Measurement Studio NI-DAQmx Projects for .NET 4.5.1

2025-03-20

Contents

Measurement Studio NI-DAQmx Projects

This section provides detailed information about the NI Measurement Studio tools that integrate into Visual Studio and help you build measurement and automation applications that use the NI-DAQmx .NET class library.

Note You must have NI Measurement Studio installed to use these features.

NI-DAQmx is a task-based, object-oriented architecture with .NET class library interfaces. In addition, Measurement Studio provides a .NET DAQmx component model and support for generating .NET DAQ components that are built on the DAQmx component model. Measurement Studio .NET DAQ components provide several common operations that are appropriate for the measurement and data type of your configured DAQmx task.

Related tasks:

- Adding Custom Functionality

Related information:

- DAQmx.ComponentModel Namespace

NI DAQ Assistant

To simplify the creation of NI-DAQmx projects, NI provides the NI DAQ Assistant. The DAQ Assistant helps you write code to measure or generate data with NI-DAQmx devices. The DAQ Assistant provides a user interface within the Visual Studio environment. You use the DAQ Assistant user interface to interactively configure measurement tasks, channels, and scales. The DAQ Assistant generates a Visual Basic .NET or Visual C# class, referred to as a DAQmx Task class. The DAQmx Task class includes code that programmatically configures the measurement task that you created interactively. For .NET projects, the DAQ Assistant also generates a .NET DAQ

component that uses the generated DAQmx Task class.

Note To use the DAQ Assistant, your DAQ device must be connected to your system and you must be able to communicate with your DAQ device.

The code generated by the DAQ Assistant uses the Measurement Studio DAQmx .NET class library. You also can use these class libraries to programmatically develop a DAQ project.

Related information:

- Task Class

.NET DAQ Components

To use a DAQ component in a Measurement Studio .NET project, you must first create the DAQmx task within the project. You can create a DAQmx task in the following ways:

- Using the **Add New Item** dialog.
- Directly instantiating the component.

Related tasks:

- Creating a .NET DAQ Component with Add New Item
- Creating a .NET DAQ Component Programmatically

Creating a .NET DAQ Component with Add New Item

1. Open the project in which you want to create a DAQ component.
2. Select **Project** » **Add New Item** to launch the Add New Item dialog box.
3. In the Categories pane, select **Visual Basic Items** if you are using Visual Basic .NET. Select **Visual C# Project Items** if you are using Visual C#.
4. In the Templates pane, select **NI DAQ Component**.
5. Specify a name for the DAQmx task file and click **Add**. If you are creating a project task, the name you choose for the file becomes the name of your task class.
6. In the Add DAQ Component Dialog box, specify one of the following options to

create your DAQmx task:

- **Create a new project task**—This option creates a new DAQmx task in your project. The settings that you specify are not saved in Measurement & Automation Explorer (MAX) and only affect the task in the project.
- **Create a new MAX task**—This option creates a new DAQmx task in MAX. The settings that you specify are saved in MAX.
- **Create a reference to a MAX task**—This option lets you reference a DAQmx task that already exists in MAX from your current project. Any changes that are saved to this task are saved in MAX.
- **Copy a MAX task to a project task**—This option lets you create a copy of a DAQmx task that already exists in MAX and copy it to your current project. Any changes that you make to this task are not saved in MAX and only affect the copied task in the project.
- **Copy an existing .mxb**—This option lets you create a copy of a task that is specified by an existing `.mxb` file in another project and copy it into the current project.

7. Click **Finish**.
8. If you selected an option to create a new task, the wizard launches the DAQ Assistant. Select the measurement type for the new task in the DAQ Assistant.
9. Select the channel or channels to add to the task. Click **Finish**.

After you complete these steps, the wizard adds an `.mxb` file to the project and opens the task configuration editor. You use the configuration editor to configure the DAQmx task class. If the DAQmx task is a project task, the `.mxb` file stores the DAQmx task configuration information. If the DAQmx task is a MAX task, the `.mxb` file stores the MAX task name. The wizard adds references in the project to appropriate National Instruments class libraries.

