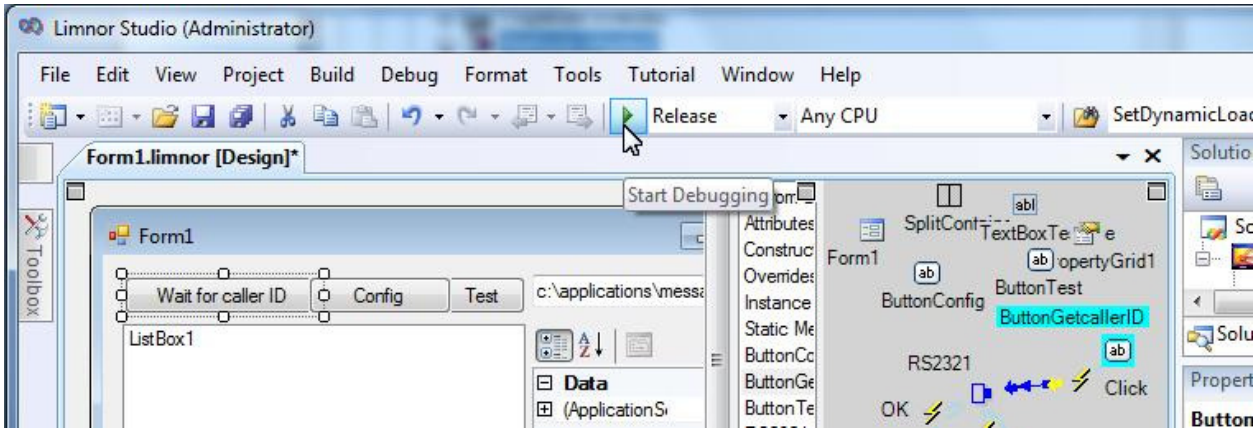


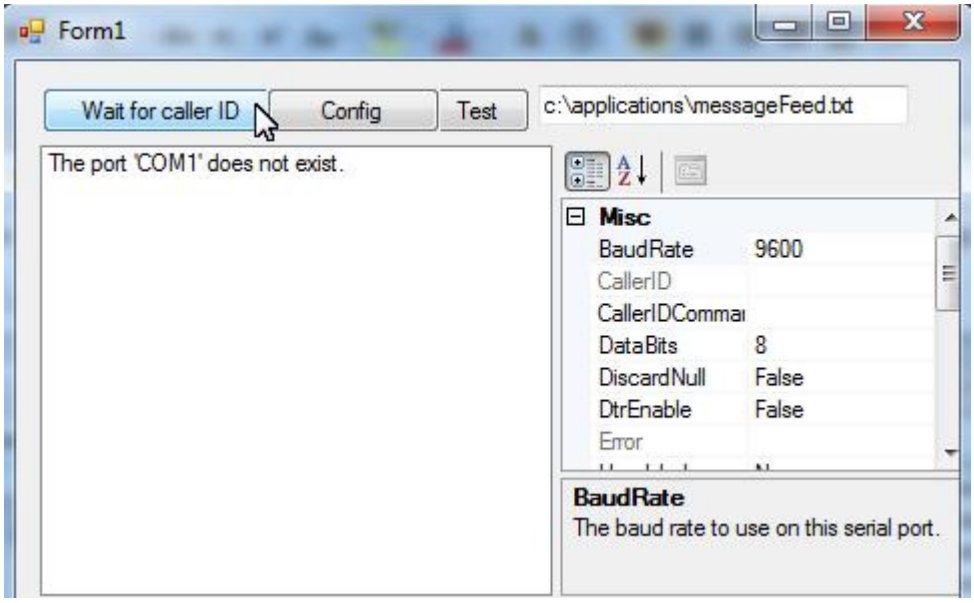

Click Ok to finish creating this action and assigning it to the button:




We may test the sample:




The form appears. Click button “Wait for caller ID”. The serial port activities are displayed in the list bx.




## Test without port connection

TestBySimulation method can be used to test signal processing without actually connecting to a RS232 device.

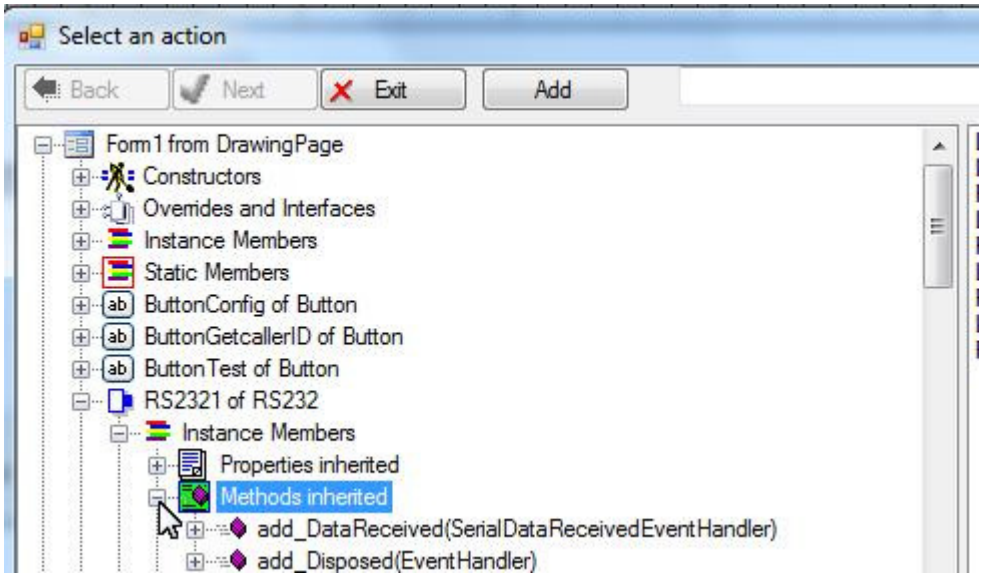
A text file can be used to simulate the port messages. When a TestBySimulation action is executed each line in the file is sent to the RS232 component as if a message from a port device.

