Instructions for C# on 4-Series and/or VC-4

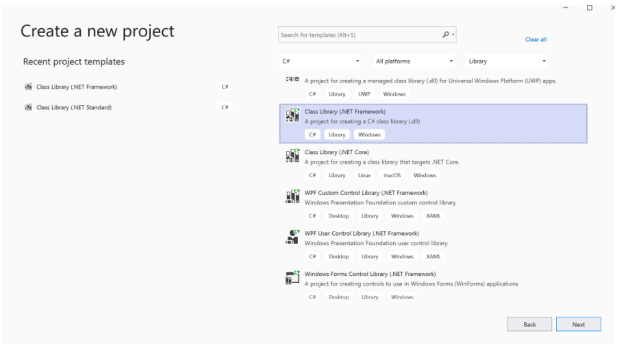
1 Create a New Project

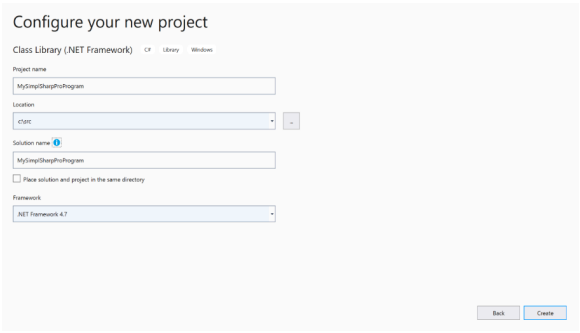
In your IDE of choice, simply create a new project of type “Class Library (.NET Framework)” and select version 4.7.

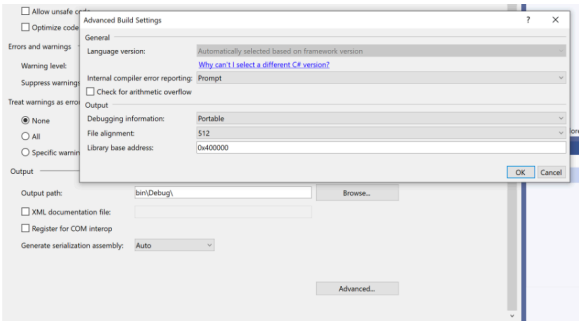
Edit your project properties and set the debug type to “Portable”

Here is what that looks like in some of the popular development environments:

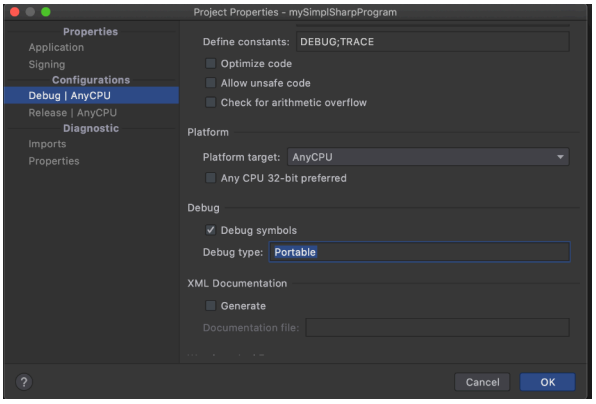
Microsoft Visual Studio 2019 (Windows)

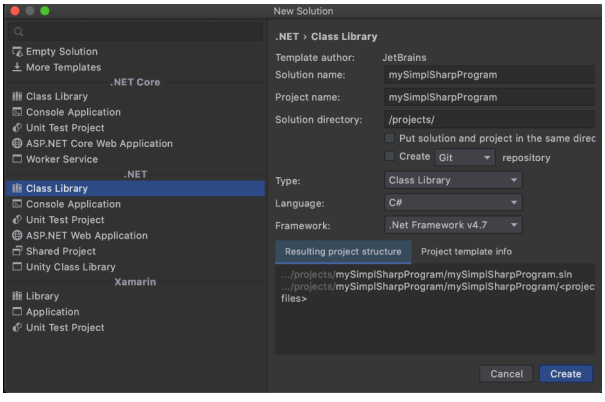






JetBrains Rider (OS X and Windows)





Making your Projects work on 4-Series and VC-4

2 Add Crestron Package

We have created NuGet Packages to make creating projects for VC-4 and 4-Series easier than ever. Once you have created your solution and your project, simply right-click on the project, and select "*Manage NuGet Packages*"

Warning: these packages create libraries compatible only with a 4-Series or VC-4. These projects will not run on a 3-Series Control System.

