


SUPrema

SUPrema

How to get started with BioStar 2 SDK

Compiled by
George Cartlidge



What is the API or SDK, and what is the difference?

API stands for Application Programming Interface, where SDK stands for Software Development Kit.

An API is a set of functions that allows you to interface with an existing bit of software (I.E BioStar 2), This means any interaction with Suprema devices will need an installation of BioStar 2 to be already installed and that you are actually interacting with the BioStar 2 server and not the devices. Starting with BioStar 2.7.10, there is a new version of the API that includes more functions.

An SDK is a collection of tools that you can use to develop your own applications using a particular framework/platform. Regarding the BioStar 2 SDK, this will be the BS_SDK_V2.dll file. This .DLL contains many APIs that can be used with C++ & C# programming languages. There is also the new G-SDK which functions along the same principal but is based upon gRPC which can support more languages than the existing SDK. Using the SDK, you cut BioStar 2 out completely and directly interact with the device.

Comparing both, the API will be used where you are happy for BioStar 2 to handle most of the work, BioStar 2 would be storing the fingerprint and user data; The API lets you use BioStar 2 from inside another application, so it does not look like BioStar 2. The SDK will be used when you want to connect directly to the device from your own application; Your application would have to handle the connection and the storage of data. The SDK will also allow you to use specific API's that are not available in the BioStar 2 API.

SDK

In the next part of this guide, I will be going through the standard device SDK as it currently supports more features than the new G-SDK, the G-SDK will be covered in a future guide (Note: The G-SDK will support more features than the device SDK eventually). As they can use the same languages and devices, there will be a lot of overlap at least, I will also be looking at the C# examples.

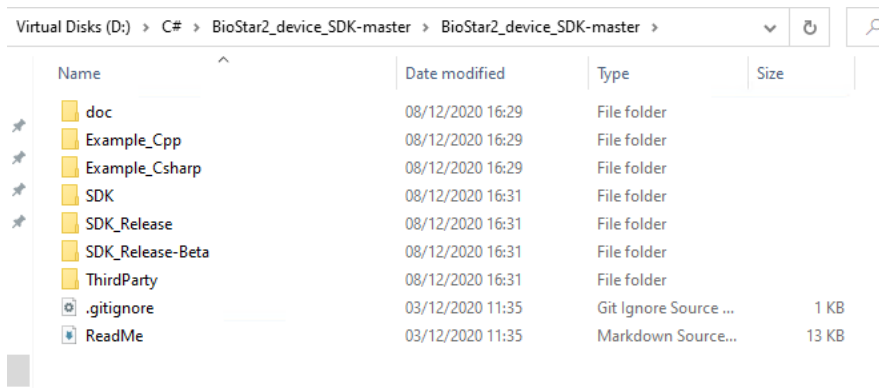
Loading through Visual Studio

So! The first thing you will need is a copy of your favourite code compiler, I will use Visual studio, if you are downloading Visual Studio for the first time, Download Visual Studio Community 2019 (Not Visual Studio Code, with that, you'll be able to view the code, but not Build and interact with it).

