UCCNC Macro Templates

Introduction

These Visual Studio project templates can be used to create macro or macroloop files for the UCCNC machine control application. This allows users to create macros within a compiler environment that mimics the UCCNC run-time environment. In addition, a Visual Studio extension is provided which is used to import and export text files to be executed by the UCCNC application.

NOTE: These templates are not part of the UCCNC application. They are provided for informational purposes only, without cost or license fees, by a third party that is not affiliated with the author(s) of UCCNC.

Prerequisites

These project templates are targeted to all editions of the Visual Studio 2019 platform. The Visual Studio 2019 IDE (Integrated Development Environment) can be downloaded from Microsoft using the following link:

https://my.visualstudio.com/Downloads?q=visual%20studio%202019&wt.mc_id=o~msft~vscom~older-downloads

It is recommended that the most recent VS 2019 version be installed. For non-commercial users, the Community Edition is available without any license costs. Please refer to Microsoft's licensing documents for further information.

NOTE: These template and extension files are not compatible with Visual Studio 2022.

Installation Files

The installation files are available on Google Drive, using this link:

https://drive.google.com/drive/folders/1FzjDcvGU_I5E2FebyoIbafR1-ILOgAdS?usp=share_link

- UCCNC CSharp Macro Template.zip
- UCCNC CSharp Macroloop Template.zip
- UCCNC VBasic Macro Template.zip
- UCCNC VBasic Macroloop Template.zip

  These are the Visual Studio project template files, and should be downloaded to the ProjectTemplates folder:

  %userprofile%\Documents\Visual Studio 2019\Templates\ProjectTemplates\

NOTE: Do not extract the contents of these ZIP files. Visual Basic requires the ZIP files in their compressed form.

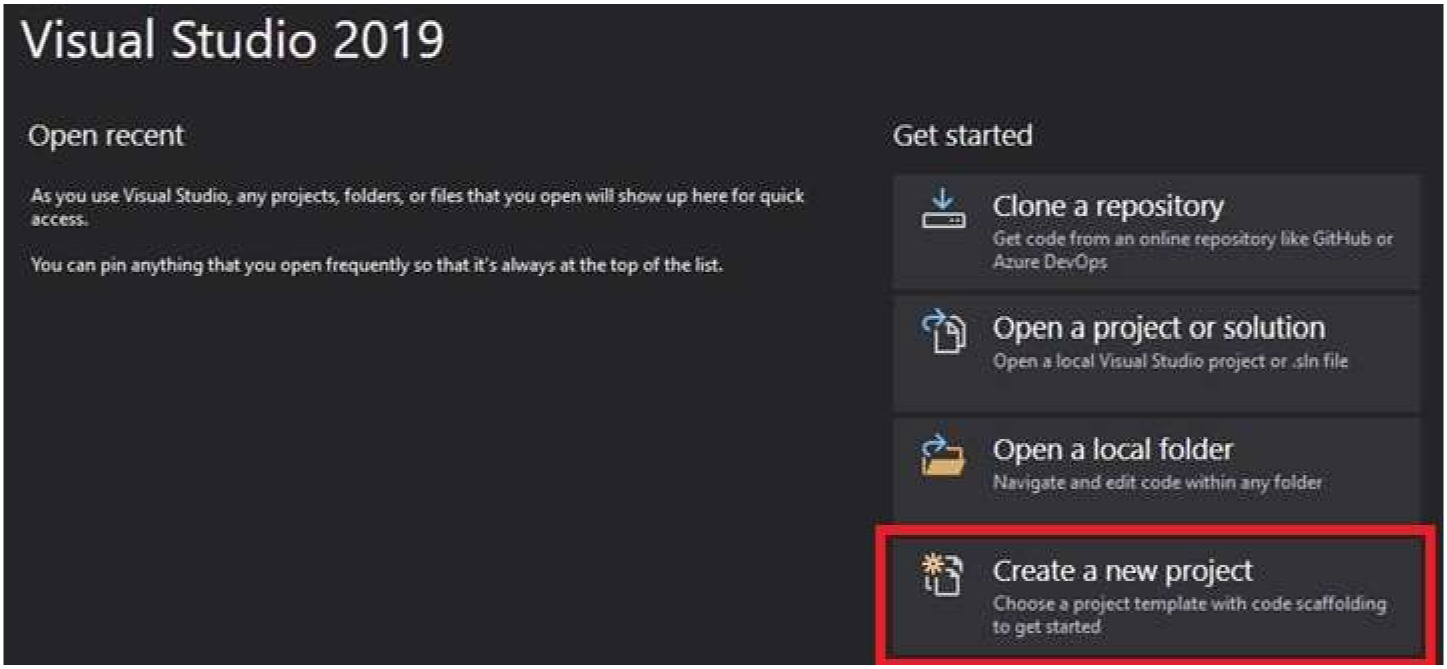
- UCCNC MacroExtension.vsix

  This is the Visual Studio extension package which implements the import/export tools. Download this file to any folder, then double-click to open and install it into your Visual Studio IDE.