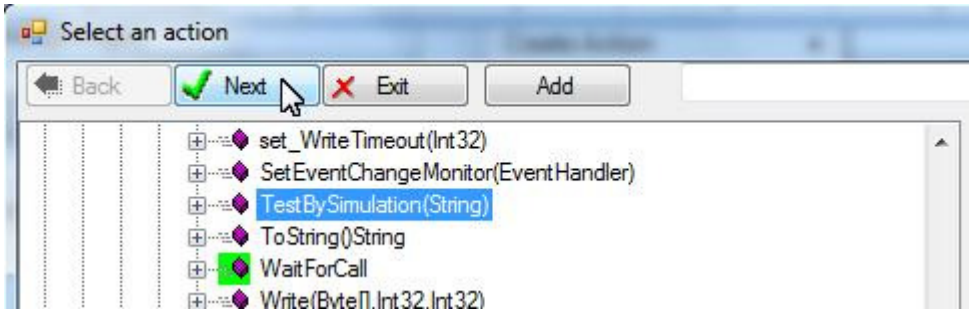
Below are the contents of a sample test file:

```
OK
RING
OK
NMBR   8001234567
my data
```

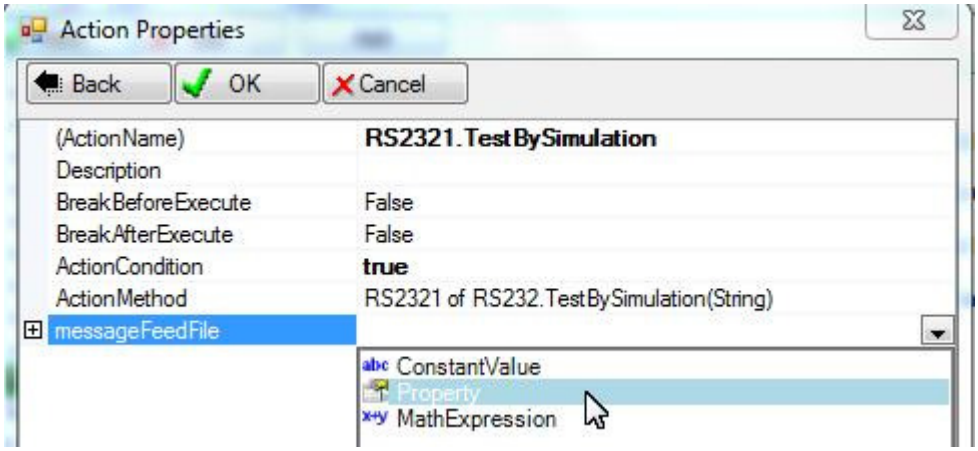
```
signal1
more data
signal2
```

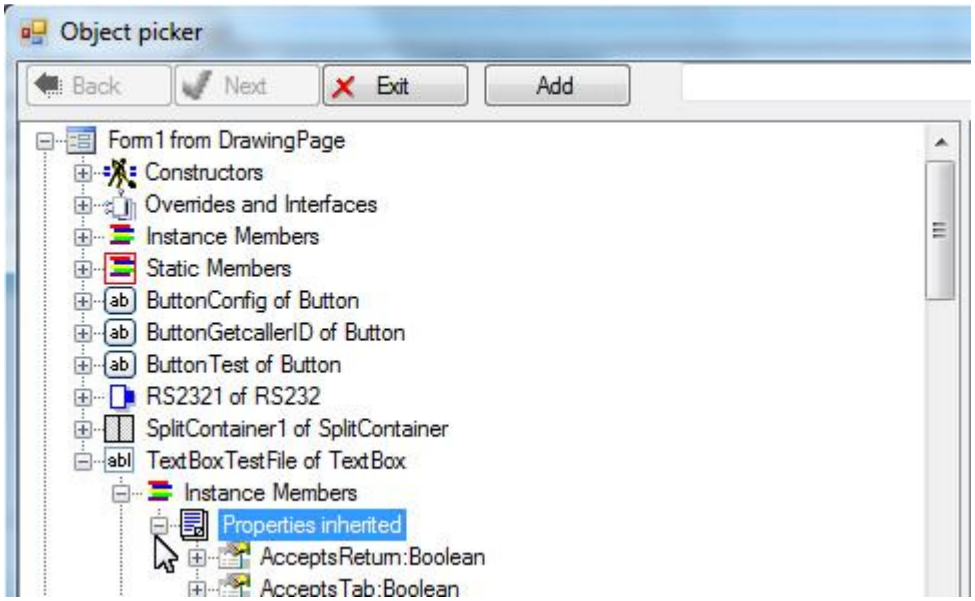
We use a button for executing TestBySimulation:

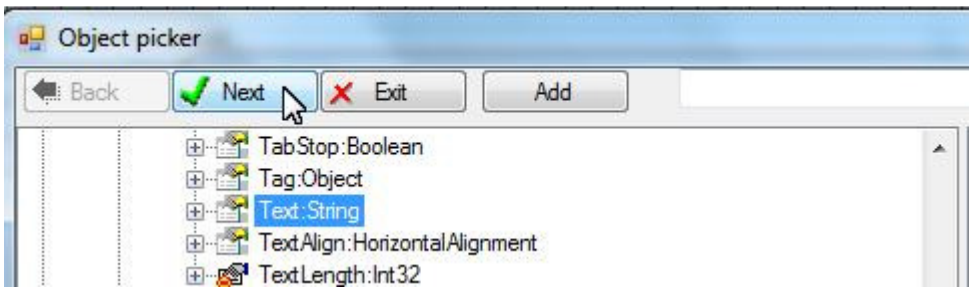



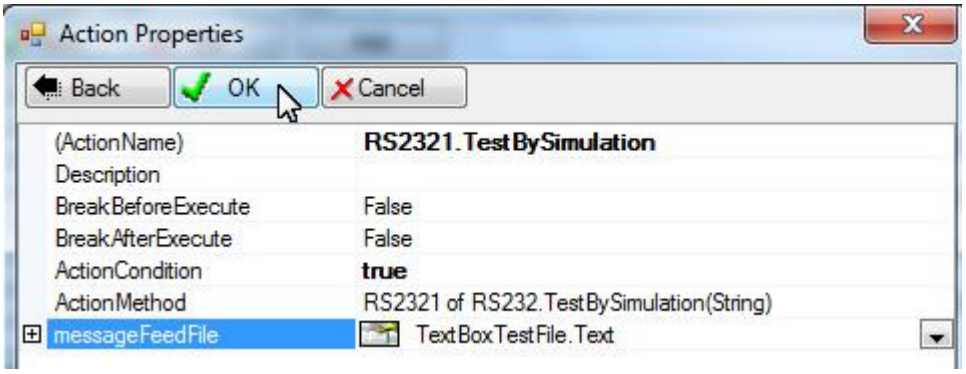




The test file name is in a text box:


Action Properties
Back
OK
Cancel
(ActionName)
RS2321.TestBySimulation
Description
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
RS2321 of RS232.TestBySimulation(String)
messageFeedFile
ConstantValue
Property
MathExpression

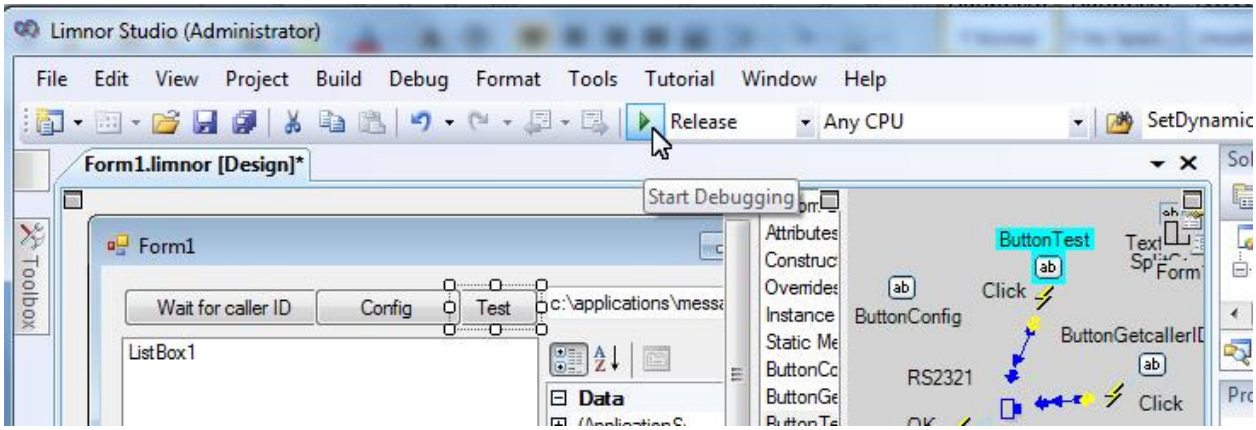

Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
ButtonConfig of Button
ButtonGetcallerID of Button
ButtonTest of Button
RS2321 of RS232
SplitContainer1 of SplitContainer
TextBoxTestFile of TextBox
Instance Members
Properties inherited
AcceptsReturn:Boolean
AcceptsTab:Boolean


Object picker
Back
Next
Exit
Add
TabStop:Boolean
Tag:Object
Text:String
TextAlign:HorizontalAlignment
TextLength:Int32