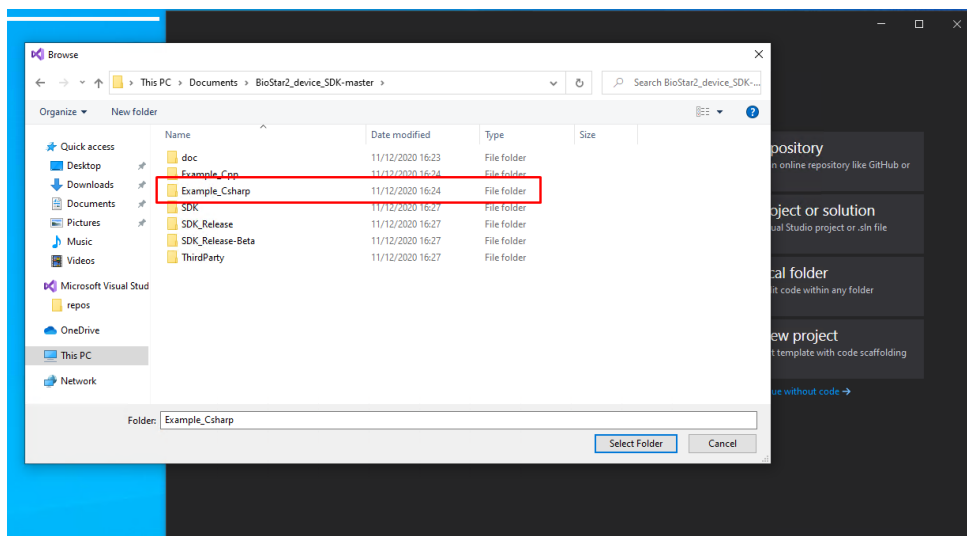
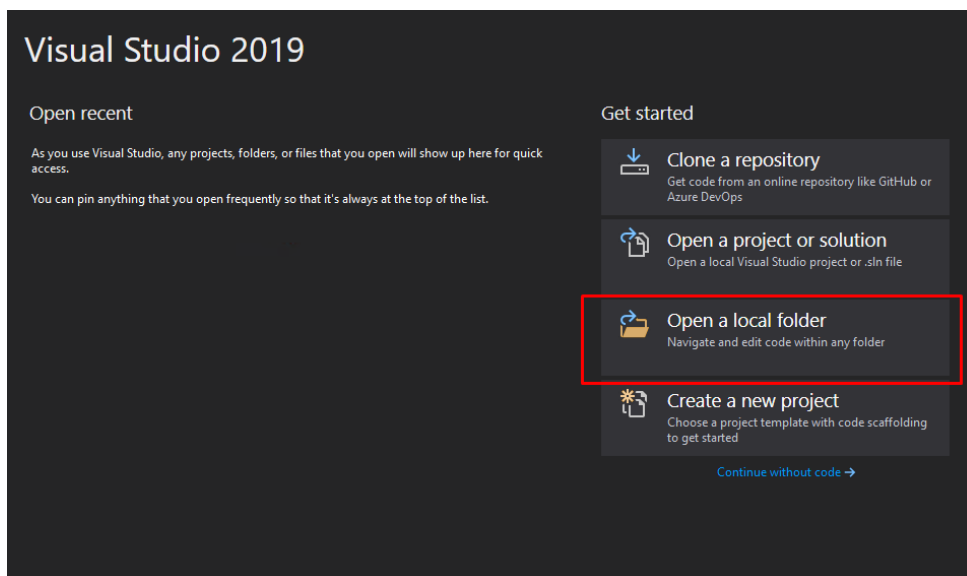
For both C++ and C#, examples are included of how to interact with the SDK, for the C# Example this is the entire BSDemo project, which is a very good set of tools that you can use to troubleshoot Suprema devices (Its under 5MB too!).

Second, head over to the Suprema GitHub & download a copy of the SDK components (https://github.com/supremainc/BioStar2_device_SDK)