NOTE: If an earlier version of the macro extension was installed, it must first be uninstalled using the Manage Extensions utility (accessible from the Extensions menu in the Visual Studio IDE).

Using The Templates

The templates are available when creating a new Visual Studio solution. After launching Visual Studio, the following dialog screen is presented:



Visual Studio 2019

Open recent

As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

You can pin anything that you open frequently so that it's always at the top of the list.

Get started

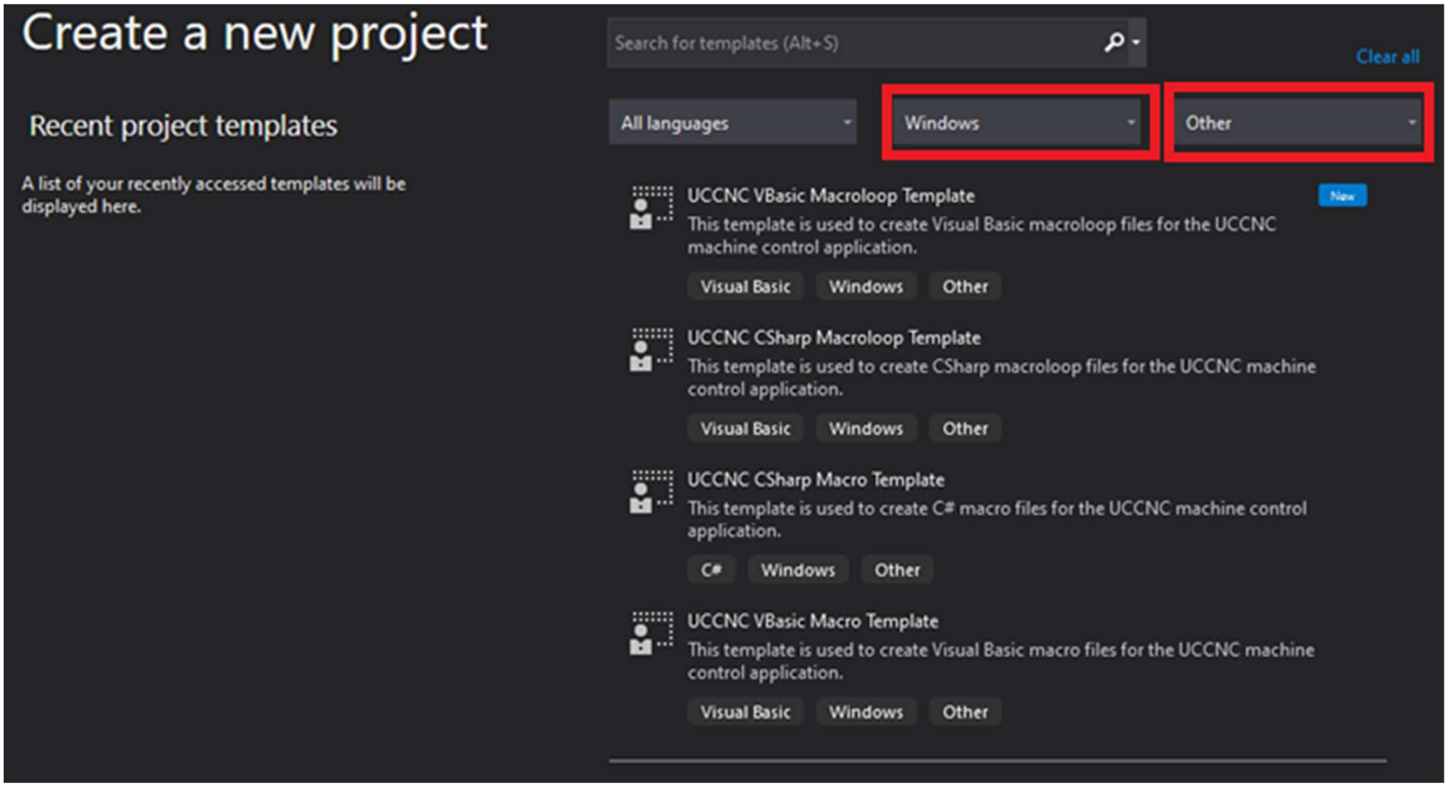
Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Click on "Create a new project". This will open a list of all available project templates. Using the two drop-down lists at the upper right, select the Windows platform and Other project types:



Create a new project

Search for templates (Alt+S)

Clear all

Recent project templates

A list of your recently accessed templates will be displayed here.