Note To view these references, the `.mxb` file, and its source code in Visual Basic .NET, select **Show All Files** in the Solution Explorer toolbar. This step is not necessary in Visual C#.

The wizard also adds a partial class for the DAQ component which you can use to extend your application. This partial class has a `.User.vb` suffix for Visual Basic .NET projects and a `.User.cs` suffix for Visual C# projects.

Note The DAQmx task class and DAQ component are generated every time you change and save your configuration information, so any changes made in these classes are overwritten. Use the component partial class to add functionality to your DAQ component.

Save the `.mxb` file to generate source code. If the DAQmx task is a project task, the DAQ Assistant generates a DAQmx task class that contains your configuration settings and a DAQ component that uses the generated task. If the DAQmx task is a MAX task, the DAQ Assistant generates a DAQ component that loads the specified MAX task from MAX.

After you create the .NET DAQ component, you can use the DAQ Assistant task configuration editor to configure the DAQmx task of the component.

Related concepts:

- Task Configuration Editor
- .NET DAQ Components
- Custom Methods and Properties

Related information:

- Task Class

Creating a .NET DAQ Component Programmatically

You can use a .NET DAQ component programmatically by directly creating an instance of the component and calling its methods and handling its events. The specific methods and events that you use differ depending on the type of task you configured. The following are task types and corresponding DAQ component members that are commonly used.

Finite Input

Finite input DAQ components derive from FiniteInputDaqComponent<TReader,TData>.

1. Create an instance of a finite input component.

2. Call Read to read the data.

```
VB.NET

Dim component As MyFiniteInputComponent = New MyFiniteInputComponent()
Dim data() As Double = component.Read()
```

```
C#

MyFiniteInputComponent component = new MyFiniteInputComponent();
double [] data = component.Read();
```

Related concepts:

- Task Configuration Editor
- .NET DAQ Components

Finite Output

Finite output DAQ components derive from FiniteOutputDaqComponent<TWriter,TData>.

1. Create an instance of a finite output component.
2. Call WriteAsync to write the data.

```
VB.NET
Dim component As MyFiniteOutputComponent = New MyFiniteOutputComponent()
Dim data() As Double

'' Populate data …

component.WriteAsync(data)
```

```
C#
MyFiniteOutputComponent component = new MyFiniteOutputComponent();
double[] data;

// Populate data …

component.WriteAsync(data);
```

Continuous Input

Continuous input DAQ components derive from ContinuousInputDaqComponent<TReader,TData>.

1. Create an instance of a continuous input component.
2. Add an event handler for the DataReady event.
3. Call StartRead to start the continuous input data acquisition.

```
VB.NET
Dim component As MyContinuousInputComponent = New MyContinuousInputComponent()
AddHandler component.DataReady, AddressOf OnDataReady
component.StartRead()

' …

Sub OnDataReady(ByVal sender As Object, ByVal e As
MyContinuousInputComponentDataReadyEventArgs)
Dim data() As Double = e.GetData()
' …
End Sub
```

```
C#
MyContinuousInputComponent component = new MyContinuousInputComponent();
component.DataReady += OnDataReady;

// …

void OnDataReady(object sender, MyContinuousInputComponentDataReadyEventArgs e)
{
double[] data = e.GetData();
// …
}
```

Continuous Output

Continuous output DAQ components derive from ContinuousOutputDaqComponent<TWriter,TData>.

1. Create an instance of a continuous output component.
2. Add an event handler for the GenerateData event.
3. Call StartWrite to start the continuous output data acquisition.

```
VB.NET

 Dim component As MyContinuousOutputComponent = New MyContinuousOuptutComponent()
AddHandler component.GenerateData, AddressOf OnGenerateData
component.StartWrite()

' …

Sub OnGenerateData(ByVal sender As Object, ByVal e As
MyContinuousOutputComponentGenerateDataEventArgs e)
Dim data() As Double

' Populate data …

e.SetData(data)
End Sub
```

```
C#

MyContinuousOutputComponent component = new MyContinuousOutputComponent();
component.GenerateData += OnGenerateData();
component.StartWrite();

// …

void OnGenerateData(object sender, MyContinuousOutputComponentGenerateDataEventArgs
e)
{
double[] data;

// Populate data …

e.SetData(data);
}
```

Regenerative Output

Regenerative output DAQ components derive from RegenerativeOutputDaqComponent<TWriter,TData>.