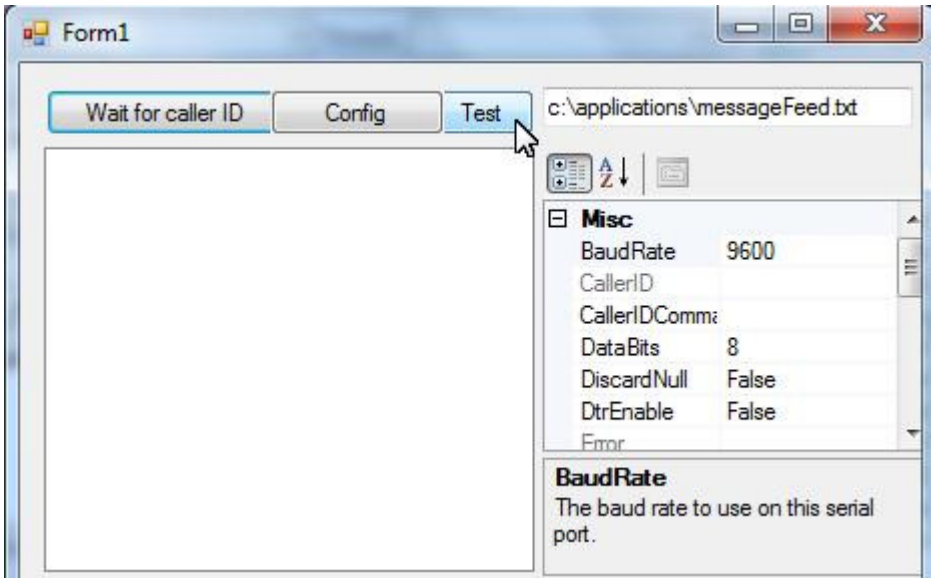



The action is created and assigned to the button.

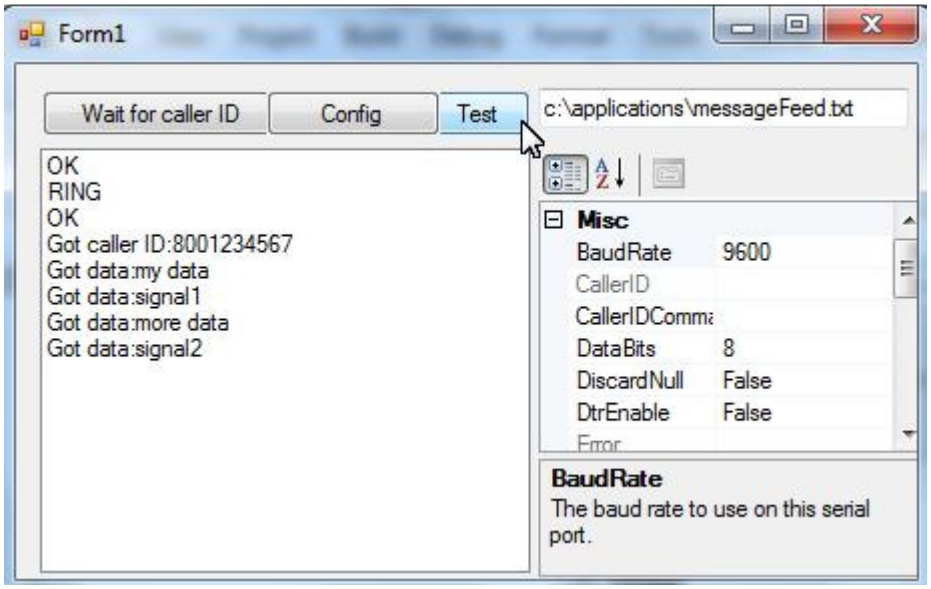
We may test the sample:




The form appears:



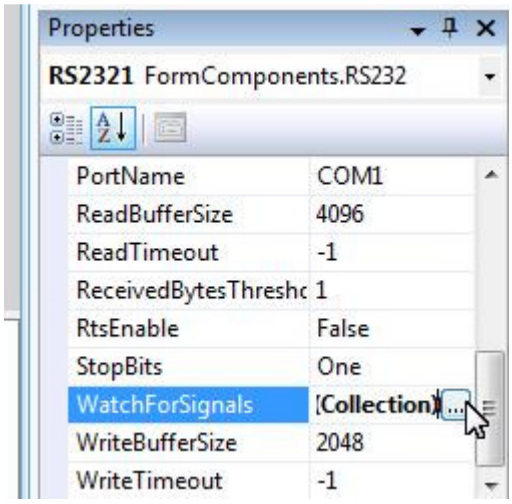
Click the Test button, the test file contents are used as port messages. The processing of the messages is displayed in the list box:



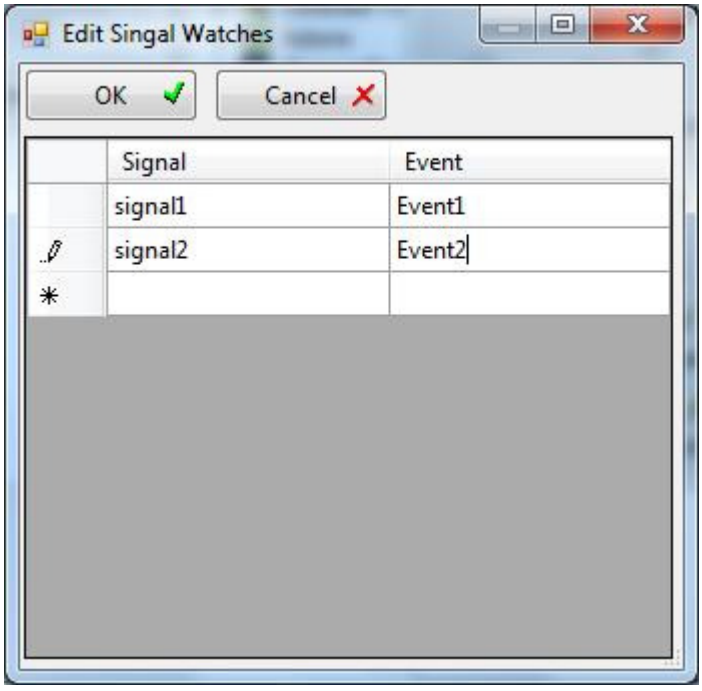

## Catch specific signals

OK, RING, ERROR, and caller ID are standard port signals. Each signal has one corresponding event for your programming purpose. You may define your own signals and events and make corresponding event processing.

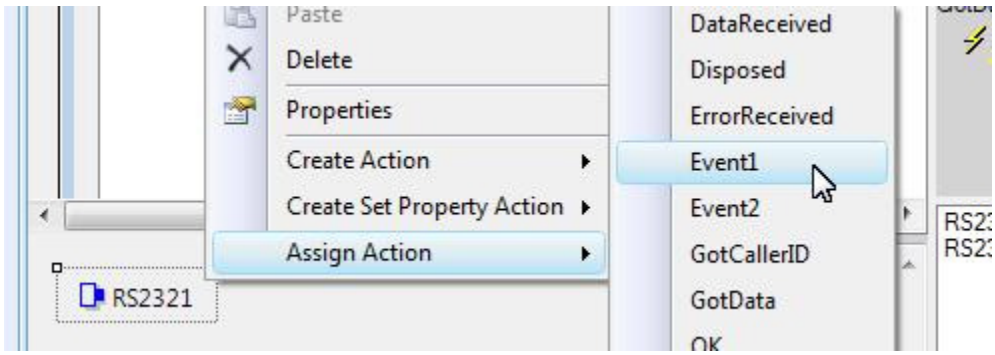
Set WatchForSignals property to define your signals and choose event names for the signals.

