

Getting Started NX Journaling

Jeff Roark

jeffrey.roark@yfai.com

Yanfeng Automotive Interiors

Journaling, NXOpen, and Block UI Styler

- NXOpen
 - Compile dll
 - Windows GUI
 - Visual Studio IDE
 - VB.Net, C++, Java and Python
- Journaling
 - Free
 - Based on VB
 - Not compiled
- Block UI Styler
 - NX compliant dialogs
 - Advantages
 - data types
 - User Interaction consistent with NX
 - Requires a license
- Winforms
 - Free
 - Not compliant with NX standards
 - More flexible (Pros/Cons)

What is a Journal?

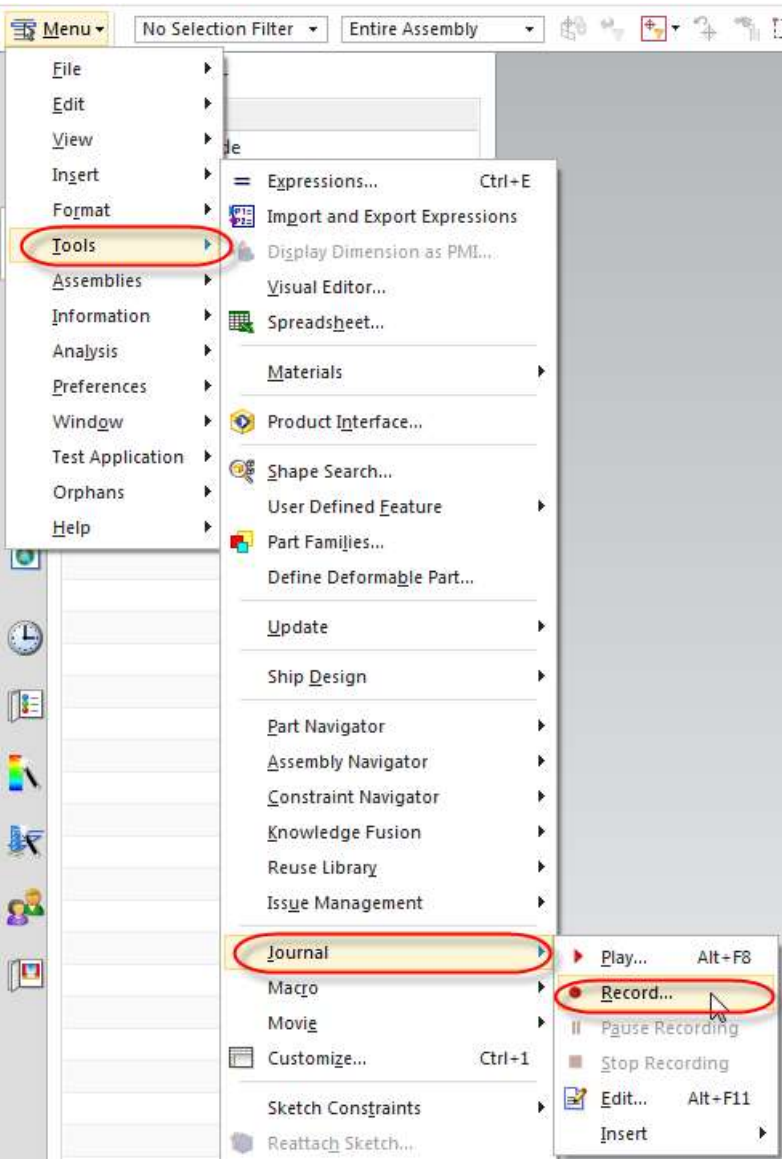
- A Journal is a programming language based on VB.
- The Journal file can be created interactively while the designer is using standard NX functions.
- The Journal file is interpreted by NX when the designer replays the Journal.
- It is a text file.
 - The Journal can be edited with NX, Notepad, Notepad++, Visual Studio
- It is not compiled.
- It does not require a special license to create, use or distribute the file.

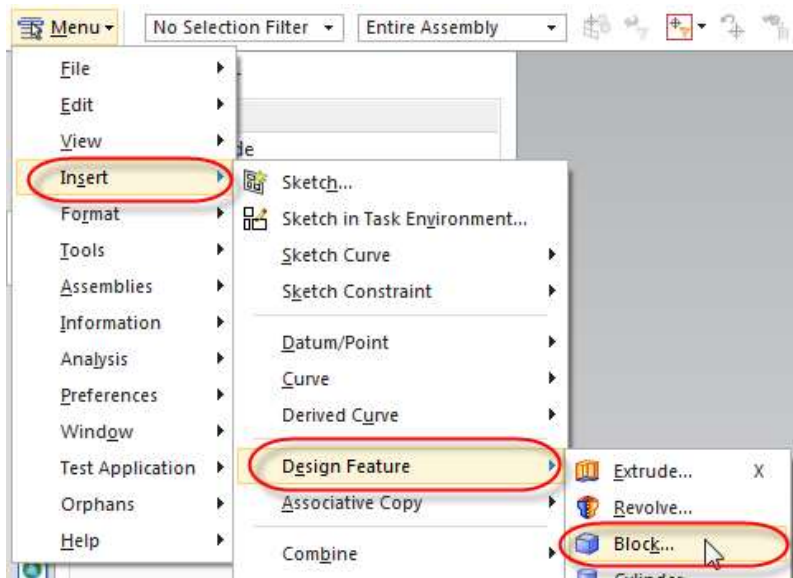
What are we going to do?

- We are going to record a Journal with working interactively with NX.
- Play the Journal back to verify the Journal recorded the steps.
- Look at the code with NX.
- Edit the Journal with Visual Studio Express.
- Add an Inputbox as a user interface.
- Play the edited Journal to verify it is working as expected.

Record a Journal

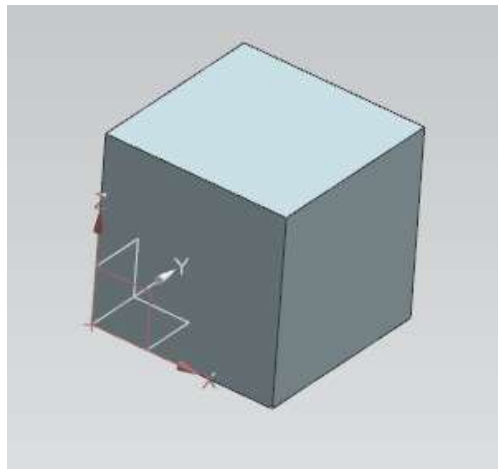
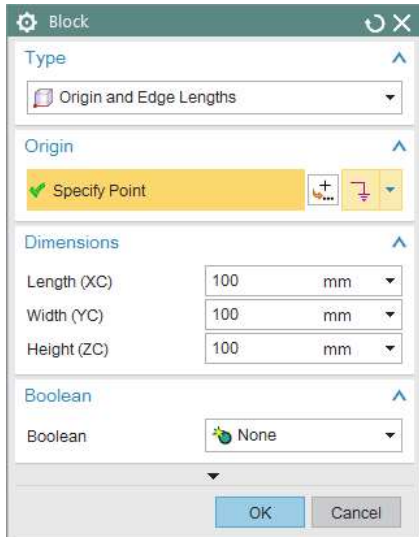
- Start NX
- Start a New Part
- Start Recording
- Insert a Block using default values
- Stop the Recording
- Delete the Block
- Play the Journal
- Edit the Journal to include the Inputbox
- Play the modified Journal to verify it works.