1. Create an instance of a regenerative output component.
2. Call StartWrite to start the continuous input data acquisition.

```
VB.NET
Dim component As MyRegenerativeOutputComponent = New
MyRegenerativeOutputComponent()
Dim data() As Double

'' Populate data …

component.StartWrite(data)
```

```
C#
MyRegenerativeOutputComponent component = new MyRegenerativeOutputComponent();
double[] data;

// Populate data …

component.StartWrite(data);
```

Related information:

- FiniteInputDaqComponent Class
- FiniteOutputDaqComponent Class
- ContinuousInputDaqComponent Class
- ContinuousOutputDaqComponent Class
- RegenerativeOutputDaqComponent Class

Task Configuration Editor

After you create a .NET DAQ component within a project, the DAQ Assistant task configuration editor launches automatically inside Visual Studio. Use the DAQ Assistant task configuration editor to configure the DAQmx task of the component.

Note To use the DAQ Assistant configuration editor, the DAQ device must be connected to the system, and you must be able to communicate with the DAQ device. You can also use simulation.

Use the configuration editor to configure settings such as scaling, timing, ranges, and triggering. Click **Test** to open the test panel and test the task.

Note Refer to the DAQ Assistant embedded help for detailed information about using the DAQmx task configuration editor. To view the embedded help, click **Show Help** in the DAQ Assistant.

Related tasks:

- Creating a .NET DAQ Component with Add New Item
- Creating a .NET DAQ Component Programmatically

Generating and Viewing Source Code

After you configure a .NET DAQ component and save the task, a DAQ component class and its supporting classes are generated and added to the project. The DAQ component provides several common operations that are appropriate for the measurement and data type of your configured DAQmx task. If the configured DAQmx task is a project task, a DAQmx Task class is also generated that contains code that reflects your specified configuration settings.

Note When you use the DAQ Assistant configuration editor to make changes to the DAQmx task, the `.mxb` file is saved and the DAQ component code is regenerated. If you make changes directly to the generated source file, those changes are lost when the source code for the `.mxb` file is regenerated. To enhance or extend the code that the DAQ Assistant generates, you can use the component partial class that the DAQ Component wizard adds to your project. This partial class has a `.User.vb` suffix for Visual Basic .NET projects and a `.User.cs` suffix for Visual C# projects.

The DAQ component class code is contained in a `.vb` or `.cs` source code file in the project. This source code file has the same base name as the `.mxb` file, but has an extension that corresponds to the language of the source code. Complete the following steps to view the source code of the component.

1. In Visual Basic .NET, select **Show All Files** in the Solution Explorer toolbar. This step is not necessary in Visual C#.
2. In the Solution Explorer, expand the `.mxb` file node.
3. Double-click the `.vb` or `.cs` file to view the code. If the DAQmx task is a project

task, the generated task configuration code has the same name as the `.mxb` file. The DAQ component has the same name as the `.mxb` file and a Component suffix. For example, if you have an `.mxb` file that represents a project task, the DAQmx task configuration code class is named `MyTask`, and the DAQ component class is named `MyTaskComponent`.

4. Right-click the `.User.vb` file in Visual Basic .NET or `.User.cs` file in Visual C# and select **View Code** to view or edit the code for your component partial class.

The code is automatically regenerated every time you change and save the `.mxb` file. To explicitly regenerate the source code, right-click the `.mxb` file and select **Run Custom Tool**.