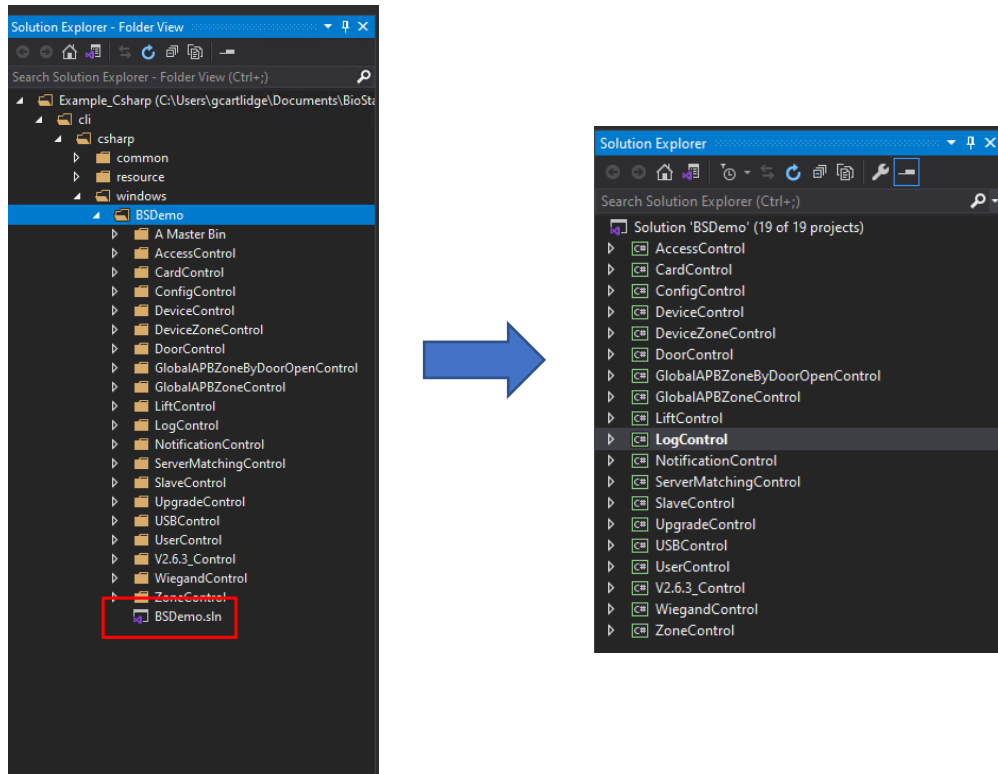
This will give you a zip 'BioStar2_device_SDK-master', extract this. So it looks like the below.



We will first load through the BSDemo project, just to get an overview. From the Visual Studio window, click 'Open a local folder' and browse to the 'Example_CSharp' folder



Now we're getting somewhere! On the right-hand side, it will now load the folder view, navigate so it looks like the picture below and select the 'BSDemo.sln' file to load the project.



The view on the right should appear, these are all the separate programs. Let's use 'Upgrade control' as a test, as it's the one I most often use (You can connect to devices and update their firmware).

Clicking this project will open the files view, the important one is the 'Program.cs', its looks very simple, but this is because it uses the other .cs files in the folder

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using Suprema;
6
7 namespace BSDemo
8 {
9     1 reference
10     class Program : UnitTest This means it is using the 'UnitTest.cs', This is what connects to the devices
11     {
12         private UpgradeControl uc = new UpgradeControl(); This means this is using the
13                                     'UpgradeControl.cs'
14                                     (Which can upgrade the FW)
15         23 references
16         protected override void runImpl(UInt32 deviceID)
17         {
18             uc.execute(sdkContext, deviceID, true);
19         }
20
21         0 references
22         static void Main(string[] args)
23         {
24             Program program = new Program();
25             program.Title = "Test for upgrading";
26             program.run();
27         }
28     }
29 }
```

Text Above:

Unit test = This Means it is using the 'UnitTest.cs', This is what connects to the devices#

UpgradeControl = This means it is using the 'UpgradeControl.cs' (Which can upgrade the FW)

I'll go into a view of the 'UpgradeControl.cs' code, The 'UnitTest.cs' is a bit bigger and not as easily viewable as to what each part does.

The screenshot shows the Visual Studio IDE with the file `UpgradeControl.cs` open. The code is for a class `UpgradeControl` in the `Suprema` namespace, which inherits from `FunctionModule`. The code is annotated with red, green, and blue boxes and text explaining different sections.

