Generating a trivial STEP-NC program file - C++.

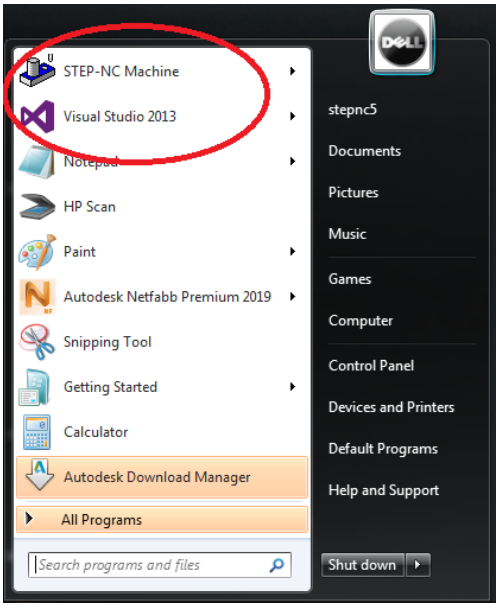
Contributor: Efrain Rodriguez

STEP-NC LaDPRER

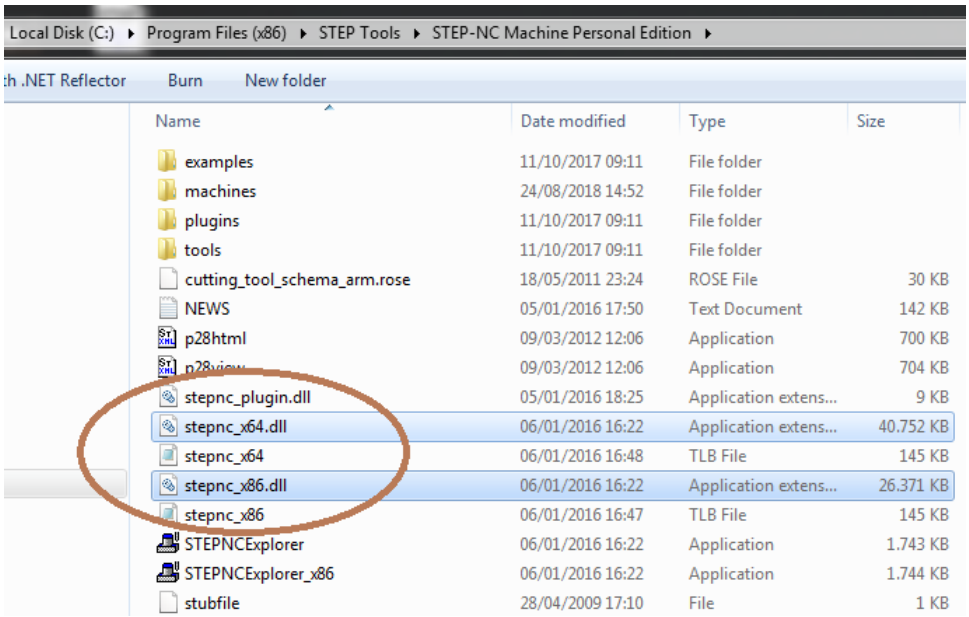
2018

This is a tutorial version of the “stepnc-hello” demo shared by STEP Tools, Inc. for building a trivial STEP-NC program file. This consists of a small code project written in C++ that creates a machining program as STEP-NC.

