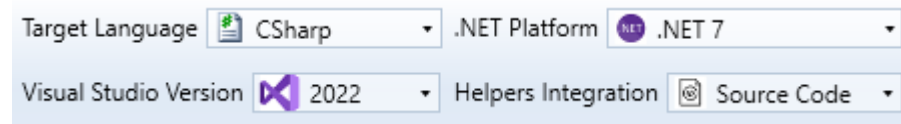


Visual Basic Upgrade Companion Version 9.5 Release Notes

.NET 7 Support

VBUC now fully supports the .NET 7 platform. This option will allow the user to generate code using this version as the target for C# and VB.NET.



Note: .NET 7 platform is available only when selecting Visual Studio 2022 as target.

GoTo support for C#

Previously, VBUC allowed you to convert the GoSub statements to local functions to have a VB6 like behavior, however, if a GoTo statement was in the code, it was not converted and an EWI and message was added instead.

With this release, VBUC can now convert some GoTo patterns to local functions, adding more VB6 like behavior.

VB6 snippet code:

```
Private Sub FormTest01(v As Integer)
    On Error GoTo ErrHandler
    Debug.Print "Some Code 01"
    GoTo Test01SomeLabel

Test01theFirstReturn:
    Debug.Print "Some Code 03"
    If v > 4 Then Err.Raise 1
    GoTo 100

Test01theSecondReturn:
    Debug.Print "Some Code 05"
    Exit Sub

Test01SomeLabel:
    Debug.Print "Some Code 02"
    GoTo Test01theFirstReturn

100
    Debug.Print "Some Code 04"
    GoTo Test01theSecondReturn

ErrHandler:
    Exit Sub
End Sub
```

```
Sub main()
    FormTest01 5
    FormTest01 0
End Sub
```

Output:

```
Some Code 01
Some Code 02
Some Code 03
Some Code 01
Some Code 02
Some Code 03
Some Code 04
Some Code 05
```

There are 2 calls to the FormTest01 method, the first one will be ended after print "Some Code 03", the second one will execute the whole code.

C# upgrade code with feature turned off:

```

private static void FormTest01(int v)
{
    //UPGRADE_TODO: (1065) Error handling statement (On Error Goto) could not be converted.
    UpgradeHelpers.Helpers.NotUpgradedHelper.NotifyNotUpgradedElement("On Error Goto Label (ErrorHandler)");
    Debug.WriteLine("Some Code 01");
    goto Test01SomeLabel;

    Test01theFirstReturn:
    Debug.WriteLine("Some Code 03");
    if (v > 4)
    {
        throw new System.Exception("1");
    }
    goto label100;

    Test01theSecondReturn:
    Debug.WriteLine("Some Code 05");
    return;

    Test01SomeLabel:
    Debug.WriteLine("Some Code 02");
    goto Test01theFirstReturn;

    label100:
    Debug.WriteLine("Some Code 04");
    goto Test01theSecondReturn;

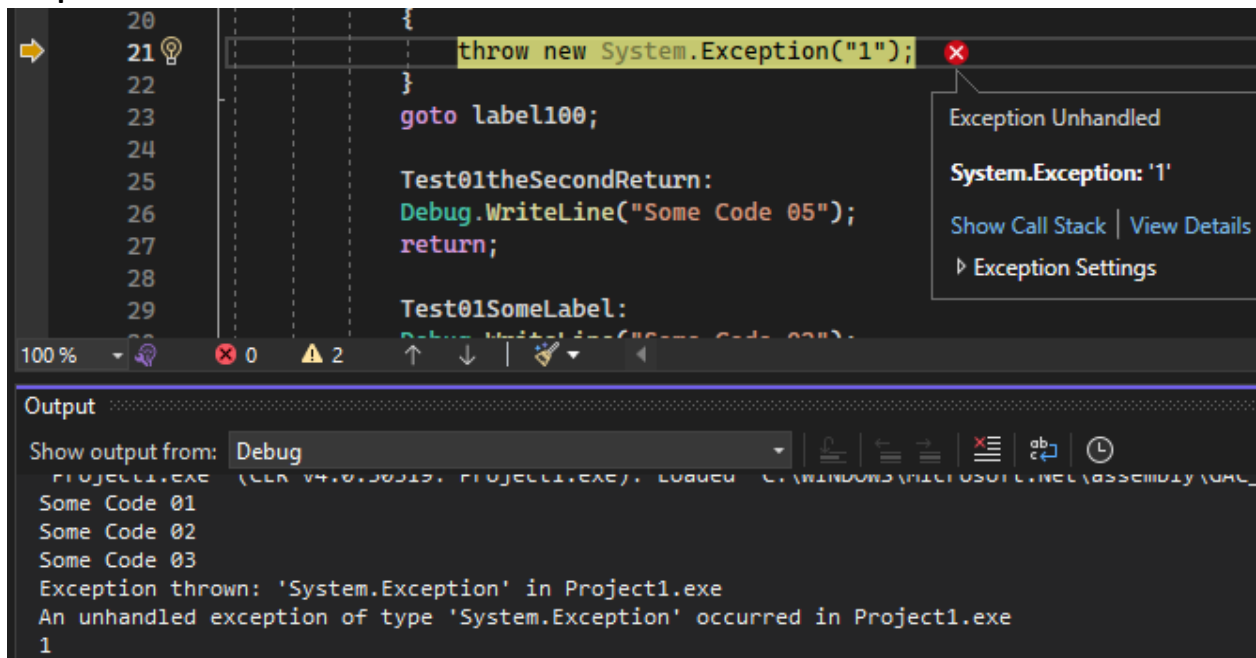
    ErrorHandler;
}

[STAThread]
0 references
public static void Main()
{
    FormTest01(5);
    FormTest01(0);
}