From there, browse for Crestron, and you'll see three packages. Install the one you need:

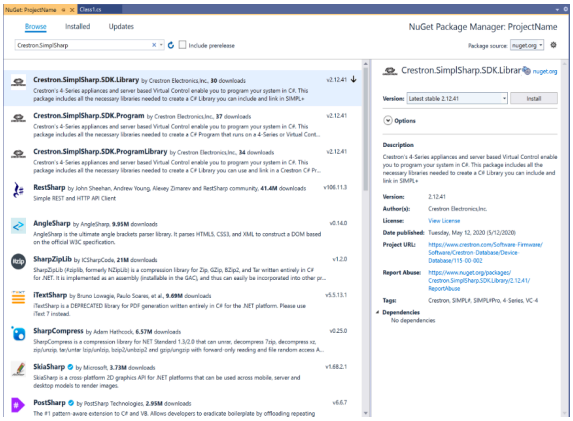
- **Crestron.SimplSharp.SDK.Library**
  This package is required to create a C# Library to be used with SIMPL+ or a Crestron C# Program. It will add references to all available Crestron SDK DLLs, and automatically create the CLZ on build.

- **Crestron.SimplSharp.SDK.ProgramLibrary**
  This package is required to create a C# Library that can be used with a Crestron C# Program. It will add references to all available Crestron SDK DLLs. **This package will pull in Crestron.SimplSharp.SDK.Library as a dependency.**

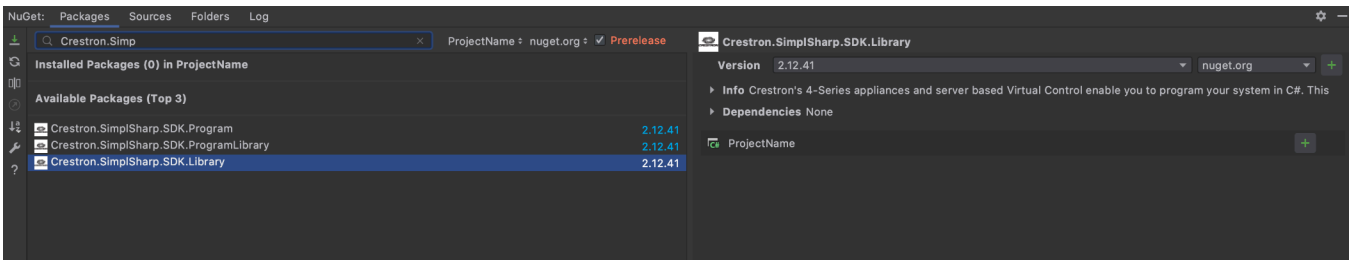
- **Crestron.SimplSharp.SDK.Program**
  This package is required to create a Crestron C# Program. It will add references to all available Crestron SDK DLLs, and automatically create the CPZ on build. **This package will pull in Crestron.SimplSharp.SDK.ProgramLibrary and Crestron.SimplSharp.SDK.Library as dependencies.**

Here is what those screens look like in some of the popular development environments:

Microsoft Visual Studio 2019 Windows



JetBrains Rider (OS X and Windows)



Using the NuGet Packages

Notice About Upgrading from NuGet Package Version 2.12.41

If you are upgrading your project from NuGet package **v2.12.41**, please first remove the old Crestron NuGet package(s) from your project(s), and **also remove any project references to Crestron DLLs and SIMPLSharpPro.EXE**. This is a one-time operation, only applicable when upgrading from version v2.12.41.