Related tasks:

- Generating a .NET DAQ Component User Interface

Adding Custom Functionality

You can enhance and extend the code that the DAQ Assistant generates through the component partial class that the DAQ Component wizard adds to your project. For example, you may want to add functionality to your DAQ component that leverages capabilities of your device that are not supported by the DAQ Assistant, such as advanced timing or triggering settings. You can also add methods and properties to the DAQ Component class by making changes to the `.User.cs` file or the `.User.vb` file.

Note The DAQ Assistant code generation changes do not affect the `.User.cs` or `.User.vb` files. You use the `.User.cs` and `.User.vb` files to customize the DaqComponent class.

1. In Visual Basic .NET, select **Show All Files** in the Solution Explorer toolbar. This step is not necessary in Visual C#.
2. In the Solution Explorer, expand the **.mxb** file node.
3. Right-click the `.User.vb` file in Visual Basic .NET or `.User.cs` file in Visual C#, and select **View Code** to edit the code for your component partial class.

The `.User.vb` and `.User.cs` files always include the following three methods:

- Initialize—Initialize is called when the component is created. Place code in Initialize to perform any required custom initialization. For example, you can use Initialize to set up file I/O or database connections.
- Dispose—Use this method to Dispose any resources that might have been initialized in the Initialize method.
- OnTaskCreated—Use OnTaskCreated to perform any custom DAQmx task configuration, such as setting properties that might not be exposed in the DAQ Assistant.

Related concepts:

- Measurement Studio NI-DAQmx Projects

Related information:

- DaqComponent Class
- Initialize Method
- Dispose Method
- OnTaskCreated Method

OnTaskCreated Examples

The following examples show how you can use the OnTaskCreated method to customize the DAQ Component generated by the DAQ Assistant.

```
VB.NET

Protected Overrides Sub OnTaskCreated(ByVal e As EventArgs)
  MyBase.OnTaskCreated(e)

  ' Configure a sample clock digital filter
  Task.Timing.SampleClockDigitalFilterEnable = True
  Task.Timing.SampleClockDigitalFilterMinimumPulseWidth = 0.01
End Sub
```

```
C#
```

```
protected override void OnTaskCreated(EventArgs e)
{
  base.OnTaskCreated(e);

  // Configure a sample clock digital filter
  Task.Timing.SampleClockDigitalFilterEnable = true;
  Task.Timing.SampleClockDigitalFilterMinimumPulseWidth = 0.01;
}
```

Related information:

- OnTaskCreated method

Custom Methods and Properties

In addition to customizing the DAQ Component through existing methods, you can add your own custom methods and properties.

The following example demonstrates how to modify the `.User.vb` and `.User.cs` files for code that is generated by the DAQ Assistant for an analog input operation. You can add a public method in the DaqTaskComponent class called SetTaskRange that sets the analog input range of the first channel in the task. The code also includes a read-only property that exposes the AIChannel object associated with the task on the DAQ Component.

VB.NET

```
' Use this file to enhance and extend the component generated by the DAQ Assistant.
'
' This file is never regenerated by the DAQ Assistant, so any changes you make here
will not be overwritten
' if you modify your DAQmx task with the DAQ Assistant configuration editor.


Imports System
Imports NationalInstruments
Imports NationalInstruments.DAQmx
Imports NationalInstruments.DAQmx.ComponentModel

' This partial class extends the class defined by the DAQ Assistant in DaqTask.vb
```

```
Partial Public Class DaqTaskComponent

Private aiChannel As AIChannel

  ' Initializes a new instance of the component.
  Protected Overrides Sub Initialize()
    MyBase.Initialize()

    ' TODO: Perform custom initialization here.
    '       This method is called when the component is created.
  End Sub

  ' Clean up any resources being used.
  Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    ' TODO: Perform custom clean-up here.
    '
    ' Refer to "Implementing a Dispose Method" in the .NET Framework documentation
for more
    ' information on the Disposable Pattern.
    MyBase.Dispose(disposing)
  End Sub

  ' Raises the TaskCreated event.
  Protected Overrides Sub OnTaskCreated(ByVal e As EventArgs)
    aiChannel = Task.AIChannels(0)
    Task.Control(TaskAction.Verify)

    MyBase.OnTaskCreated(e)
  End Sub

#Region "User Added Code"

  Public Sub SetTaskRange(ByVal minRange As Double, ByVal maxRange As Double)
    If aiChannel Is Nothing Then
      OnTaskCreated(Nothing)
    End If

    If minRange > maxRange Then
      Exit Sub
    End If

    aiChannel.RangeLow = minRange
    aiChannel.RangeHigh = maxRange

  End Sub
```

```
  Public ReadOnly Property Channel() As AIChannel
    Get
      If aiChannel Is Nothing Then
        OnTaskCreated(Nothing)
      End If

      Return aiChannel
    End Get
  End Property
#End Region

End Class
```

