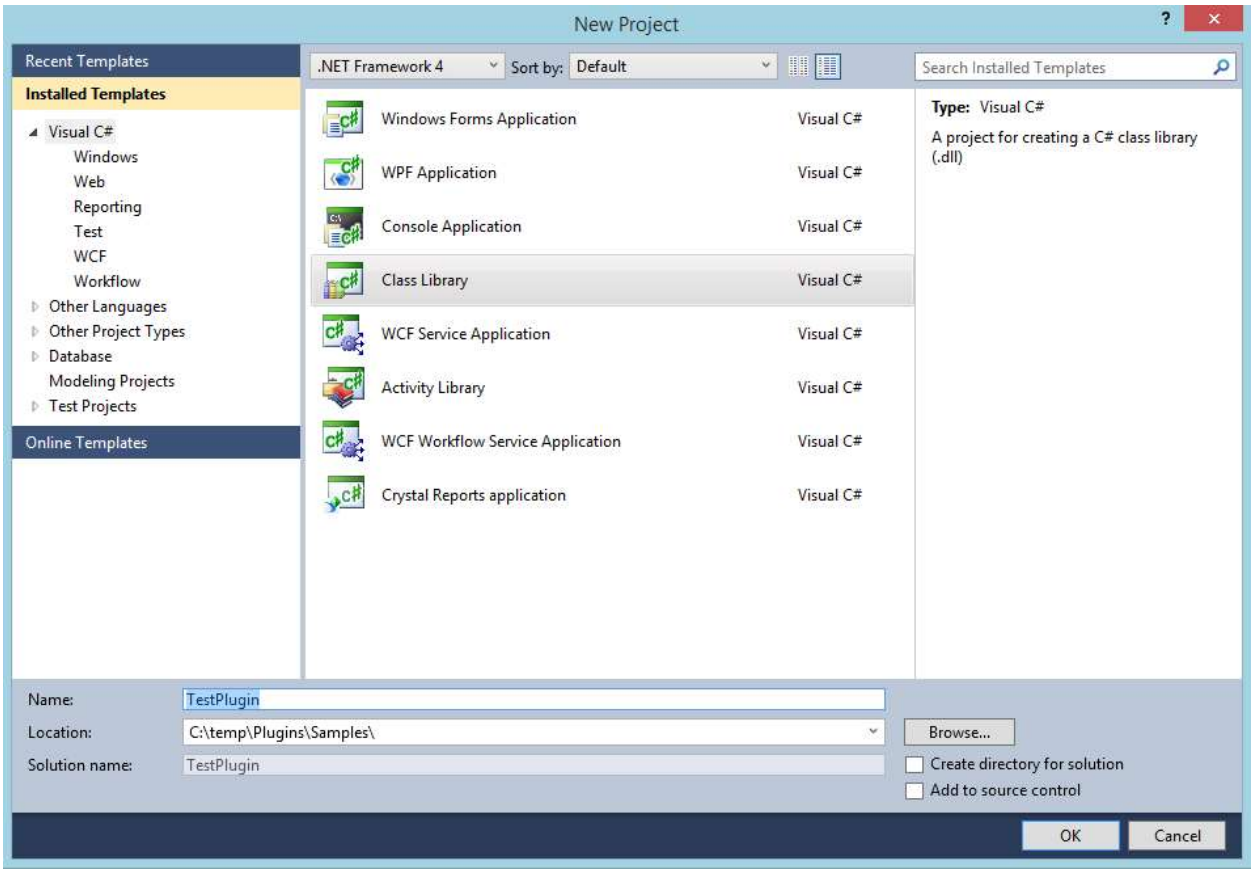
How to make DMC plugin

Visual Studio C# is needed to create plugin for DMC.

1. Install 32bit DMC version and use 32bit DMC version for creating plugins (windows form display error might occur in Visual Studio when using 64bit dll's).