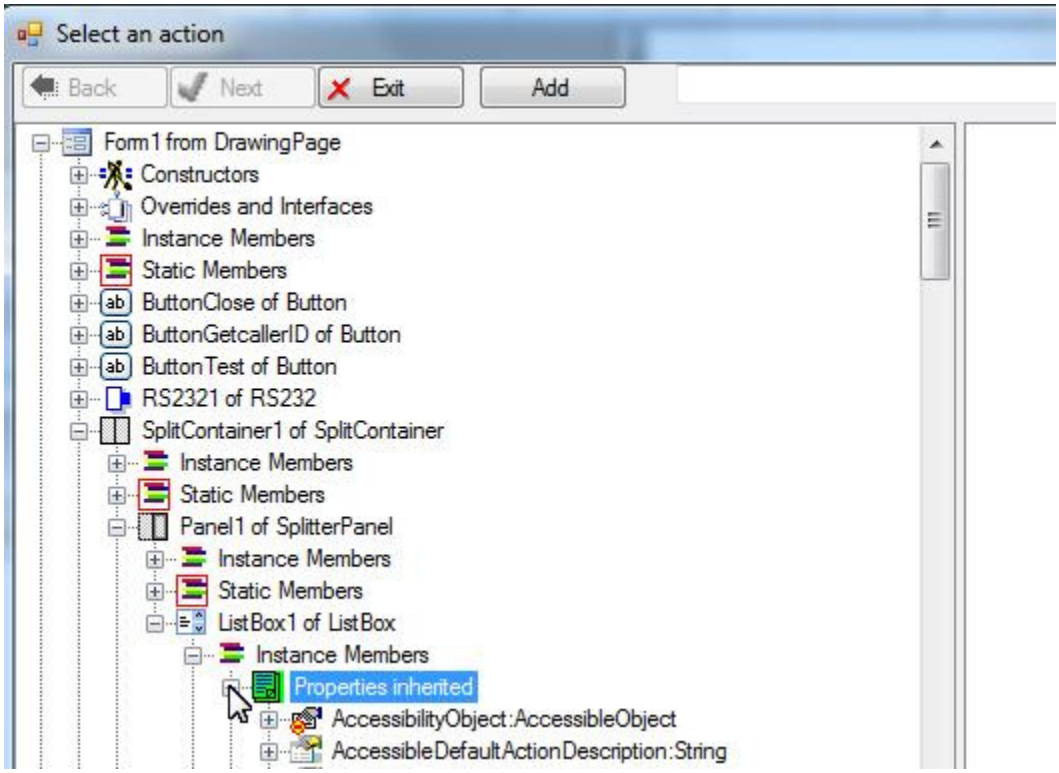


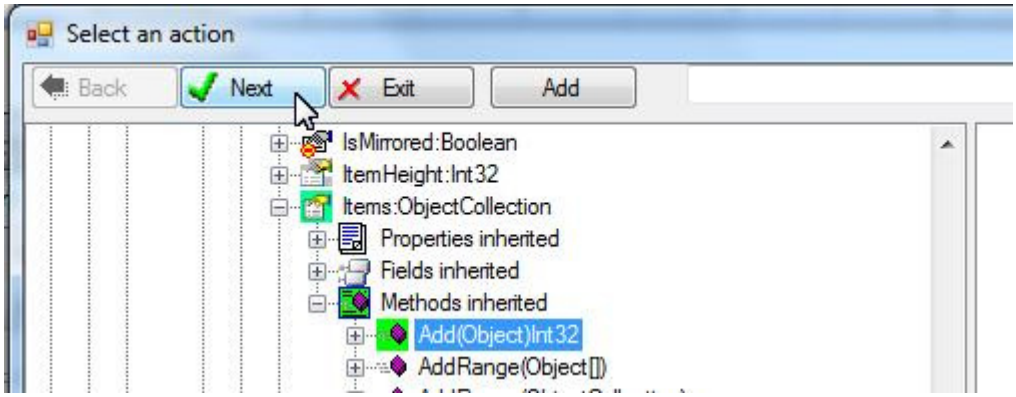

"Signal" column represents the message to watch for from the port device. "Event" column represents the event name for each signal. Do not use standard event names such as OK, RING, etc.



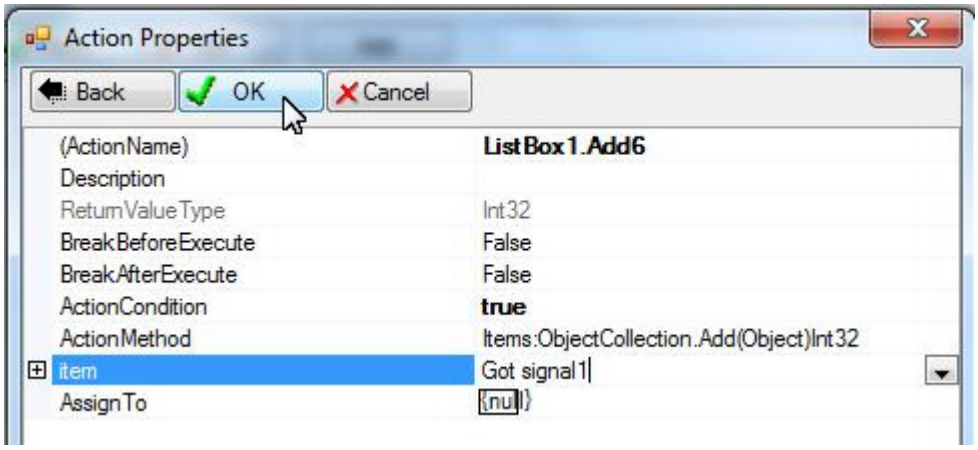

Once you defined <signal, event>, you may assign actions to events. In this sample, we also use list box to display events:





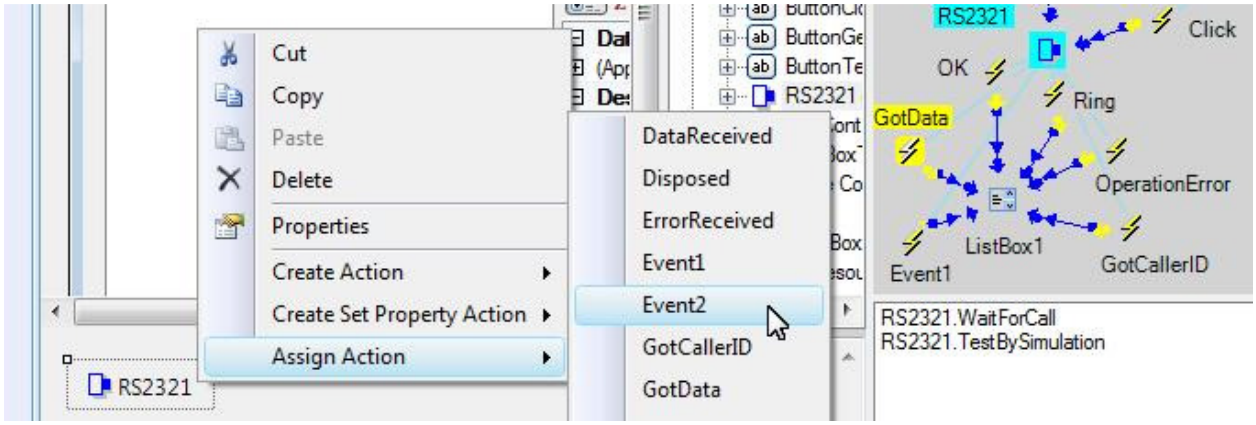



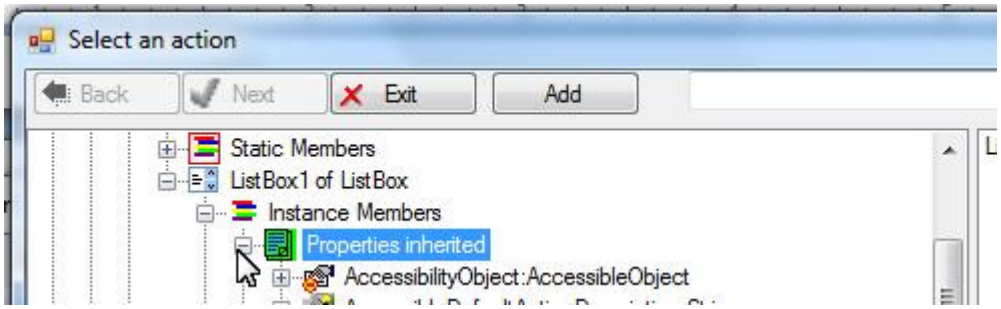

We simply display a text “Got signal1” in the list box when Event1 occurs:

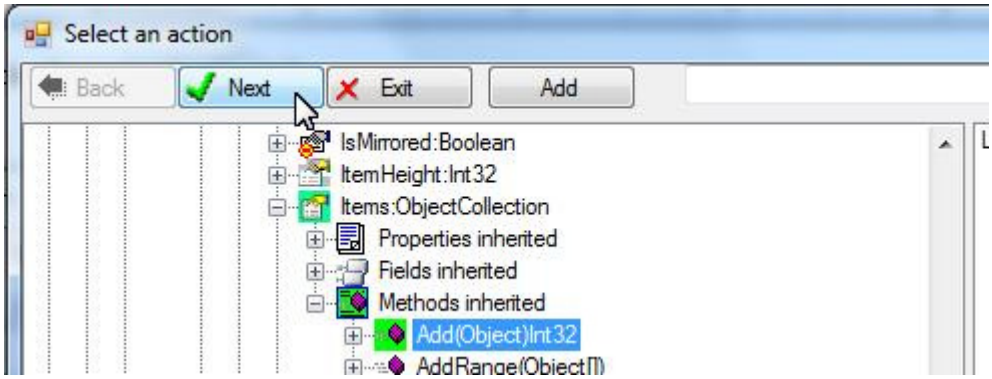



Event1 is assigned an action executed by the list box.

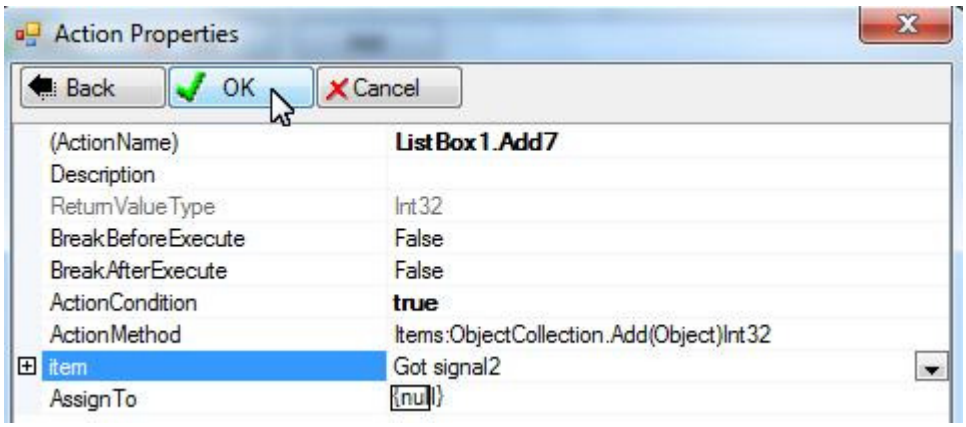
We may do the same for Event2:



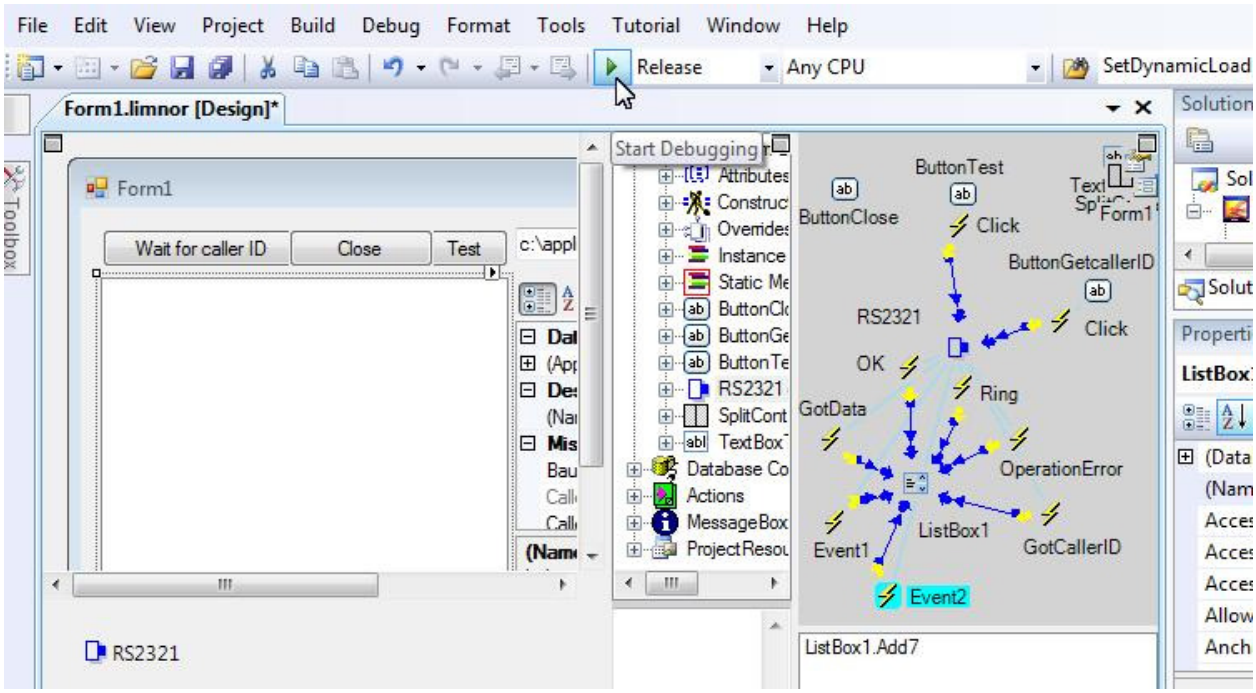



Simply display a text “Got signal2” in the list box:

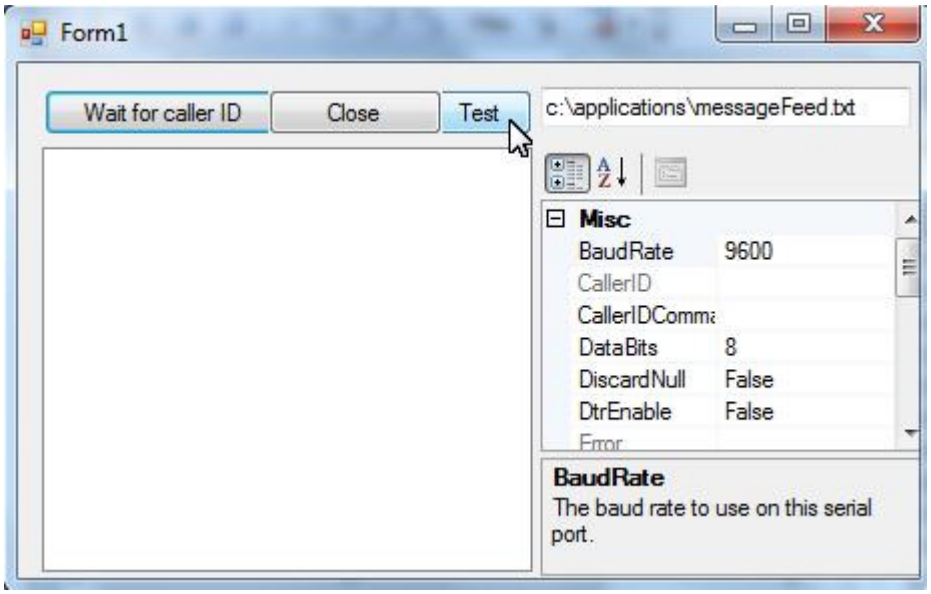



The action is created and assigned to Event2.

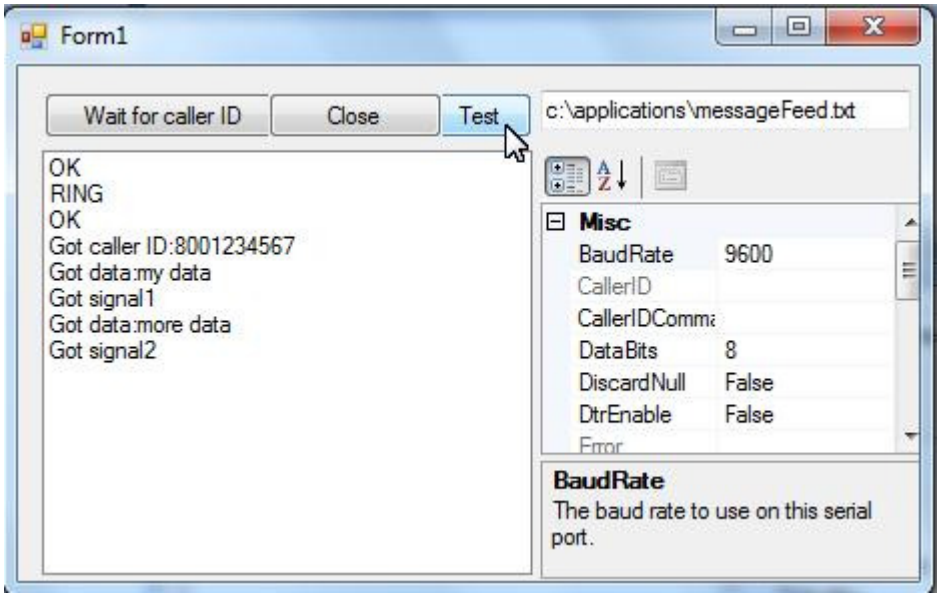
We may test the sample now.




The form appears:




Click the Test button. This time we see that Event1 and Event2 occur:




Note that this time we have “Got signal1” and “Got signal2”. Last time we had “Got data:singal1” and “Got data:signal2”.