All languages | Windows | Other

UCCNC VBasic Macroloop Template
New
This template is used to create Visual Basic macroloop files for the UCCNC machine control application.
Visual Basic | Windows | Other

UCCNC CSharp Macroloop Template
This template is used to create CSharp macroloop files for the UCCNC machine control application.
Visual Basic | Windows | Other

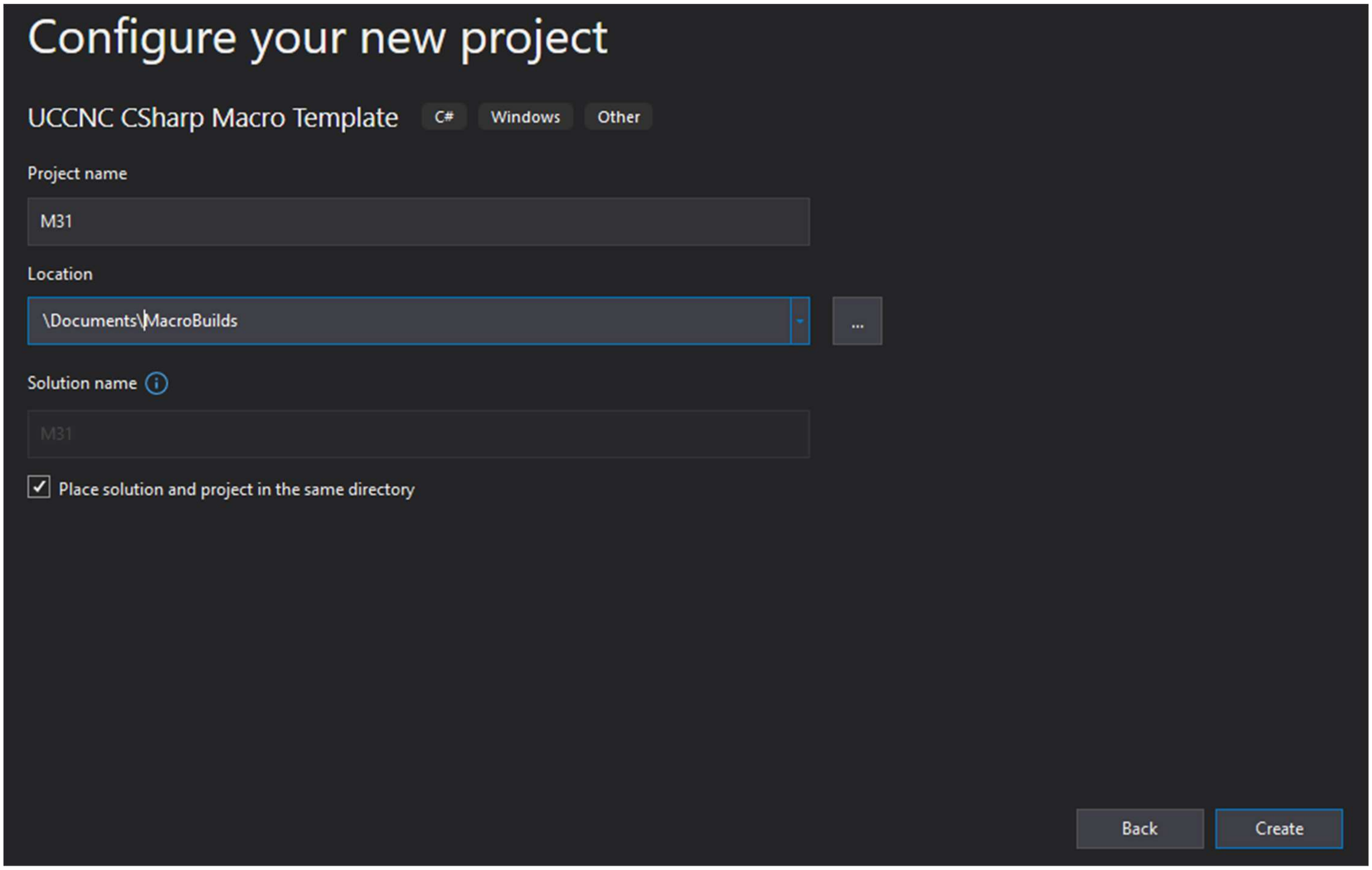
UCCNC CSharp Macro Template
This template is used to create C# macro files for the UCCNC machine control application.
C# | Windows | Other

UCCNC VBasic Macro Template
This template is used to create Visual Basic macro files for the UCCNC machine control application.
Visual Basic | Windows | Other

From this list of project templates, select the template appropriate for the programming language in which you will be developing the macro or macroloop.