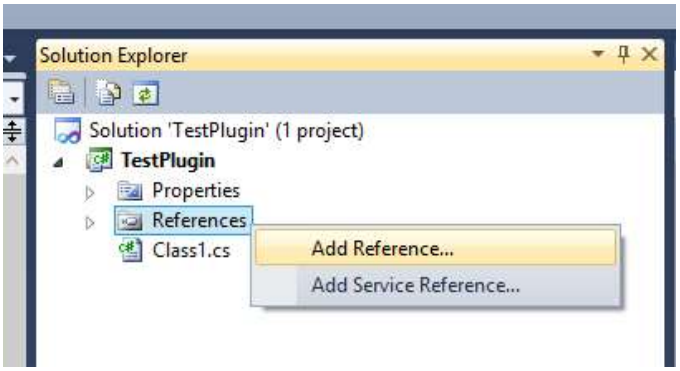
   Optional: Copy "Base.xml", "Core.xml", "GUI.xml" to installed version. These files contain descriptions of DMC functions. When description files are available, descriptions of the DMC functions are automatically imported by the Visual Studio.

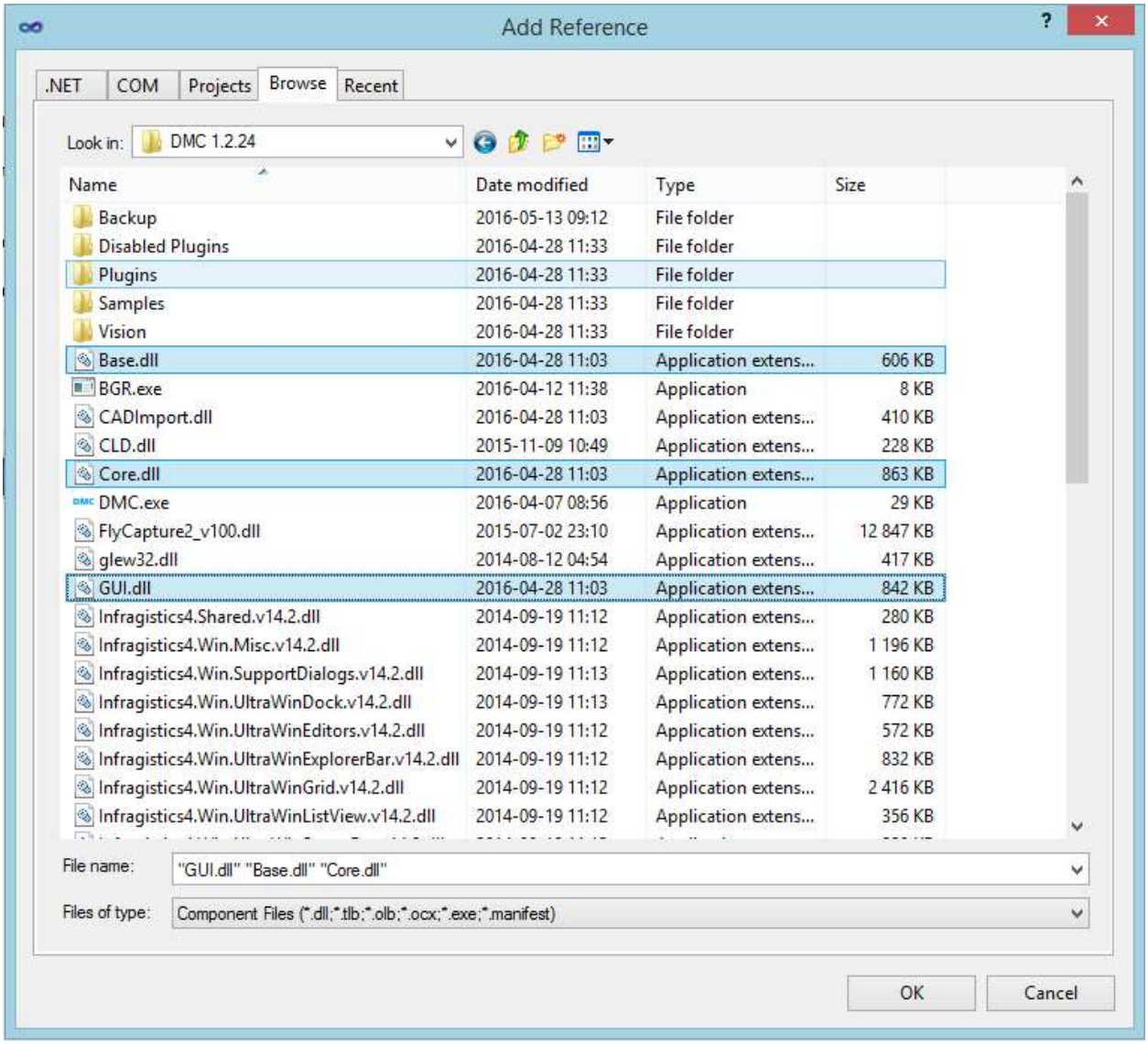
2. Run Visual Studio and create new Project.