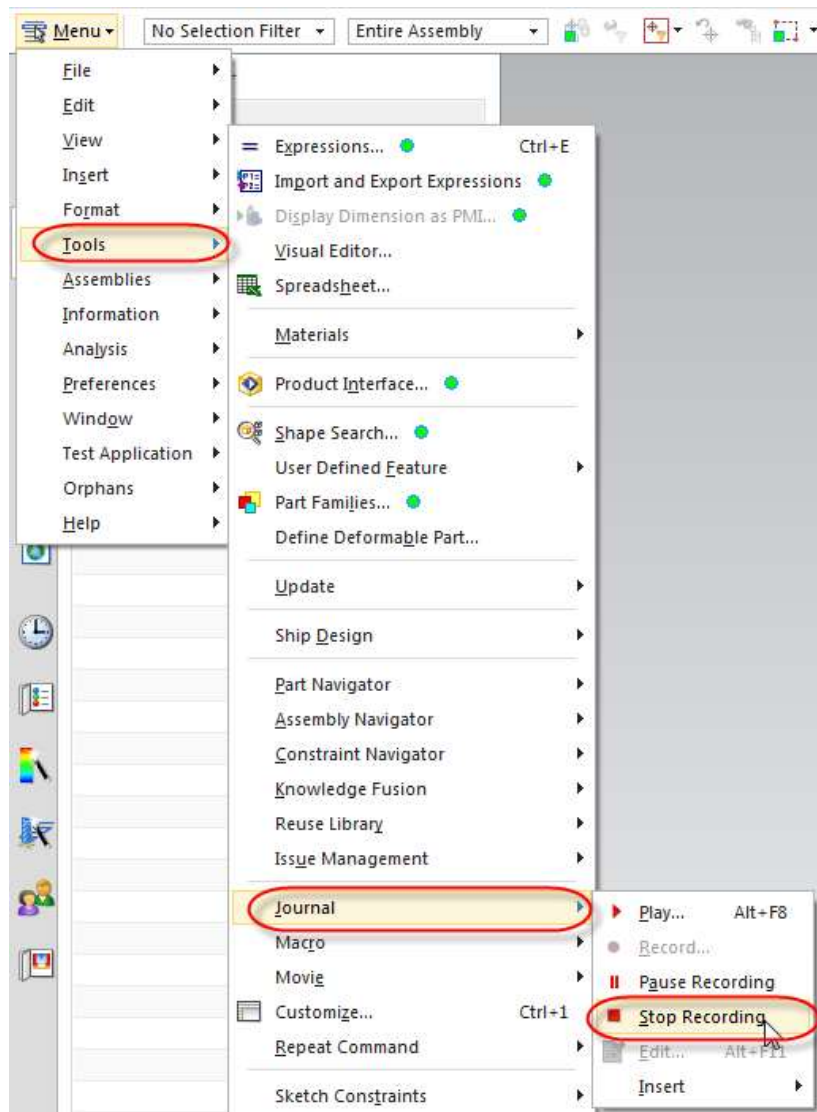




Make a Block

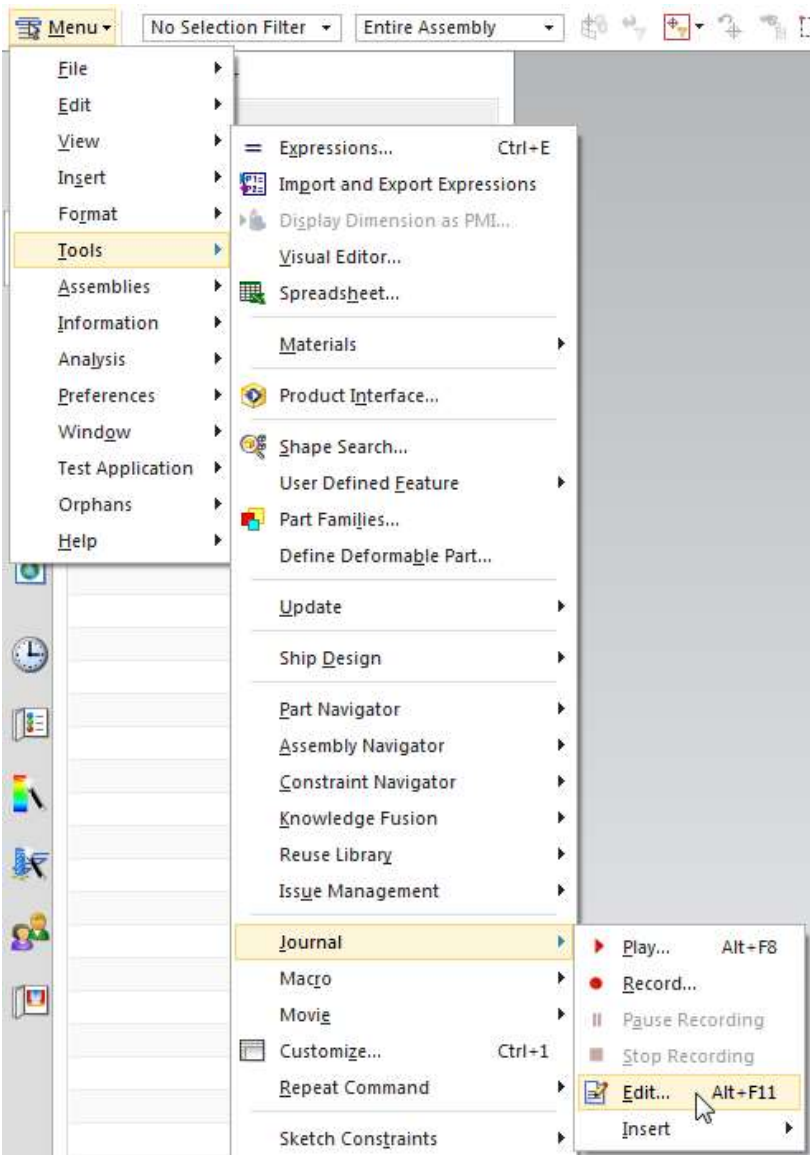
- Use the default Values

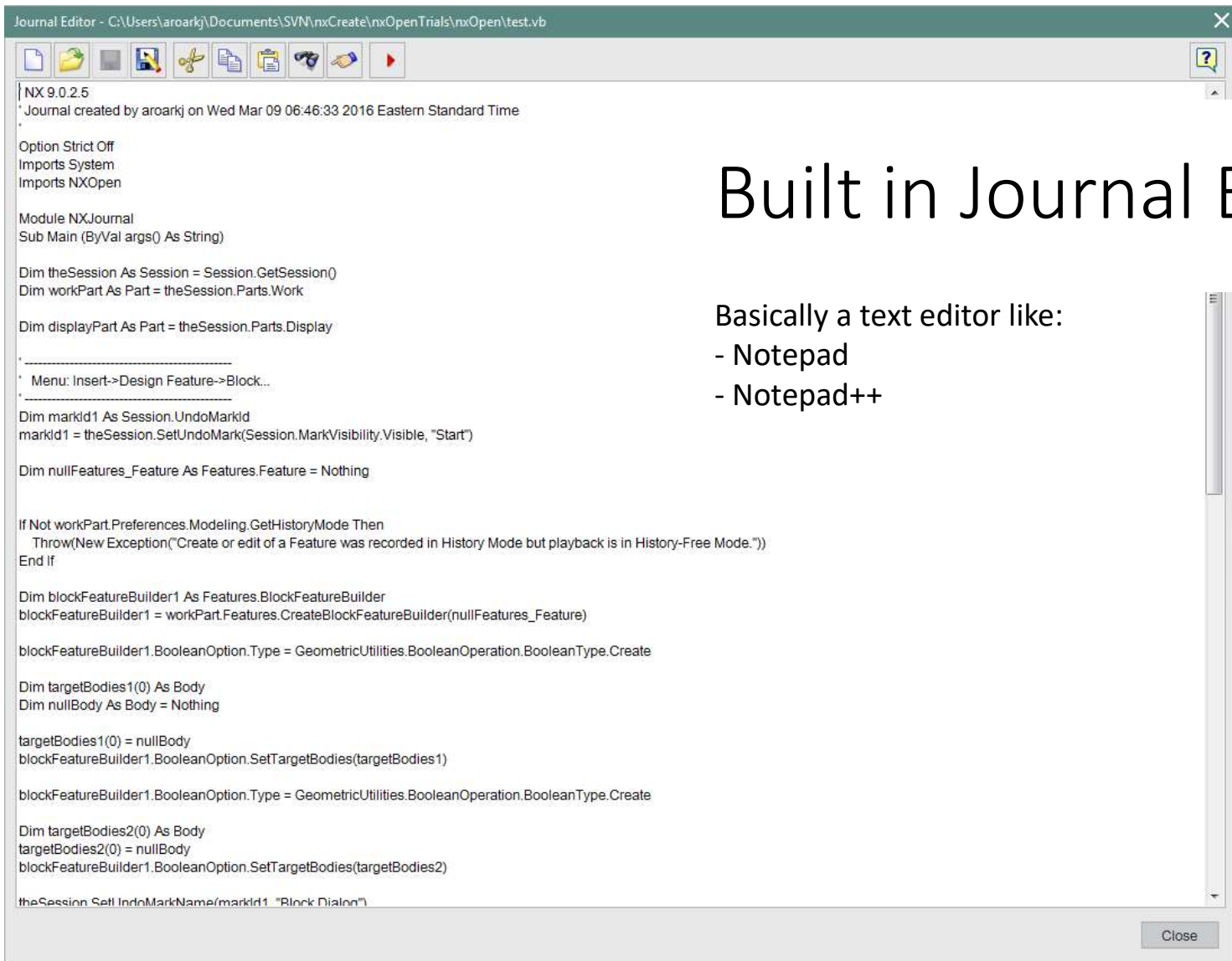




Stop the Recording

Edit the recorded Journal





Built in Journal Editor

Basically a text editor like:

- Notepad
- Notepad++

<https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx>

Visual Studio Express

Visual Studio Express editions provide free tools to develop applications for a specific platform, such as Windows Universal Platform applications, web sites, and Windows desktop applications.

New edition available

Visual Studio can be used
to edit a Journal



Visual Studio Community has all the features of Express and more, **and is still free** for individual developers, open source projects, academic research, education, and small professional teams.

*I got this from Microsoft's Visual Studio website. I can not responsible for the content of polices of Microsoft or Visual Studio. Please refer to their website for current up to date content and polices.

<https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx>

Express editions