NOTE: While many macro and macroloop files are interchangeable, the run-time environments in the UCCNC application for each of these are not identical. Therefore, you should select the correct project template based on whether you will be developing a macro or macroloop. This will provide you with the correct compile-time environment.

After selecting the project template, configure the project name and source code folder, then click “Create”:



Configure your new project

UCCNC CSharp Macro Template

Project name: M31

Location: \Documents\MacroBuilds

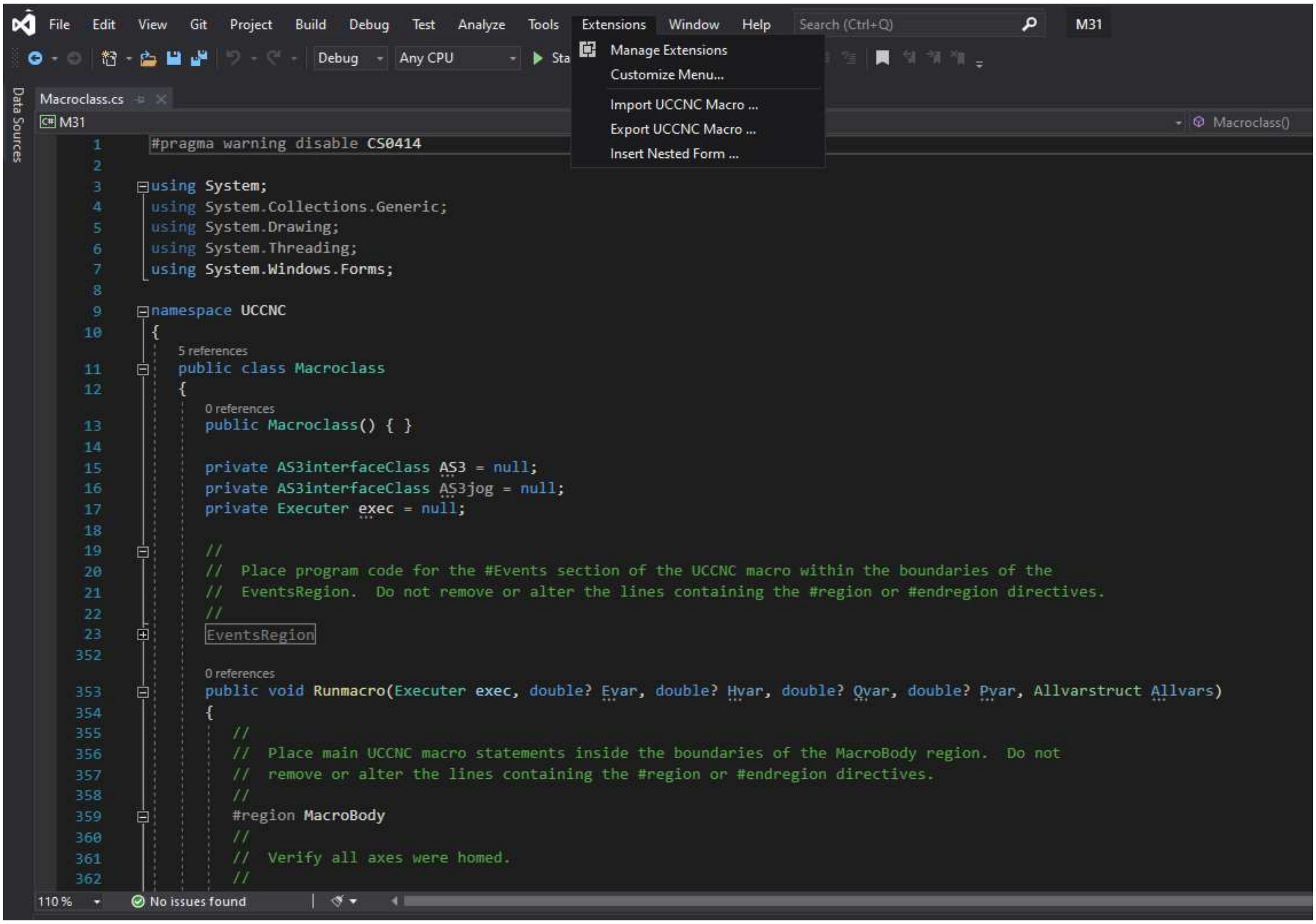
Solution name: M31

☑ Place solution and project in the same directory

NOTE: The default project name is “UCCNC Macro1”. You can use a single project to develop multiple macros, and import/export the text files as necessary. However, you should also consider creating a separate project for each macro or macroloop, and name the project to match the macro name. For example, M31, as shown above.

After Visual Studio has created your new project, the main IDE window will be displayed. The Macroclass (or Macroloopclass) file will be opened. You can begin writing your source code into the template, within the MacroBody region, or you can import an existing UCCNC macro text file.

The import and export tools are available in the Extensions drop-down menu. However, these tools will only be visible when the Macroclass or Macroloopclass source code file is selected for editing.



```
#pragma warning disable CS0414

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace UCCNC
{
    5 references
    public class Macroclass
    {
        0 references
        public Macroclass() { }

        private AS3interfaceClass AS3 = null;
        private AS3interfaceClass AS3jog = null;
        private Executer exec = null;

        //
        //  Place program code for the #Events section of the UCCNC macro within the boundaries of the
        //  EventsRegion.  Do not remove or alter the lines containing the #region or #endregion directives.
        //
        EventsRegion

        0 references
        public void Runmacro(Executer exec, double? Evar, double? Hvar, double? Qvar, double? Pvar, Allvarstruct Allvars)
        {
            //
            //  Place main UCCNC macro statements inside the boundaries of the MacroBody region.  Do not
            //  remove or alter the lines containing the #region or #endregion directives.
            //
            #region MacroBody
            //
            //  Verify all axes were homed.
            //
```

Using Windows Forms

You can develop UCCNC macros which include one or more Windows Form classes. You create these using the standard Visual Studio tools (Forms Designer), and instantiate the class in your macro the same way it would be done in a Windows application.