```

Note that there is a `NotUpgradedElement` notification where the `On Error Goto` statement is located, and the `Err.Raise` is converted to `throw Exception` statement.

Output:



The screenshot displays a Visual Studio IDE with a C# code file. Line 21 contains the statement `throw new System.Exception("1");`, which is highlighted in yellow. A tooltip for this line indicates an unhandled exception: "Exception Unhandled", "System.Exception: '1'", and provides options to "Show Call Stack", "View Details", and "Exception Settings".

The code in the editor includes:

```
20 {  
21     throw new System.Exception("1");  
22 }  
23 goto label100;  
24  
25 Test01theSecondReturn:  
26     Debug.WriteLine("Some Code 05");  
27     return;  
28  
29 Test01SomeLabel:  
30     Debug.WriteLine("Some Code 03");
```

The Output window at the bottom shows the following text:

```
Output  
Show output from: Debug  
PROJECT1.EXE (CLR v4.0.30319; PROJECT1.EXE): LOADED C:\WINDOWS\Microsoft.NET\Assembly\MSIL  
Some Code 01  
Some Code 02  
Some Code 03  
Exception thrown: 'System.Exception' in Project1.exe  
An unhandled exception of type 'System.Exception' occurred in Project1.exe  
1
```

Because of the *throw Exception*, the execution will be stopped when the Exception statement is reached and the second call will not be executed, causing a different behavior.

C# upgraded code with feature turned on:

```
private static void FormTest01(int v)
{
    try
    {
        Debug.WriteLine("Some Code 01");
        if (Label_Test01SomeLabel_function())
        {
            return;
        }

        Debug.WriteLine("Some Code 03");
        if (v > 4)
        {
            throw new System.Exception("1");
        }
        if (Label_100_function())
        {
            return;
        }

        Debug.WriteLine("Some Code 05");

        bool label_Test01SomeLabel_function()
        {
            Debug.WriteLine("Some Code 02");
            return false;
        }
        bool label_100_function()
        {
            Debug.WriteLine("Some Code 04");
            return false;
        }
    }
    catch
    {
        return;
    }
}

[STAThread]
0 references
public static void Main()
{
    FormTest01(5);
    FormTest01(0);
}
```

The On Error Goto statement caused to convert the whole method in a try-catch, and the GoTo labels were converted to local functions, as the method was converted into a Try-Catch when the *throw*

Exception statement is reached, the catch will prevent the execution to be stopped and the execution will continue with the second call.

Output:

```
Some Code 01
Some Code 02
Some Code 03
Exception thrown: 'System.Exception' in Project1.exe
Some Code 01
Some Code 02
Some Code 03
Some Code 04
Some Code 05
```

Note that this output is similar to the original VB6 code, an Exception was added to the output but the execution was the same as VB6.

Other Improvements include but are not limited to:

- Databases mappings
- COM Interop improvements
- Microsoft controls mappings
- Color mappings