Used for todays demo

Express for Desktop

Supports the creation of desktop applications for Windows.

Download

Express for Web

Create standards-based, responsive websites, web APIs, or real-time online experiences using ASP.NET.

Download

Express for Windows

Provides the core tools for building compelling, innovative apps for Universal Windows Platform. Windows 10 is required.

Download

Team Foundation Server 2015 Express

Free source-code-control, project-management, and team-collaboration platform.

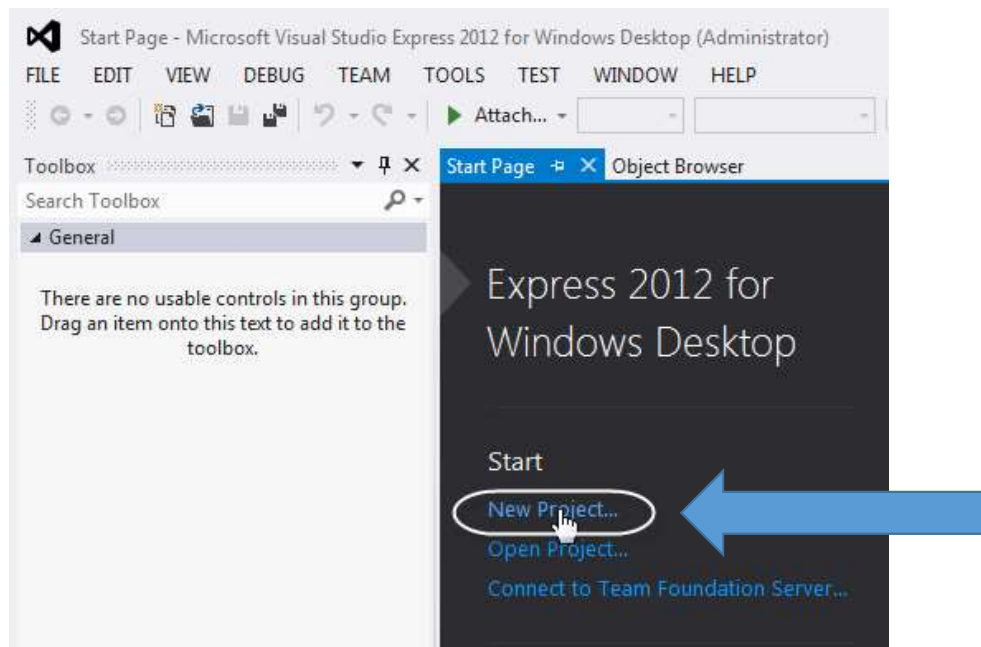
Download

*I got this from Microsoft's Visual Studio website. I can not responsible for the content of polices of Microsoft or Visual Studio. Please refer to their website for current up to date content and polices.

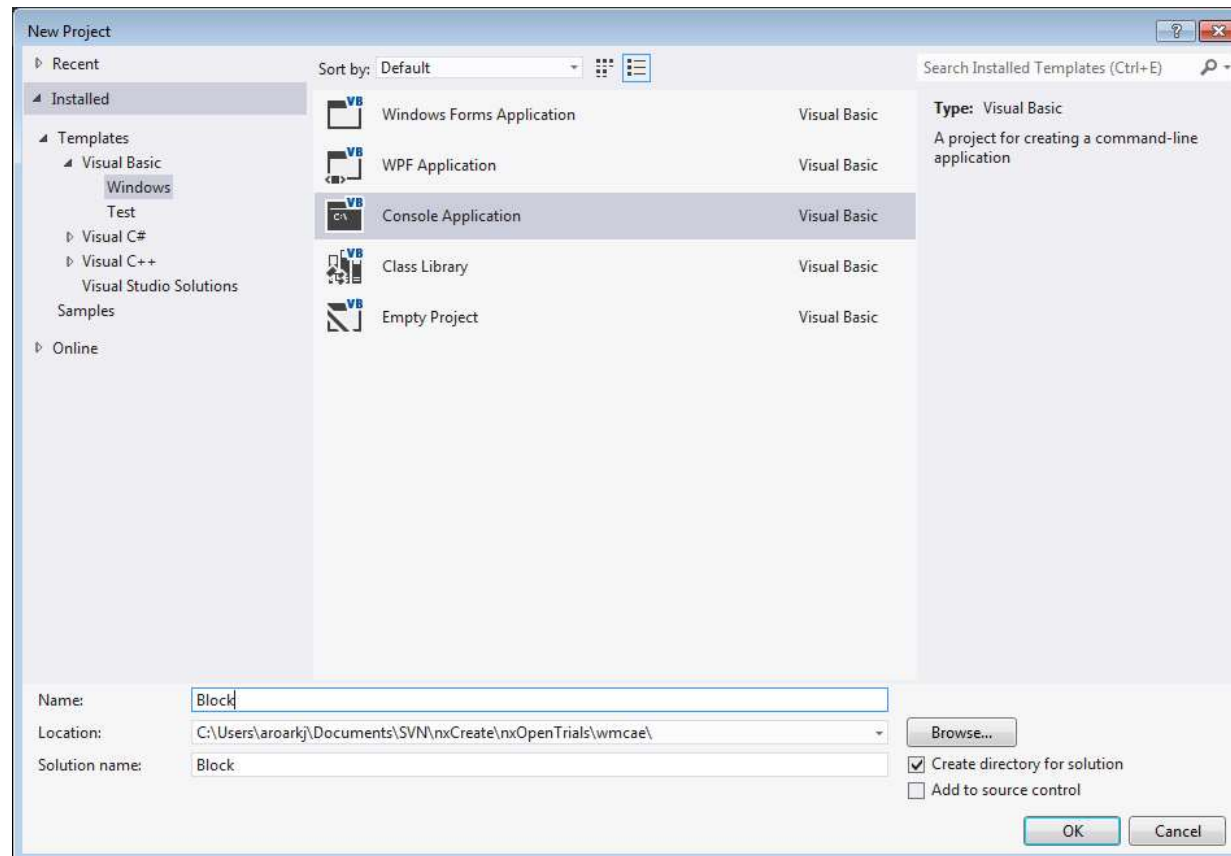
Commercial use of Express products

Visual Studio Express products are available at no charge and may be used for commercial, production usage subject to the license terms provided with each product. For example, you can use Express for Windows to create apps that you can then submit for sale in the Windows Store.