C#

```
// Use this file to enhance and extend the component generated by the DAQ
Assistant.
//
// This file is never regenerated by the DAQ Assistant, so any changes you make
here will not be overwritten
// if you modify your DAQmx task with the DAQ Assistant configuration editor.

using System;
using NationalInstruments;
using NationalInstruments.DAQmx;
using NationalInstruments.DAQmx.ComponentModel;

namespace DaqApplication
{
  // This partial class extends the class defined by the DAQ Assistant in
DaqTask.cs
  public partial class DaqTaskComponent
  {
      private AIChannel aiChannel;

      protected override void Initialize()
      {
          base.Initialize();

          // TODO: Perform custom initialization here.
          //       This method is called when the component is created.
      }

      protected override void Dispose(bool disposing)
      {
```

```
            // TODO: Perform custom clean-up here.
            //
            // Refer to "Implementing a Dispose Method" in the .NET Framework
documentation for more
            // information on the Disposable Pattern.

            base.Dispose(disposing);
        }

        protected override void OnTaskCreated(EventArgs e)
        {
            aiChannel = Task.AIChannels[0];
            Task.Control(TaskAction.Verify);

            base.OnTaskCreated(e);
        }

        public void SetTaskRange(double minRange, double maxRange)
        {
            if(aiChannel == null)
                OnTaskCreated(null);
            aiChannel.RangeLow = minRange;
            aiChannel.RangeHigh = maxRange;
        }

        public AIChannel Channel
        {
            get
            {
                if (aiChannel == null)
                    OnTaskCreated(null);

                return aiChannel;
            }
        }
    }
}
```

Related tasks:

- Creating a .NET DAQ Component with Add New Item

Related information:

- AIChannel Class

- Task Class

Generating a .NET DAQ Component User Interface

The DAQ UI Generation Wizard creates a user interface that is appropriate for your configured DAQ component and integrates it in a Windows Forms Form class in your project.

To generate a user interface for your DAQ component, complete the following steps:

1. Open the project that contains your DAQ component.
2. Open the Windows Forms Form class to which you want to add a user interface for your DAQ component in the Windows Forms designer.
3. Select **View** » **Toolbox** to open the Visual Studio toolbox, and expand the Measurement Studio tab.
4. Drag and drop the **DaqComponent** toolbox item from the toolbox to the Windows Forms designer surface. The Add DAQ Component dialog box opens.
5. In the Add DAQ Component dialog box, select the DAQ component that you want to add to the form and click **Finish**.
6. Right-click the DAQ component in the component tray at the bottom of the designer and select **Generate UI**. The Generate DAQ UI dialog box opens.
7. In the Generate DAQ UI dialog box, you can specify control and event handler names, customize how code is generated, and preview the user interface and code. Click **Finish**.

After you complete these steps, a user interface is generated into your form and event handler code is added to your form class. The event handler contains code to call the appropriate methods on the component, which read and write data and update the user interface to present this data.

Note Generating the user interface multiple times from the same DAQ component may cause the generated code to behave unexpectedly upon execution. For best results, remove any existing generated DAQ component user interface controls from your form and any event handler code before regenerating the user interface.

Related tasks:

- Generating and Viewing Source Code