- Select project type "Class Library"
- Define project name. **Project name must contain word "Plugin"**. Only dll files with name containing word "Plugin" will be loaded.
- Select location where to save project

3. Add references to created project.

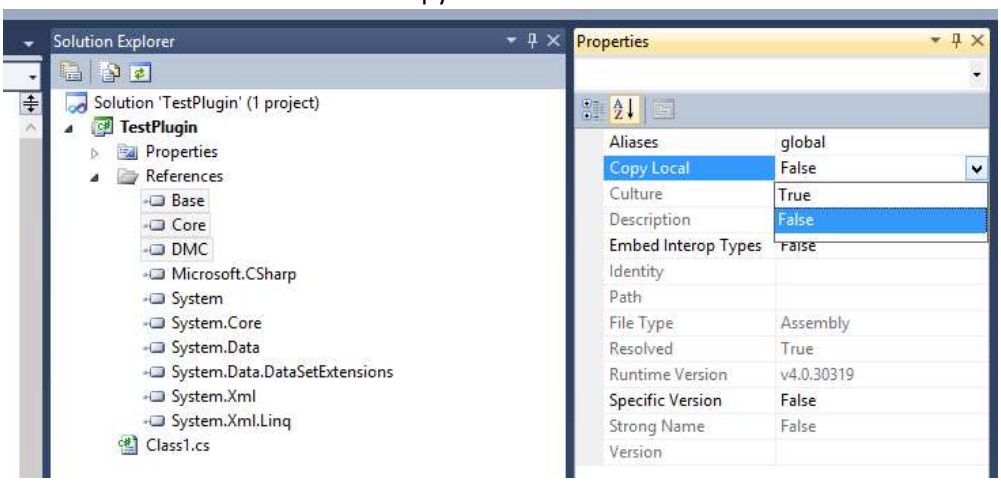




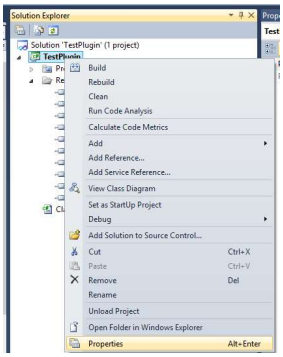
- **Base.dll**. Contains settings, interfaces (IAxis, , communicates with hardware, states.