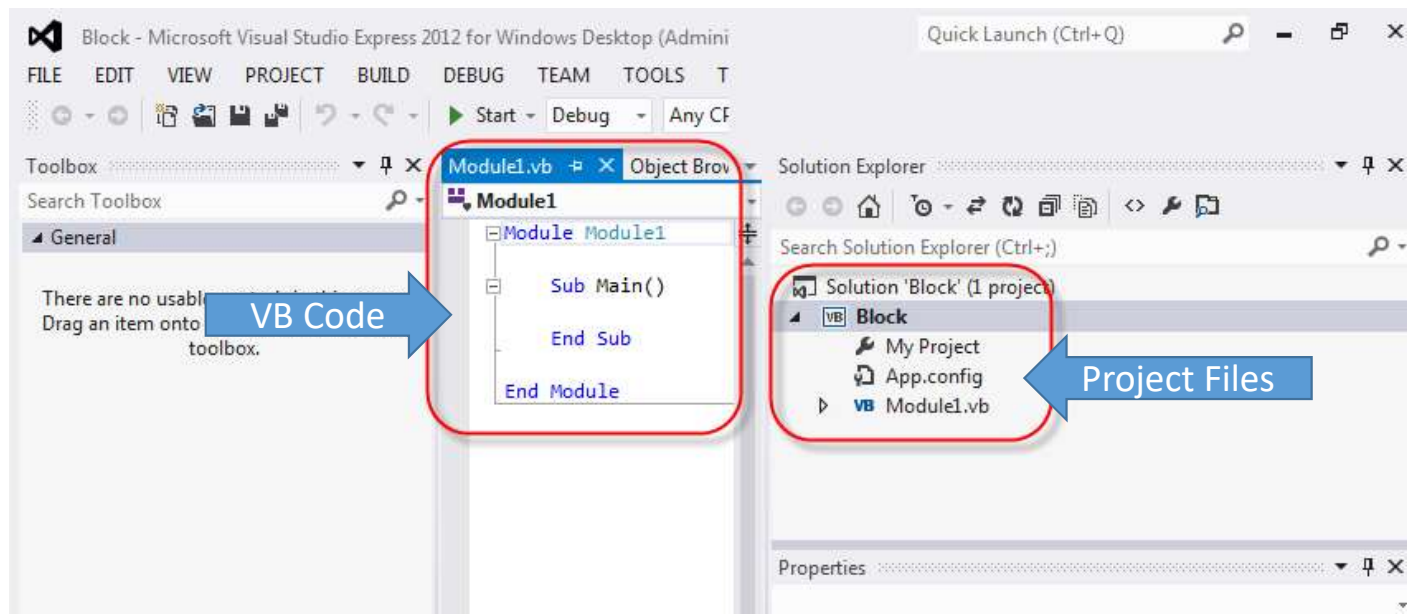
Starting Visual Studio Express



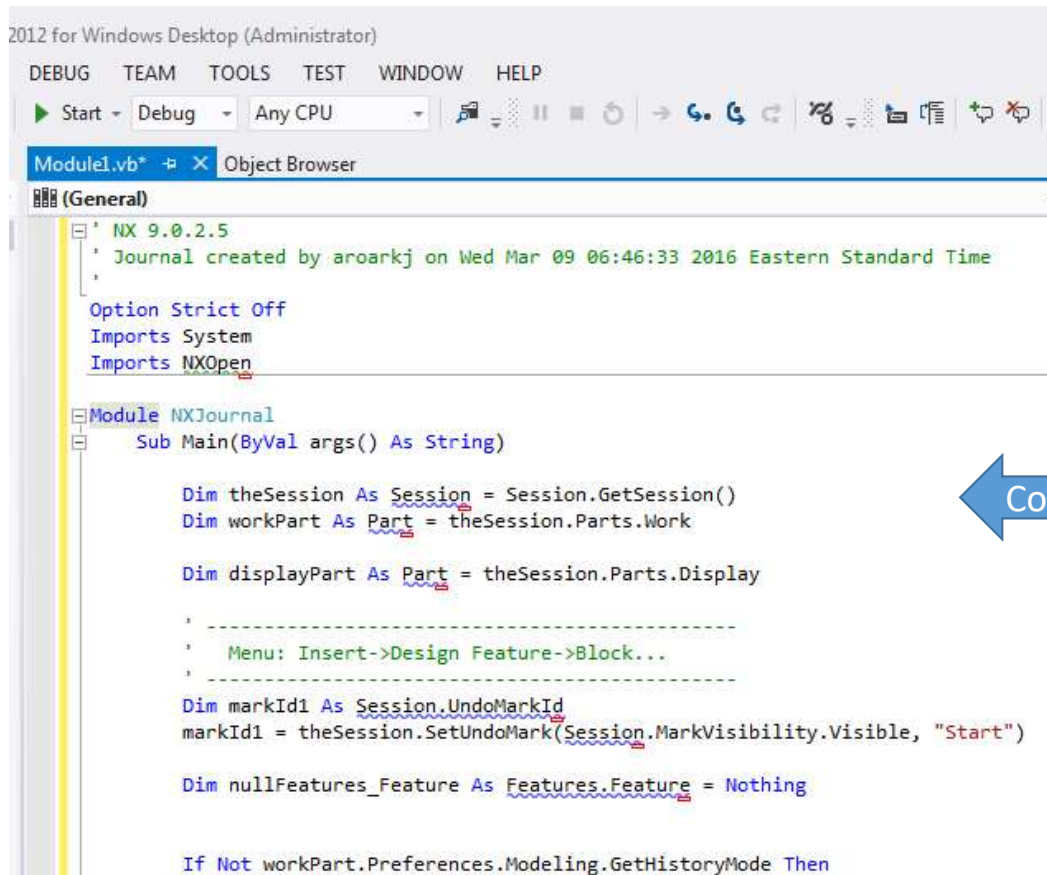
Console Application



Visual Studio Interface



Copy/Paste the Journal to Visual Studio



```
2012 for Windows Desktop (Administrator)
DEBUG TEAM TOOLS TEST WINDOW HELP
Start Debug Any CPU
Module1.vb* Object Browser
(General)
' NX 9.0.2.5
' Journal created by aroarkj on Wed Mar 09 06:46:33 2016 Eastern Standard Time
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
Sub Main(ByVal args() As String)

Dim theSession As Session = Session.GetSession()
Dim workPart As Part = theSession.Parts.Work

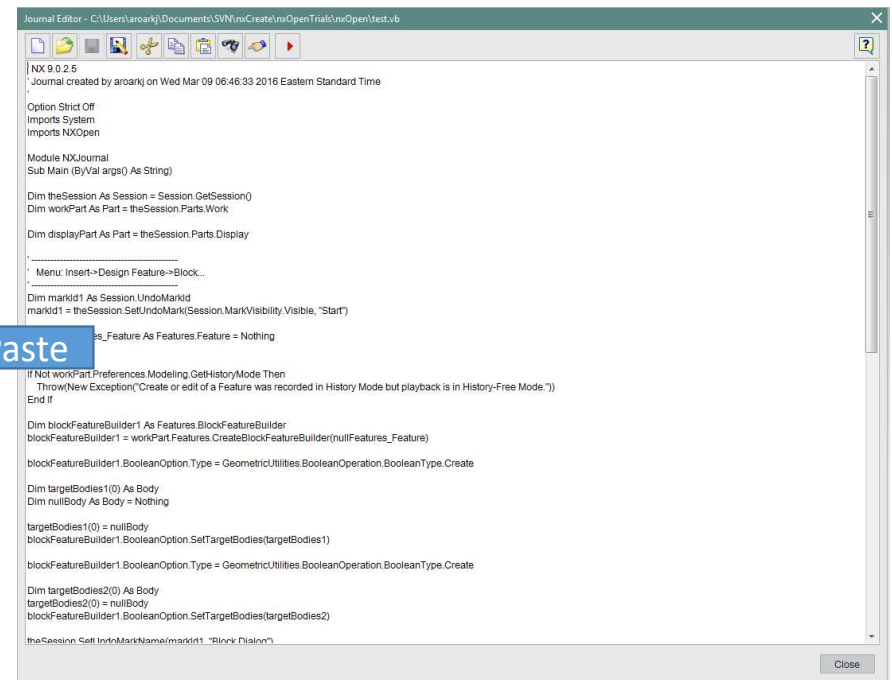
Dim displayPart As Part = theSession.Parts.Display

' -----
' Menu: Insert->Design Feature->Block...
' -----

Dim markId1 As Session.UndoMarkId
markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

Dim nullFeatures_Feature As Features.Feature = Nothing

If Not workPart.Preferences.Modeling.GetHistoryMode Then
```



```
Journal Editor - C:\Users\aroarkj\Documents\SVN\CreateNXOpenTrials\NXOpen\test.vb
NX 9.0.2.5
Journal created by aroarkj on Wed Mar 09 06:46:33 2016 Eastern Standard Time

Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
Sub Main (ByVal args() As String)

Dim theSession As Session = Session.GetSession()
Dim workPart As Part = theSession.Parts.Work

Dim displayPart As Part = theSession.Parts.Display

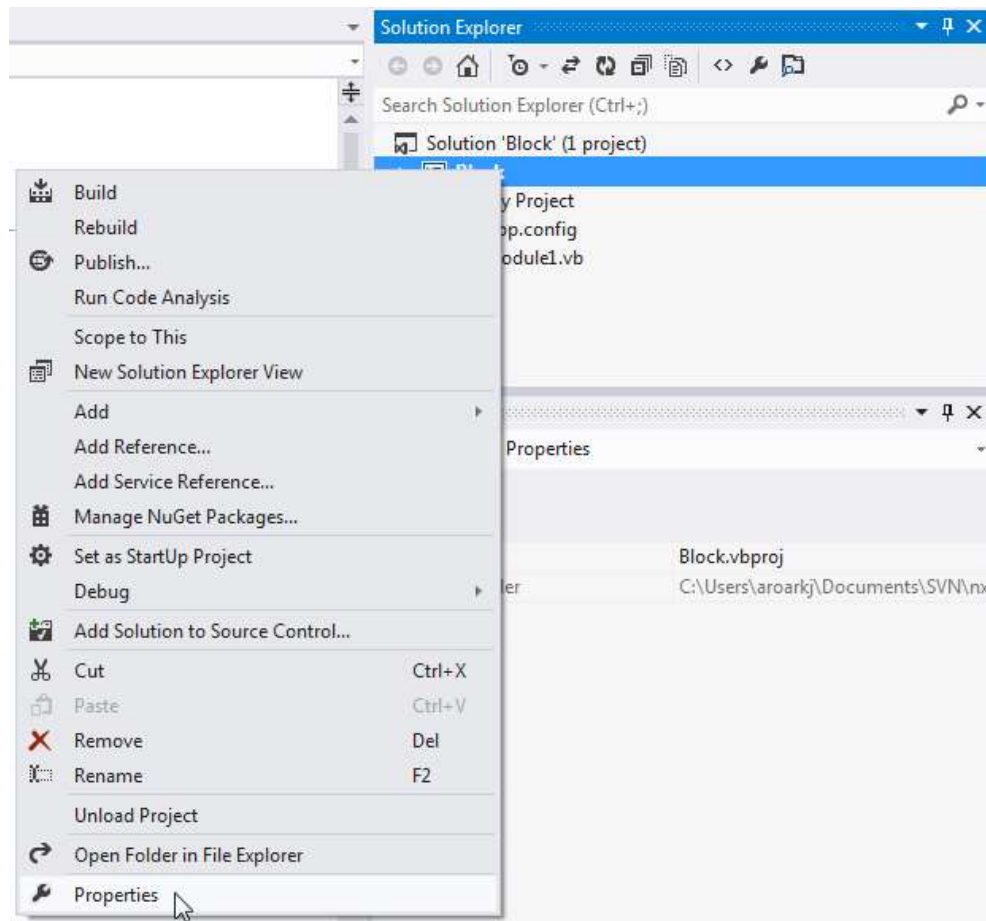
' -----
' Menu: Insert->Design Feature->Block...
' -----

Dim markId1 As Session.UndoMarkId
markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

Dim nullFeatures_Feature As Features.Feature = Nothing

If Not workPart.Preferences.Modeling.GetHistoryMode Then
```


Reference NX API



Application programming interface (API)

Enable Visual Studio to help you with syntax of NX API.

The API consist of Classes and functions that have been provided by NX.

Reference API

Installation Path:

Navigate to where NX is installed

C:\tcplm\products\nx\9.0.2.5\UGII\managed