```

8
9 namespace Suprema
10 {
11     1 reference
12     public class UpgradeControl : FunctionModule
13     {
14         private const UInt32 MAX_PERCENT = 40;
15         private API.OnProgressChanged cbOnProgressChanged = null;
16
17         20 references
18         protected override List<KeyValuePair<string, Action<IntPtr, UInt32, bool>>> getFunctionList(IntPtr sdkContext, UInt32 deviceID, bool isMasterDevice)
19         {
20             List<KeyValuePair<string, Action<IntPtr, UInt32, bool>>> functionList = new List<KeyValuePair<string, Action<IntPtr, UInt32, bool>>>();
21             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Upgrade firmware", upgradeFirmware));
22             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Upgrade language pack", upgradeLanguage));
23             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Update background image", updateBackgroundImage));
24             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Update notice message", updateNoticeMessage));
25             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Update slide image", updateSlideImage));
26             functionList.Add(new KeyValuePair<string, Action<IntPtr, UInt32, bool>>("Update sound resource", updateSoundResource));
27
28             return functionList;
29         }
30
31         1 reference
32         public void upgradeFirmware(IntPtr sdkContext, UInt32 deviceID, bool isMasterDevice)
33         {
34             Console.WriteLine("Enter the path of firmware file which you want to upgrade.");
35             Console.Write(">>>> ");
36             string firmwarePath = Console.ReadLine();
37
38             if (!File.Exists(firmwarePath))
39             {
40                 Console.WriteLine("Invalid firmware path");
41                 return;
42             }
43
44             IntPtr firmwareData = IntPtr.Zero;
45             UInt32 firmwareDataLen = 0;
46
47             if (Util.LoadBinary(firmwarePath, out firmwareData, out firmwareDataLen))
48             {
49                 Console.WriteLine("Trying to upgrade firmware.");
50                 cbOnProgressChanged = new API.OnProgressChanged(firmwareProgressChanged);
51                 BS2ErrorCode result = (BS2ErrorCode)API.BS2_UpgradeFirmware(sdkContext, deviceID, firmwareData, firmwareDataLen, 0, cbOnProgressChanged);
52                 Marshal.FreeHGlobal(firmwareData);
53
54                 The above part is the important bit! This uses the 'BS2_UpgradeFirmware' API to send the data from 'firmwareData' to
55                 if (result != BS2ErrorCode.BS_SDK_SUCCESS) the device that matches the 'deviceID' variable that the main program forwards from 'UnitTest.cs'. Upgrading the device!
56                 {
57                     Console.WriteLine("Got error({0}).", result);
58                 }
59                 If the API doesnt return a 'BS_SDK_SUCCESS' message, this part mentions the error recieved!
60
61                 cbOnProgressChanged = null;
62             }
63         }
64     }
65 }

```

This section prompts the user for a selection of what they wish to do, the first option is 'Upgrade Firmware' and will forward to the 'upgradeFirmware' section.

This is the section that is loaded from above, pulling through the variables from above too

This section prompts the user for a 'Firmware Path' variable, IE Where the file is stored. This is then used in the below process.

This section uses a function from 'Util.cs' ('LoadBinary') to load the Firmware file and outputs it as the 'firmwareData' variable

The above part is the important bit! This uses the 'BS2_UpgradeFirmware' API to send the data from 'firmwareData' to

if (result != BS2ErrorCode.BS_SDK_SUCCESS) the device that matches the 'deviceID' variable that the main program forwards from 'UnitTest.cs'. Upgrading the device!

If the API doesnt return a 'BS_SDK_SUCCESS' message, this part mentions the error recieved!

Text Above:

RED: “Upgrade firmware” : This Section prompts the user for a selection of what they wish to do, the first option is ‘Upgrade Firmware’ and will forward to the ‘upgradeFirmware’ section.

YELLOW: upgradeFirmware : This is the section that is loaded from above, pulling through the variables from above too.

GREEN: ‘Firmware Path’: Rgus sectuib orinots tge yser fir a ‘Firmware Path’ variable, IE Where the file is stored. This is then used in the below process.

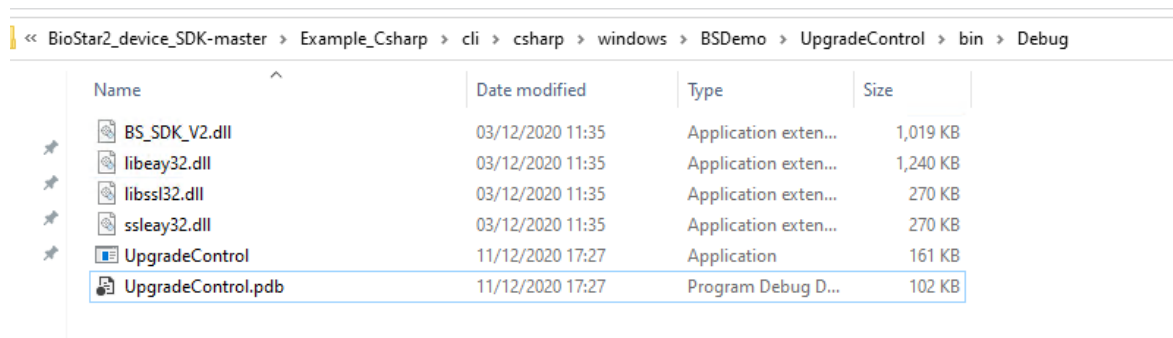
“Util.cs” “(Load Binary)”: This section uses a function from ‘Util.cs’ “(Load Binary)” to load the firmware file and outputs it as the ‘firmwareData’ variable.

BLUE: “UpgradeFirmware”: The above part is the important bit! This uses the ‘BS2_UpgradeFirmware’ API to send the data from ‘firmwareData’ to the device that matches the ‘deviceId’ variable that the pain program forwards from ‘UnitTest.cs’. Upgrading the device!

Building the program

While browsing any specific project, pressing 'Ctrl + B' will build the project, which means we get a shiny new .exe out of Visual Studio! As this is an already setup project, it shouldn't have a problem creating the program and will export to the bin\Debug folder of the UpgradeControl program.

'BioStar2_device_SDK-master\Example_Csharp\cli\csharp\windows\BSDemo\UpgradeControl\bin\Debug' if you are using the SDK folder like above.



The screenshot shows the Visual Studio File Explorer with the path: < < BioStar2_device_SDK-master > Example_Csharp > cli > csharp > windows > BSDemo > UpgradeControl > bin > Debug. The file list is as follows:

Name	Date modified	Type	Size
BS_SDK_V2.dll	03/12/2020 11:35	Application exten...	1,019 KB
libeay32.dll	03/12/2020 11:35	Application exten...	1,240 KB
libssl32.dll	03/12/2020 11:35	Application exten...	270 KB
ssleay32.dll	03/12/2020 11:35	Application exten...	270 KB
UpgradeControl	11/12/2020 17:27	Application	161 KB
UpgradeControl.pdb	11/12/2020 17:27	Program Debug D...	102 KB

Open the program and give it a test! (You may be prompted for firewall access)

You can say no to setting up SSL (This uses a secure certificate to talk to the device) and no to outputting the debug message (We don't need it)

Below I will search for the devices by pressing 1, it will list the devices, the BioStation2 Is closest to me so lets choose that. Now were connected to the device, it loads the code above! You can see the selection for upgrading firmware, it will then ask for a file, and send that file to the device.

A screenshot of my output is below:

```

Test for upgrading connected deviceID[547635862]
SDK version : 2.7.2.4
Do you want to set up ssl configuration? [Y/n]
>>> n
Do you want output debug message to file? [y/n]
>>> n
-----+
1. Search and connect device
2. Connect to device via Ip
3. Server mode test
-----+
How to connect to device? [2(default)]
>>> 1
Trying to broadcast on the network
[CB] Device[ 547803261] has been found.
[CB] Device[ 545394773] has been found.
[CB] Device[ 541617859] has been found.
[CB] Device[ 543714324] has been found.
[CB] Device[ 939261879] has been found.
[CB] Device[ 542342301] has been found.
[CB] Device[ 542194296] has been found.
[CB] Device[ 547635862] has been found.
[CB] Device[ 543229054] has been found.
-----+
0] ==> ID[ 541617859] Type[      BioEntry P2] Connection mode[DEVICE_TO_SERVER] Ip[   192.168.0.82] port[51211] Master/Slave[3]
1] ==> ID[ 542194296] Type[      FaceStation 2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.4] port[51211] Master/Slave[3]
2] ==> ID[ 542342301] Type[      FaceStation 2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.102] port[51211] Master/Slave[3]
3] ==> ID[ 543229054] Type[      BioEntry W] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.113] port[51211] Master/Slave[3]
4] ==> ID[ 543714324] Type[      FaceStation F2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.69] port[51211] Master/Slave[3]
5] ==> ID[ 545394773] Type[      Biolite N2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.156] port[51211] Master/Slave[3]
6] ==> ID[ 547635862] Type[      BioStation 2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.139] port[51211] Master/Slave[1]
7] ==> ID[ 547803261] Type[      FaceLite] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.184] port[51211] Master/Slave[1]
8] ==> ID[ 939261879] Type[      BioStation A2] Connection mode[SERVER_TO_DEVICE] Ip[   192.168.0.104] port[51211] Master/Slave[3]
-----+
Please, choose the index of the Device which you want to connect to. [-1: quit]
>>> 6
Trying to connect to device[547635862]
[CB] Device[ 547635862] has been connected.
>>> Successfully connected to the device[547635862].
-----+
1. Upgrade firmware
2. Upgrade language pack
3. Update background image
4. Update notice message
5. Update slide image
6. Update sound resource
7. Exit
-----+
What would you like to do?
>>> 1
Enter the path of firmware file which you want to upgrade.
>>> .