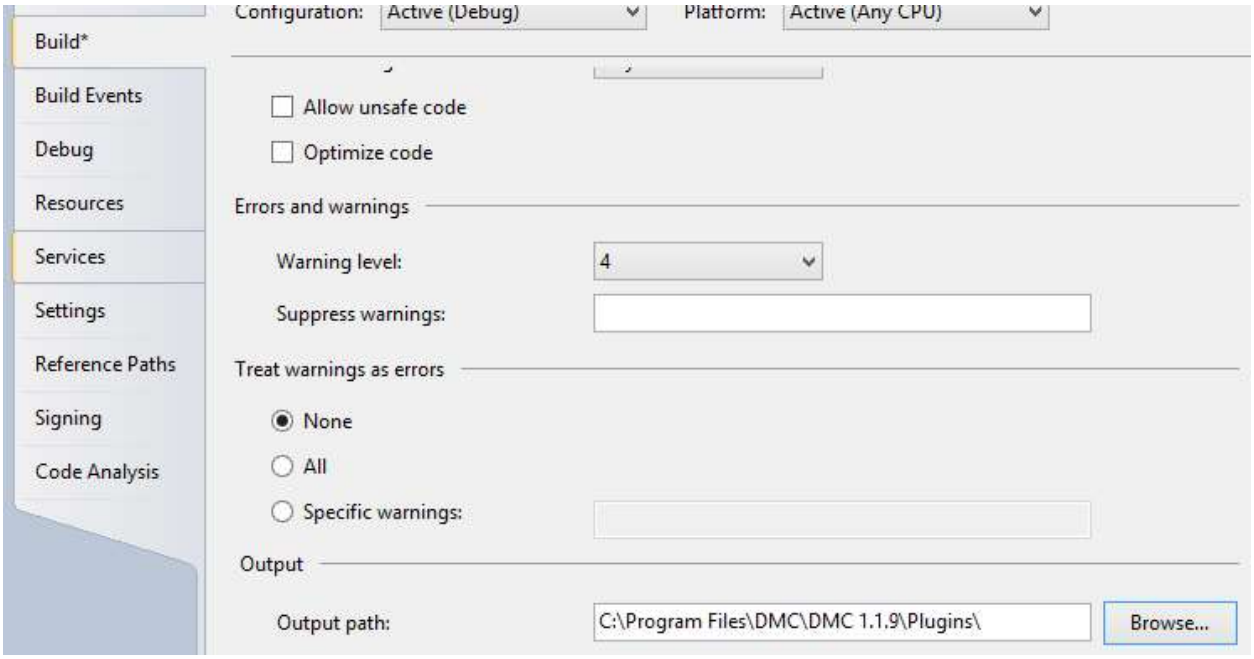
- **Core.dll**. Contains interfaces of commands, commands (Line, Arc, ...).
- **GUI.dll**. Contains top level functions that are used in main DMC window (Connect to hardware, Compile, Run, Stop, ...), also has methods for adding/hiding custom buttons.
  .

4. Select added references and set Copy Local to False.

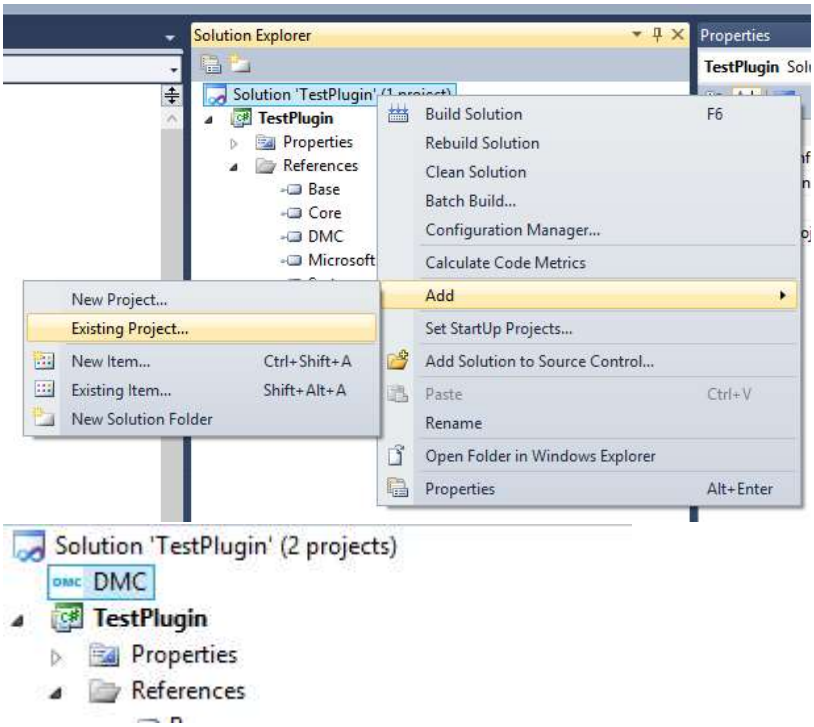


5. Go to Project settings and change project “Output path” to DMC path + Plugin directory. E.g C:\Program Files\DMC\DMC 1.1.9\Plugins\