Using Crestron SDK References in your Project

To ensure that upgrading to newer versions of the Crestron NuGet packages in the future also updates all references to Crestron's SDK DLLs, we automatically add a reference to all available DLLs to a project as part of the NuGet package install. We found out that any references added manually (even to DLLs that are part of a package), are not being upgraded to newer versions when we upgrade.

This will create a larger CPZ/CLZ, but will not affect runtime RAM utilization. If you want to reduce the CPZ/CLZ size you can mark unused references as "do not copy."

If you do delete a Crestron SDK Reference from the project, please **do not** manually add it back in (that will prevent it from being upgraded in the future), but rather uninstall / re-install all the Crestron NuGet packages, or upgrade to a newer version if available. This means that if you installed Crestron.SimplSharpPro.SDK.Program, you will have to also remove the Crestron.SimplSharpPro.SDK.ProgramLibrary and Crestron.SimplSharpPro.SDK.Library packages.

Upgrading Crestron SDK References in your Project

Each individual project will maintain a reference the exact version of the NuGet packages you last used. This will allow you to make a code change to an older project, without worrying about changes in the Crestron SDK. If you upgrade one project, other projects are not affected!

In a single solution, Crestron recommends that all projects / references are built against the same versions of the Crestron SDK, to avoid compatibility issues.

Upgrading the version of the packages could not be easier: simply go back to the NuGet Management screen, and select the **top level package** (i.e. if your project is a Crestron program, select Crestron.SimplSharp.SDK.Program, not ProgramLibrary or Library) and upgrade it to the version you'd like. This will automatically update any of the lower level packages it needs. The system will automatically update all references and required files to the correct version. Do this for all projects / dependencies in your solution

JetBrains Rider IDE
Debugging C# on 4-Series

1 Setting Up The Control System

Make sure your control system and Rider IDE have the exact same version of code. (Rebuild to be sure).

After you've done a build, use SFTP, and upload the ZIP/CPZ File that the Post-Build step created to the control system. Log into the control system's SSH console and type something like this. You can change the program number and port number to your liking.

MC4>**PROGLOAD -P:1 -D**

One of these two:

- MC4>**DEBUGPROGRAM -P:1 -Port:50000 -IP:0.0.0.0**
  (program will fully start, and you can attach at any later moment)
- MC4>**DEBUGPROGRAM -P:1 -Port:50000 -IP:0.0.0.0 -S**
  (program will wait for you to attach the debugger, so you can debug the constructor(s))

MC4>**PROGRESET -P:1**

When you are done debugging, please follow these steps, **before detaching from your processor**:

MC4>**STOPPROGRAM -P:1**

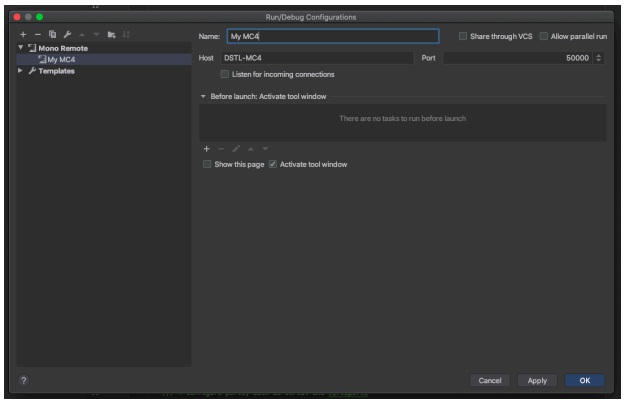
MC4>**DEBUGPROGRAM -P:1 -C**

[Safe to detach your debugger]

MC4>**PROGRESET -P:1**

2 Attaching via Rider

In Rider, go to the "Run" menu, and select "Edit Configurations." Click the "+" and select "Mono Remote." Fill in a friendly name, the IP Address / Hostname of your control system, and the port you specified in the console command from section 1.



Click "OK."