The screenshot shows the Visual Studio IDE with the 'References' tab selected in the Solution Explorer. The 'References' list shows several assemblies, including NXOpen.dll, NXOpenUI.dll, NXOpen.Utilities.dll, NXOpen.UF.dll, and ManagedLoader.dll. A red box highlights these five assemblies. A blue arrow labeled 'Reference' points to the 'References' tab, and another blue arrow labeled 'Browse' points to the 'Browse' button in the 'Add Reference' dialog. The 'Add Reference' dialog is open, showing a list of assemblies. The same five assemblies are listed, and a red box highlights the 'Add...' button. A blue arrow labeled 'Select Add' points to the 'Add...' button. The 'Add Reference' dialog also shows a 'Name' field with 'NXOpen.dll' and a 'Created by' field with 'Siemens Product Lifecycle Management'. The 'File Version' is 9.0.2.5. The 'Add...' button is circled in red.

Reference Name	Type	Vers...	Copy Local	Path
System	.NET	4.0.0.0	False	C:\Program Files (x86)\...
System.Core	.NET	4.0.0.0	False	C:\Program Files (x86)\...

Name	Path
NXOpen.UF.dll	C:\programmi\apl\ugnx85\UGII\managed\NXOpen.UF.dll
NXOpenUI.dll	C:\programmi\apl\ugnx85\UGII\managed\NXOpenUI.dll
testClass.dll	C:\Users\aroarkj\Documents\SVN\nxCreate\WinFor...
ManagedLoader.dll	C:\programmi\apl\ugnx85\UGII\managed\ManagedLoader.dll
testClass.exe	C:\Users\aroarkj\Documents\SVN\nxCreate\WinFor...
NXOpen.Utilities.dll	C:\programmi\apl\ugnx85\UGII\managed\NXOpen.Utilities.dll
NXOpen.dll	C:\programmi\apl\ugnx85\UGII\managed\NXOpen.dll
Snap.dll	C:\programmi\apl\ugnx85\UGII\managed\Snap.dll
MimSnap.dll	C:\programmi\apl\ugnx85\UGII\managed\MimSnap.dll
NXOpen.dll	C:\tcplm\products\nx\9.0.2.5\UGII\managed\NXOpen.dll
NXOpenUI.dll	C:\tcplm\products\nx\9.0.2.5\UGII\managed\NXOpenUI.dll
NXOpen.Utilities.dll	C:\tcplm\products\nx\9.0.2.5\UGII\managed\NXOpen.Utilities...
NXOpen.UF.dll	C:\tcplm\products\nx\9.0.2.5\UGII\managed\NXOpen.UF.dll
ManagedLoader.dll	C:\tcplm\products\nx\9.0.2.5\UGII\managed\ManagedLoader...