In order to export the Form classes, you must use the “Insert Nested Form…” command from the Extensions menu. This will generate an appropriate region tag within the macro source code. When exporting the macro to a text file, the export tool will recognize the region tag and automatically insert the Form classes into the generated file.

Similarly, when importing a macro text file which contains a previously exported Form class (or classes), the import tool will update the Form source files with the imported code. If the Form class in the imported file does not exist in the Visual Studio project, the source files will be created and attached to the project.

Usage Notes

- The project template provides several class files that define the run-time environment. This is not a complete set, and there are portions of the run-time environment which are not defined in the project. This was done due to the complexity of some classes within the UCCNC application.

- Within the EnumRegion of the source code files are enumerated constants for the LED, Button, and Field values used in UCCNC. These can be substituted in place of the integer value that is passed to a UCCNC routine, to improve readability. The entire EnumRegion is automatically exported to the text file if any of the enumerated constants are used in the source code file.

The LED values are prefixed by LED; button values are prefixed by BTN; and field values are prefixed by FLD. In C Sharp, the enumerated value must be cast to int.

For example:

```
// Using the LED number.
if (!exec.GetLED(58))
   Console.WriteLine("Z Axis has not been homed!");

// Using the LED enumerated value.
if (!exec.GetLED((int)LED.Zhomed))
   Console.WriteLine("Z Axis has not been homed!");


// Using the Button number.
while (!AS3.Getbutton(128))
   ;

// Using the Button enumerated value.
while (!AS3.Getbutton((int)BTN.Cyclestart))
   ;

// Using the field number.
string xPosition = AS3.Getfield(2726);

// Using the Field enumerated value.
string xPosition = AS3.Getfield((int)FLD.FixedProbePosX);
```

- There is no need to pre-edit the text files prior to importing them into Visual Studio. The import tool will identify the #VB and #Events directives, and the EnumRegion (if previously exported).

- You cannot import a text file which contains the #VB directive into a C Sharp source code project. Likewise, text files that are imported into a Visual Basic source code project must contain a #VB directive.

- These project templates provide an environment only for *compiling* macros and macroloops. The DLL file(s) that are generated by the compiler are not useful and cannot be attached to the UCCNC application. There is no capability to debug your macros or macroloops.