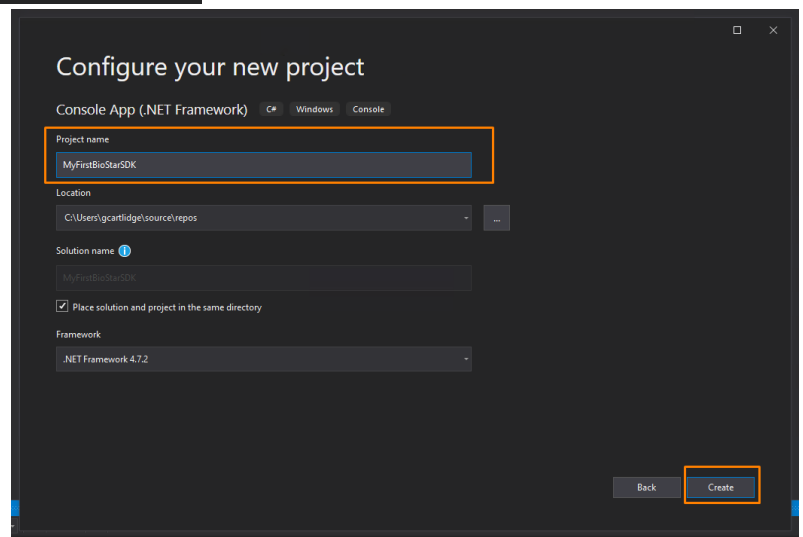
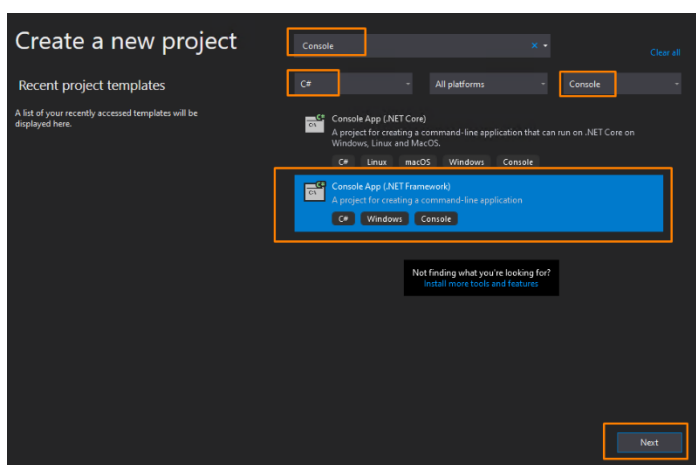
```

We have just run our first compiled program from visual studio!

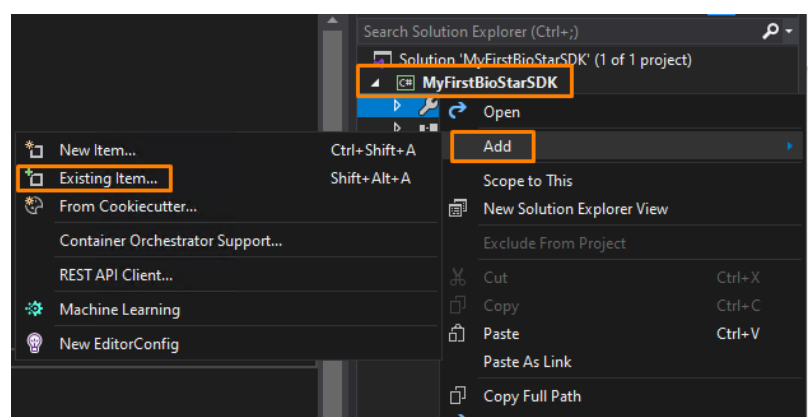
Final thoughts for making your own programs

The following requires a bit more understanding of Visual Studio and working with C#

Above, as the program was already made, it included the basics already. If you create an entirely new project, you will need to add the source files yourself. Create a new C# Project and it will look like the below:



This will generate a basic program, On the right-hand columns, right click *NameOfYourApp* and add an existing item.

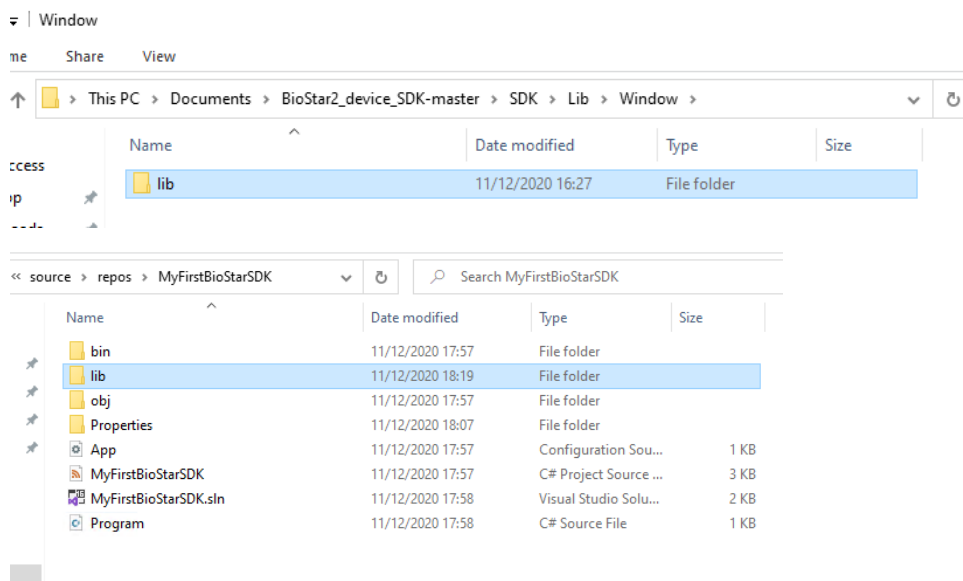


Browse to the SDK Folder and select the 'common' folder beneath 'csharp', You can include all the .cs files here, below is what they do:

- i. FunctionModule.cs : Display the functions which are implemented in BSDemo(SDK package demo source)
- ii. SEApi.cs : Lists APIs
- iii. SEEnum.cs : Lists enum
- iv. SFStruct.cs : Lists structure
- v. UnitTest.cs : Initialize the device and provides ways to connect to device.
- vi. Util.cs : Device I/O control

This will add them to the tree view. Next we will have to copy the SDK files to the Project folder, This will be the 'lib' from the SDK folder. Mine would be:

C:\Users\gcartlidge\Documents\BioStar2_device_SDK-master\SDK\Lib\Window\lib



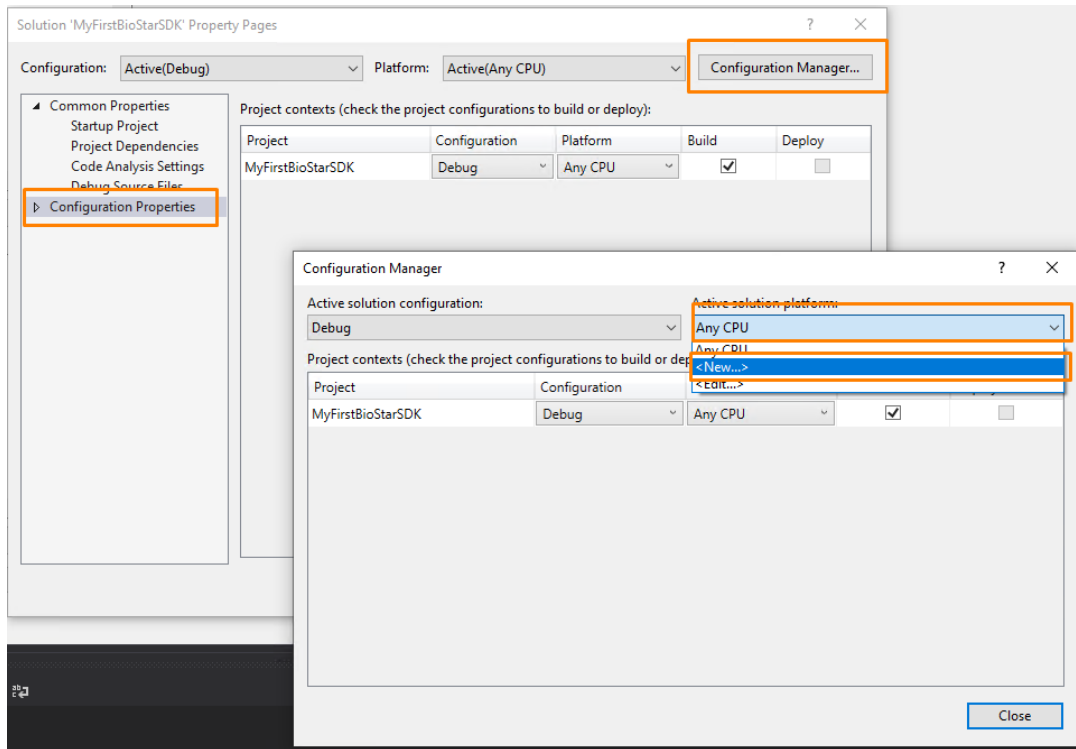
Next, Right click the project and go to properties and enter the below under 'Build Events' & 'pre-build event command line'