Name: NXOpen.dll
Created by: Siemens Product Lifecycle Management
File Version: 9.0.2.5

Add... Remove Update

The API is Now Referenced

Block: Module1.vb* X Object Browser

(General)

```
' NX 9.0.2.5
' Journal created by aroarkj on Wed Mar 09 06:46:33 2016 Eastern Sta
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
    Sub Main(ByVal args() As String)

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' -----
        ' Menu: Insert->Design Feature->Block...
        ' -----

        Dim markId1 As Session.UndoMarkId
        markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible)

        Dim nullFeatures_Feature As Features.Feature = Nothing
```

After Reference

Module1.vb* X Object Browser

(General)

```
' NX 9.0.2.5
' Journal created by aroarkj on Wed Mar 09 06:46:33 2016 Eastern Standard Time
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
    Sub Main(ByVal args() As String)

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' -----
        ' Menu: Insert->Design Feature->Block...
        ' -----

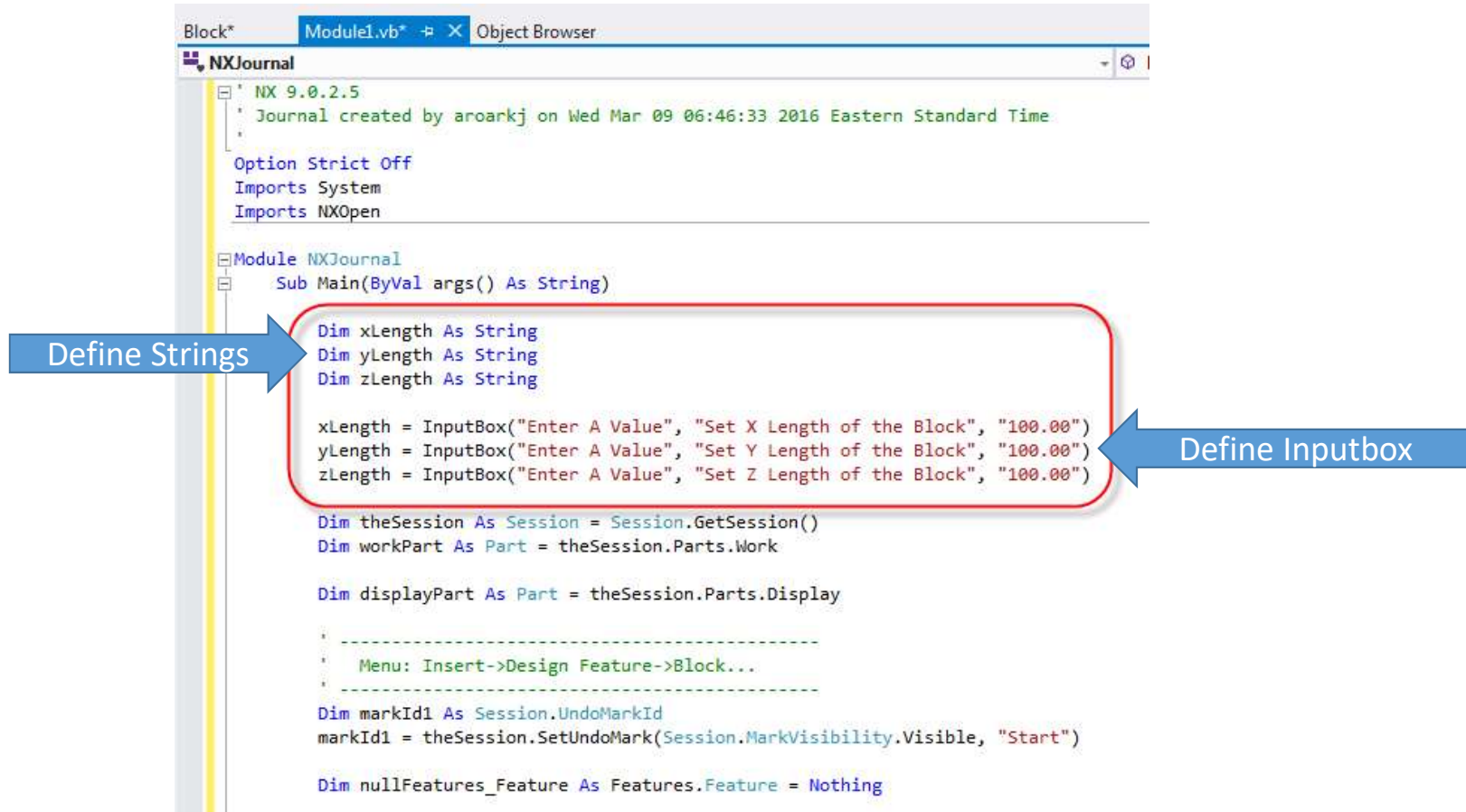
        Dim markId1 As Session.UndoMarkId
        markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

        Dim nullFeatures_Feature As Features.Feature = Nothing

        If Not workPart.Preferences.Modeling.GetHistoryMode Then
```

Before Reference

Add Inputbox to the Journal



```
Block*  Module1.vb*  Object Browser
NXJournal
' NX 9.0.2.5
' Journal created by aroarkj on Wed Mar 09 06:46:33 2016 Eastern Standard Time
'
Option Strict Off
Imports System
Imports NXOpen

Module NXJournal
    Sub Main(ByVal args() As String)

        Dim xLength As String
        Dim yLength As String
        Dim zLength As String

        xLength = InputBox("Enter A Value", "Set X Length of the Block", "100.00")
        yLength = InputBox("Enter A Value", "Set Y Length of the Block", "100.00")
        zLength = InputBox("Enter A Value", "Set Z Length of the Block", "100.00")

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' -----
        ' Menu: Insert->Design Feature->Block...
        ' -----

        Dim markId1 As Session.UndoMarkId
        markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

        Dim nullFeatures_Feature As Features.Feature = Nothing
```

Intellisense Help provided by Referencing the NX API

Popup help as you type

```
xLength = InputBox(("Enter A Value", "Set X Length of the Block", "100.00"))
```

InputBox(**Prompt** As String, [Title As String = ""], [DefaultResponse As String = ""], [XPos As Integer = -1], [YPos As Integer = -1]) As String
Displays a prompt in a dialog box, waits for the user to input text or click a button, and then returns a string containing the contents of the input.

Prompt: Required String expression displayed as the message in the dialog box. The maximum length of Prompt is approximately 1024 characters (Chr(10)), or a carriage return/line feed combination (Chr(13) & Chr(10)) between each line.

Dim displayPart As **AcceptRejectRule**

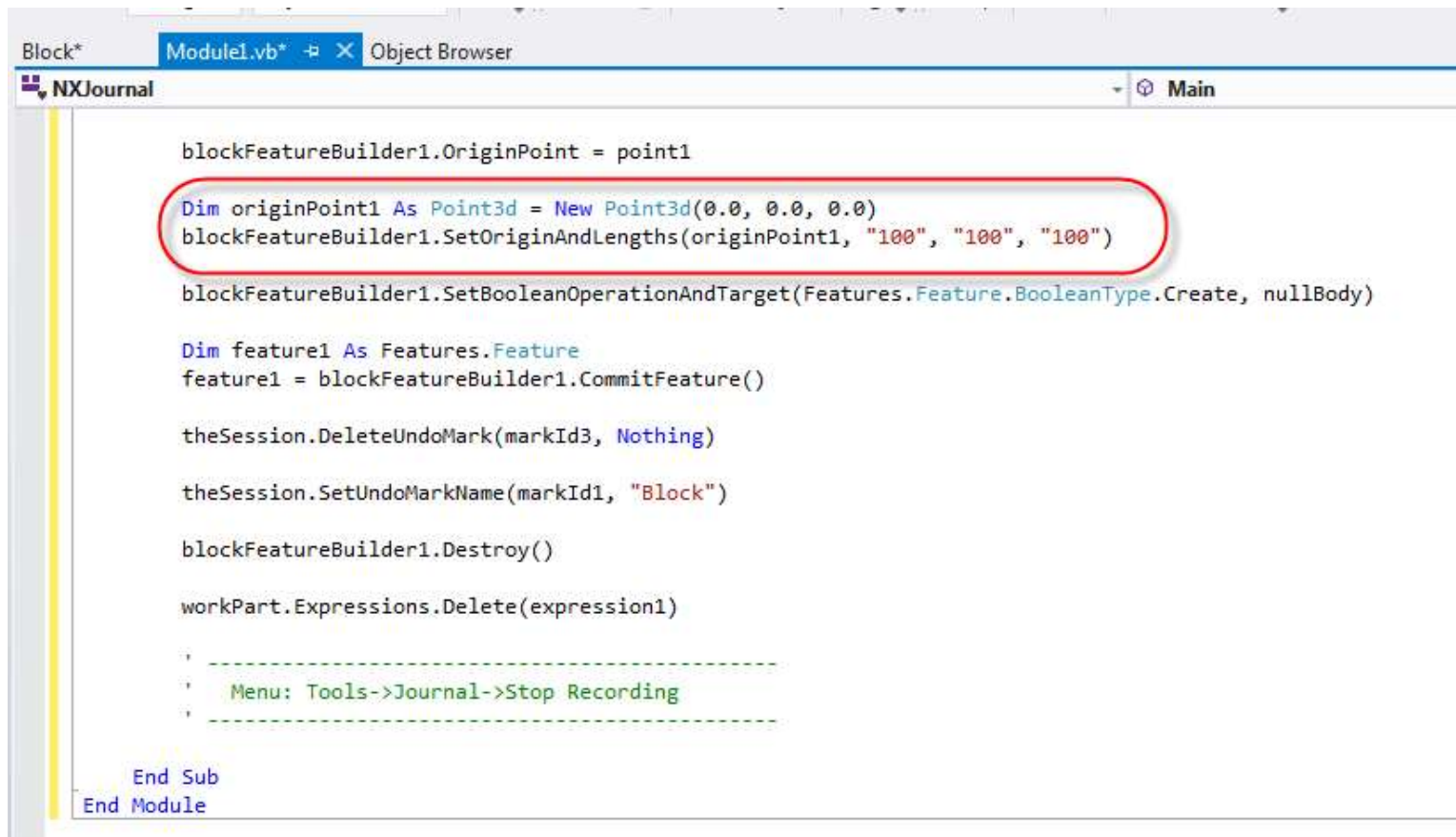
- AccessViolationException
- ActivationContext
- Activator
- AddressOf
- Aggregate
- AggregateException
- AngularLimit
- AngularSpring
- Common
- All