Go to "Run" menu, and select "Debug." A window will pop up that lets you select a profile. Pick the profile you just created.

Warning

Unless you start a room in debug mode, you cannot debug it. Follow the instructions, and restart the room. Once you detach, or stop debugging, you cannot re-attach until you restart the program.

Visual Studio 2017 / 2019 on Windows
Debugging C# on 4-Series

1 Prerequisites

In order to debug the 4-Series using Visual Studio 2017 / 2019 on Windows, you need to install a VS Extension called "VSMonoDebugger"

2 Setting Up The Control System

Make sure your control system and Visual Studio IDE have the exact same version of code. (Rebuild to be sure). After you've done a build, use SFTP, and upload the ZIP/CPZ File that the Post-Build step created to the control system. Log into the control system's SSH console and type something like this. You can change the program number and port number to your liking.

MC4>**PROGLOAD -P:1 -D**

One of these two:

- MC4>**DEBUGPROGRAM -P:1 -Port:50000 -IP:0.0.0.0**
  (program will fully start, and you can attach at any later moment)
- MC4>**DEBUGPROGRAM -P:1 -Port:50000 -IP:0.0.0.0 -S**
  (program will wait for you to attach the debugger, so you can debug the constructor(s))

MC4>**PROGRESET -P:1**
When you are done debugging, please follow these steps, **before detaching from your processor**:

MC4>**STOPPROGRAM -P:1**

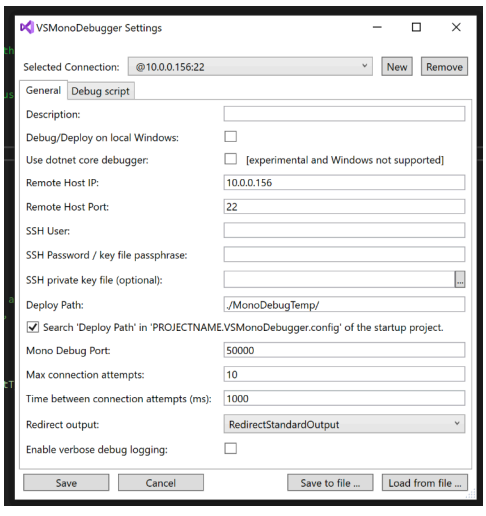
MC4>**DEBUGPROGRAM -P:1 -C**
[Safe to detach your debugger]

MC4>**PROGRESET -P:1**

3 Attaching via VisualStudio 2017 / 2019

In Visual Studio, go to the "Extensions" Menu, and select "Mono," followed by "Settings." Fill in the Remote Host IP, and the Mono Debug Port. Click "Save."

Once this is done, go to "Extensions," "Mono," and select "Attach to mono debugger." This should allow you to debug your program.



Warning

Unless you start a room in debug mode, you cannot debug it. Follow the instructions, and restart the room. Once you detach, or stop debugging, you cannot re-attach until you restart the program.

Debugging on VC-4

1 Prerequisites

The same prerequisites as debugging for 4-Series apply here. Secondly, you'll need shell access to the VC-4 server to edit the debug configuration file.

2 Setting Up Debug Configuration File

Using your editor of choice, edit the file `/opt/crestron/virtualcontrol/conf/debug.conf`

Add a line at the bottom of the file with the RoomID (all caps), and the port number. Like so:

```
MYROOM,50000
```

Save the file. Now, simply stop, and start the room you'd like to debug. On VC-4, rooms in debug mode will start not-suspended, meaning they'll fully start up. Attach using VS2017/2019 or Rider, the same way you do for 4-Series.

Be aware that your server's firewall needs to allow connections on the port number; otherwise debugging won't work.

When you are done debugging, please follow these steps, **before detaching from your processor**:

- Stop the room from the VC-4 UI
- Remove the entry from the debug.conf file
- Start the room from the VC-4 UI

Warning

Unless you start a room in debug mode, you cannot debug it. Follow the instructions, and restart the room. Once you detach, or stop debugging, you cannot re-attach until you restart the program.