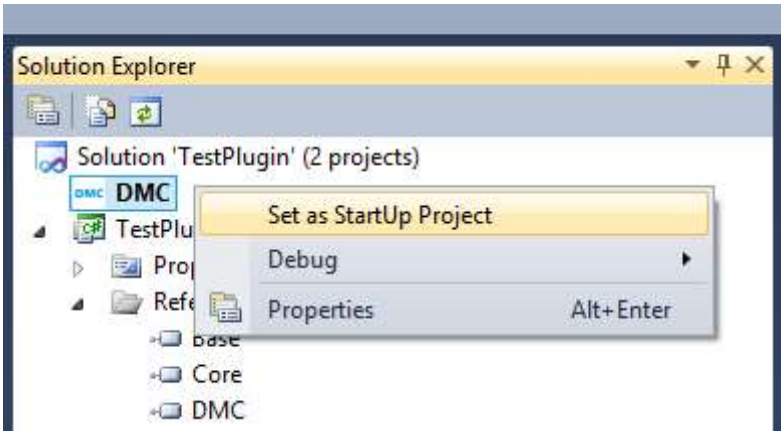



Build*
Build Events
Debug
Resources
Services
Settings
Reference Paths
Signing
Code Analysis
Configuration: Active (Debug)
Platform: Active (Any CPU)
Allow unsafe code
Optimize code
Errors and warnings
Warning level: 4
Suppress warnings:
Treat warnings as errors
None
All
Specific warnings:
Output
Output path: C:\Program Files\DMC\DMC 1.1.9\Plugins\
Browse...

6. Add DMC.exe to solution. Right click on solution, click Add->Existing project and browse for DMC.exe.



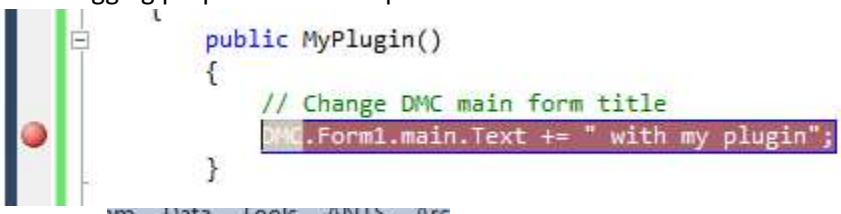
7. Set DMC as StartUp Project



8. Add Reference System.Windows.Forms and make code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Base;

namespace TestPlugin
{
    // Class name must be public, contain Plugin word, must inherit IDevice interface,
can't be abstract
    public class MyPlugin : IDevice
    {
        public MyPlugin()
        {
            // Change DMC main form title
            DMC.Form1.main.Text += " with my plugin";
        }

        // Action when user clicks Connect to hardware and IsEnabled is true
        public bool Connect() { return true; }

        // Action when user clicks Disconnect from hardware
        public void Disconnect() { }

        // Action when user clicks Stop button
        public void Stop() { }

        public string GetName() { return "My Plugin"; }

        // Action when changes to settings are confirmed or loaded during DMC startup
        public bool ApplySettings() { return true; }

        // Needs to return if device is connected
        public bool IsConnected() { return false; }

        // Is device is enabled
        public bool IsEnabled() { return false; }

        // Called before starting recipe
        public bool OnRecipeStart() { return true; }

        // Called after recipe is finished or stopped
        public void OnRecipeFinish() { }

        // Get device settings
        public IDeviceSettings GetSettings() { return null; }

        // Get error message ( is called if Connect returns false )
        public string GetErrorMessage() { return Base.Functions.GetLastErrorMessage();
}
    }
}
```

9. For debugging purpose add breakpoint.



10. Click Run.