```
copy "$(ProjectDir)lib\$(PlatformTarget)\BS_SDK_V2.dll" "$(TargetDir)"
copy "$(ProjectDir)lib\$(PlatformTarget)\libey32.dll" "$(TargetDir)"
copy "$(ProjectDir)lib\$(PlatformTarget)\libssl32.dll" "$(TargetDir)"
copy "$(ProjectDir)lib\$(PlatformTarget)\ssleay32.dll" "$(TargetDir)"
```

For the last property, right click the very top of the tree

‘Solution “*ProjectName*”’ and select

Properties > Configuration > Configuration Manager

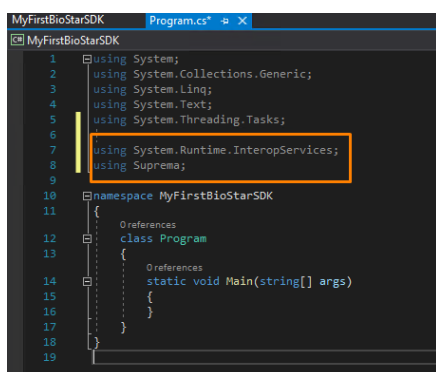


Press ‘OK’ on x64, then close and close.

Press Ctrl + B and there shouldn’t be any errors!

If there are any errors, it will be worth checking that the Build Events file names match the files in the ‘lib’ folder.

Next, add the namespace by adding the two lines below:



Suprema

To test, Building the below code should output the version of the SDK! (If your namespace name is also 'MyFirstBiostarSDK', If not, change the namespace name below.)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using Suprema;

namespace MyFirstBioStarSDK {
class Program {
static void Main(string[] args) {
//3.1.2 SDK version information check
IntPtr versionPtr = API.BS2_Version();
Console.WriteLine("SDK version : {0}", Marshal.PtrToStringAnsi(versionPtr));
Console.ReadKey();
} } }
```

That should be the basics on getting you setup to code,
Go forth and try your luck!

Note: If you wish to use a namespace other than 'Suprema', you will need to go through and update the imported .cs files with the namespace of your choice.