Menu: Insert->D

```
Dim markId1 As Session  
markId1 = theSession  
Dim nullFeatures_Fe
```

"Start")

Pass the values from the Inputbox to this line in the code



```
Block*  Module1.vb*  Object Browser
NXJournal  Main

    blockFeatureBuilder1.OriginPoint = point1

    Dim originPoint1 As Point3d = New Point3d(0.0, 0.0, 0.0)
    blockFeatureBuilder1.SetOriginAndLengths(originPoint1, "100", "100", "100")

    blockFeatureBuilder1.SetBooleanOperationAndTarget(Features.Feature.BooleanType.Create, nullBody)

    Dim feature1 As Features.Feature
    feature1 = blockFeatureBuilder1.CommitFeature()

    theSession.DeleteUndoMark(markId3, Nothing)

    theSession.SetUndoMarkName(markId1, "Block")

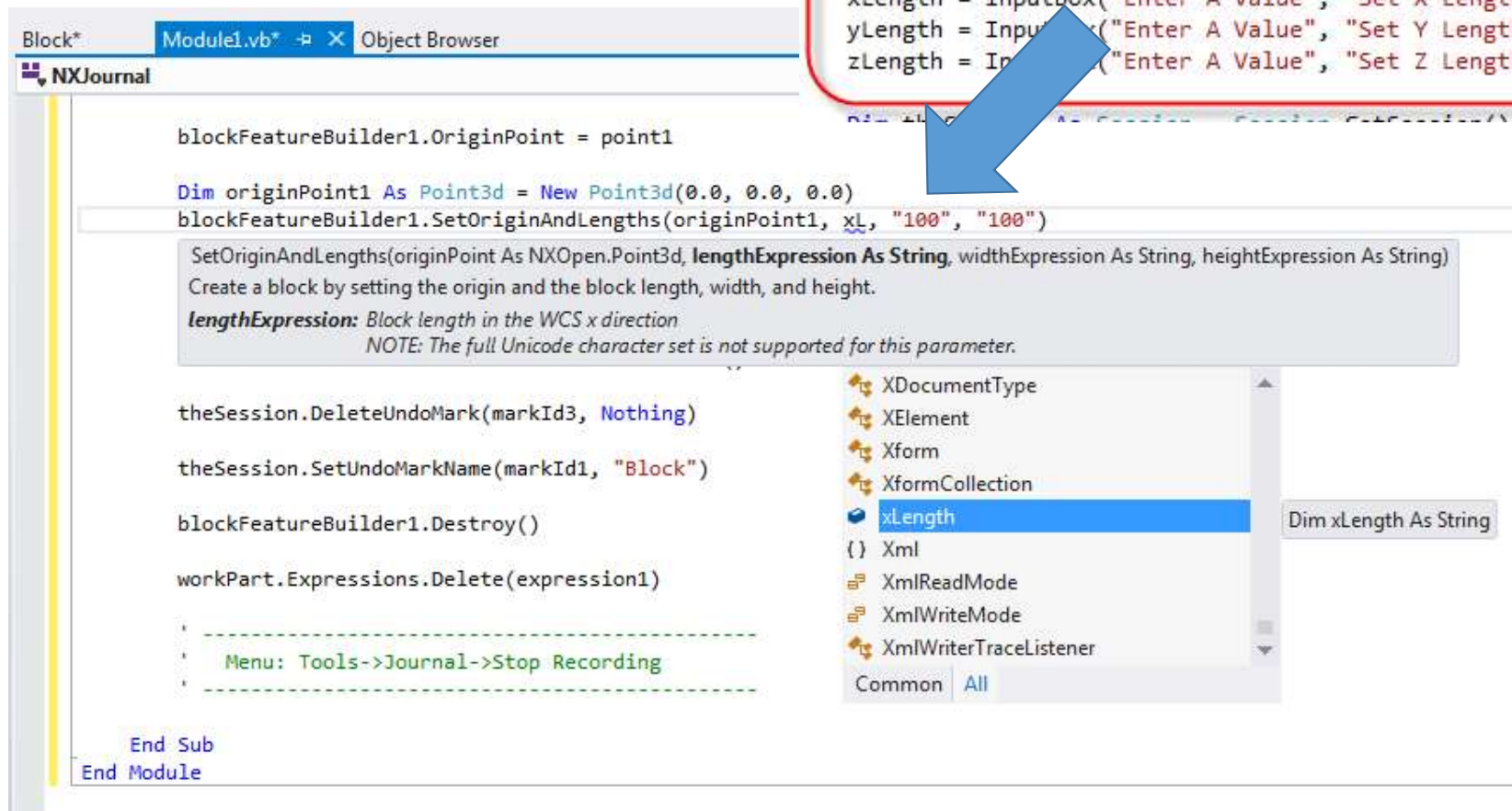
    blockFeatureBuilder1.Destroy()

    workPart.Expressions.Delete(expression1)

    ' -----
    '   Menu: Tools->Journal->Stop Recording
    ' -----

End Sub
End Module
```


Edit code Intellisense Popups help



The screenshot shows the NXJournal code editor with the following code:

```
Block* Module1.vb* X Object Browser  
NXJournal  
  
blockFeatureBuilder1.OriginPoint = point1  
  
Dim originPoint1 As Point3d = New Point3d(0.0, 0.0, 0.0)  
blockFeatureBuilder1.SetOriginAndLengths(originPoint1, xL, "100", "100")  
  
theSession.DeleteUndoMark(markId3, Nothing)  
  
theSession.SetUndoMarkName(markId1, "Block")  
  
blockFeatureBuilder1.Destroy()  
  
workPart.Expressions.Delete(expression1)  
  
Menu: Tools->Journal->Stop Recording  
  
End Sub  
End Module
```

The Intellisense popup for the `SetOriginAndLengths` method is displayed, showing the following signature and description:

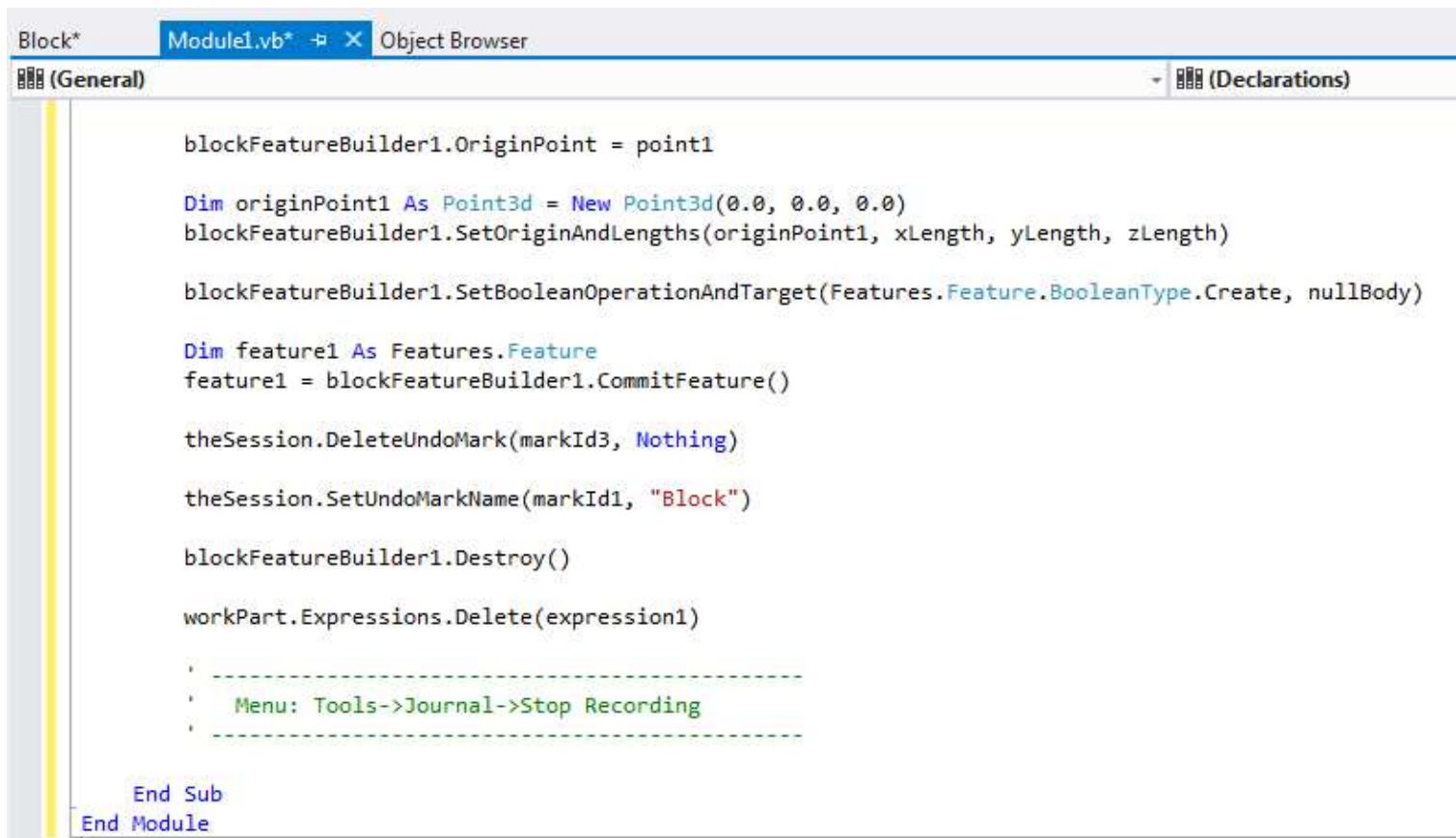
`SetOriginAndLengths(originPoint As NXOpen.Point3d, lengthExpression As String, widthExpression As String, heightExpression As String)`
Create a block by setting the origin and the block length, width, and height.
lengthExpression: Block length in the WCS x direction
NOTE: The full Unicode character set is not supported for this parameter.

The `xLength` property is highlighted in the Object Browser, showing its type: `Dim xLength As String`.

A red box highlights the following code snippet in the background:

```
Dim xLength As String  
Dim yLength As String  
Dim zLength As String  
  
xLength = InputBox("Enter A Value", "Set X Length of the Block", "100.00")  
yLength = InputBox("Enter A Value", "Set Y Length of the Block", "100.00")  
zLength = InputBox("Enter A Value", "Set Z Length of the Block", "100.00")
```

Values entered in the Inputbox to the function that generates the Box



```
Block*  Module1.vb*  Object Browser
(General)  (Declarations)

    blockFeatureBuilder1.OriginPoint = point1

    Dim originPoint1 As Point3d = New Point3d(0.0, 0.0, 0.0)
    blockFeatureBuilder1.SetOriginAndLengths(originPoint1, xLength, yLength, zLength)

    blockFeatureBuilder1.SetBooleanOperationAndTarget(Features.Feature.BooleanType.Create, nullBody)

    Dim feature1 As Features.Feature
    feature1 = blockFeatureBuilder1.CommitFeature()

    theSession.DeleteUndoMark(markId3, Nothing)

    theSession.SetUndoMarkName(markId1, "Block")

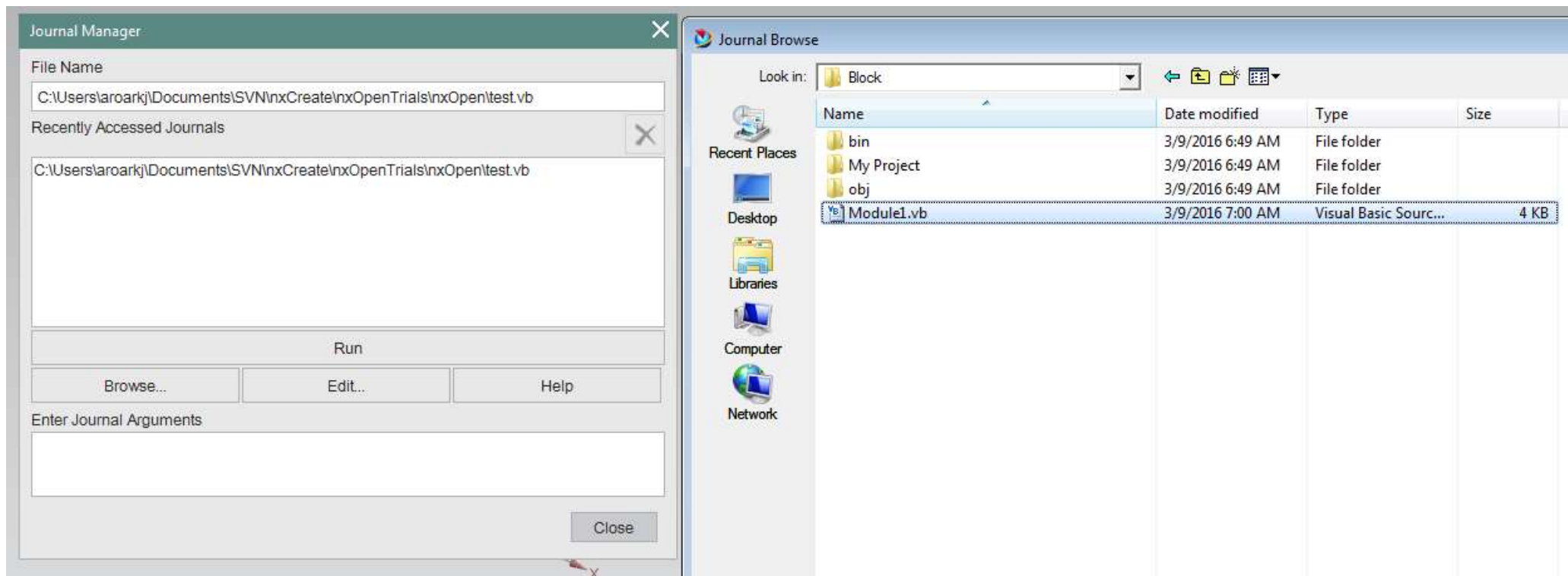
    blockFeatureBuilder1.Destroy()

    workPart.Expressions.Delete(expression1)

    ' -----
    '   Menu: Tools->Journal->Stop Recording
    ' -----

End Sub
End Module
```

Replay the modified Journal



Enter values in the Inputbox
The Journal Generates new Blocks.