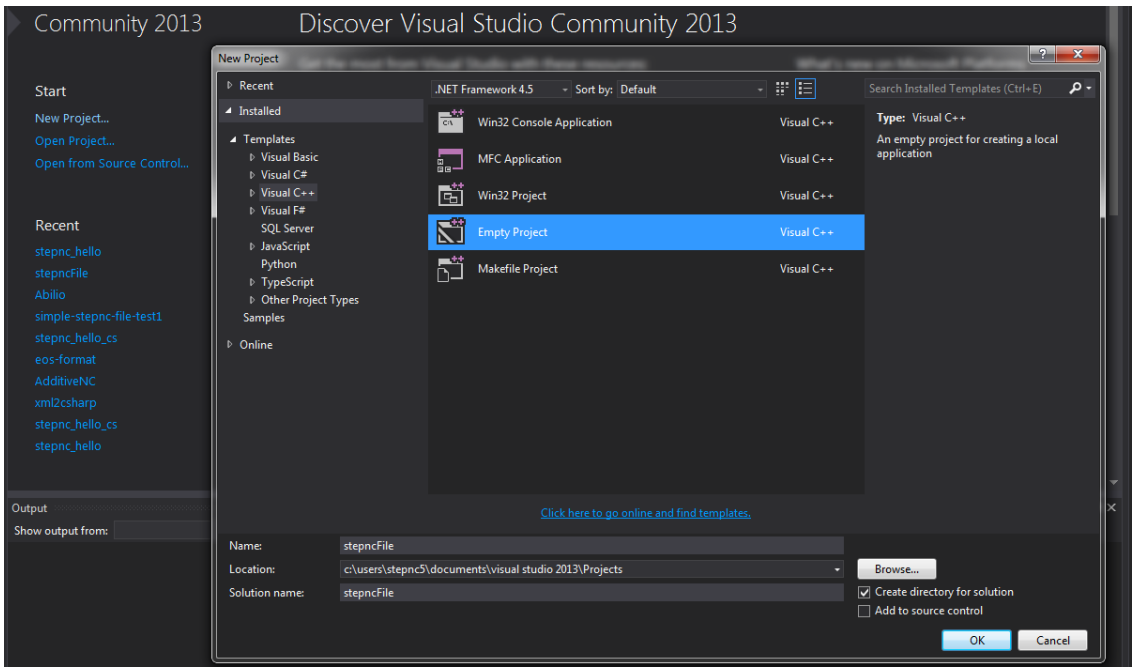
You need **Visual Studio 2013** installed on your **Windows** operating system. You also need the **STEP-NC Machine** software containing the **stepnc.dll** to create the STEP-NC data.



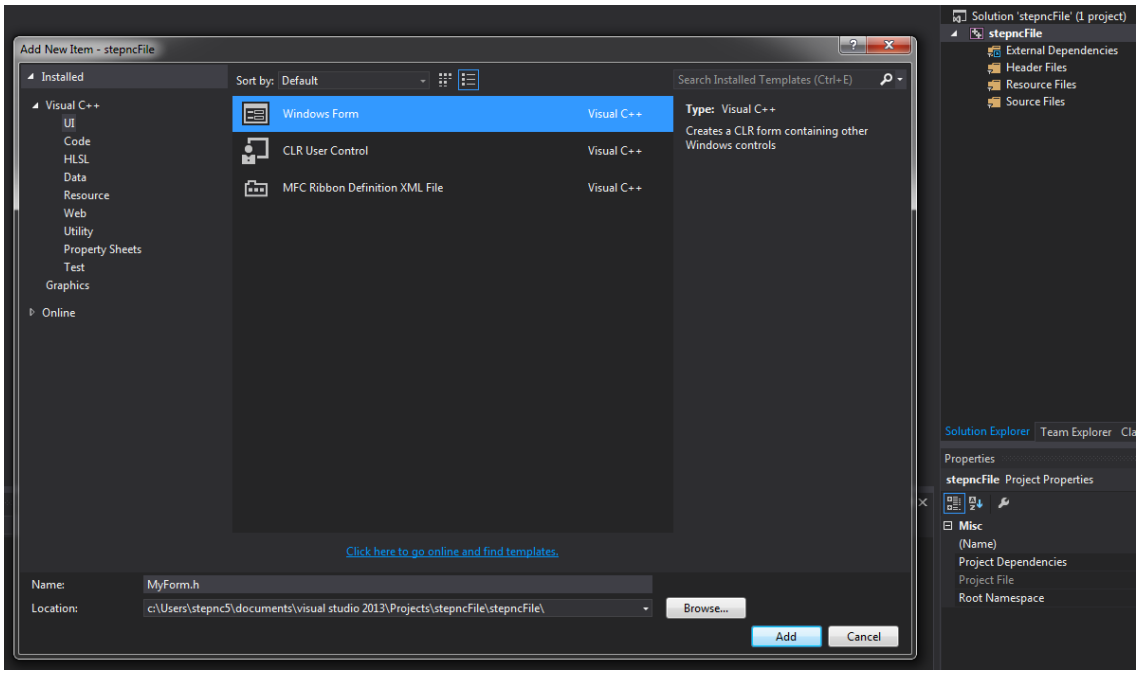
Make sure that the **stepnc_x64.dll** and/or **stepnc_x86.dll** are installed in the STEP-NC Machine directory, usually: **C:\Program Files (x86)\STEP Tools\STEP-NC Machine Personal Edition.**



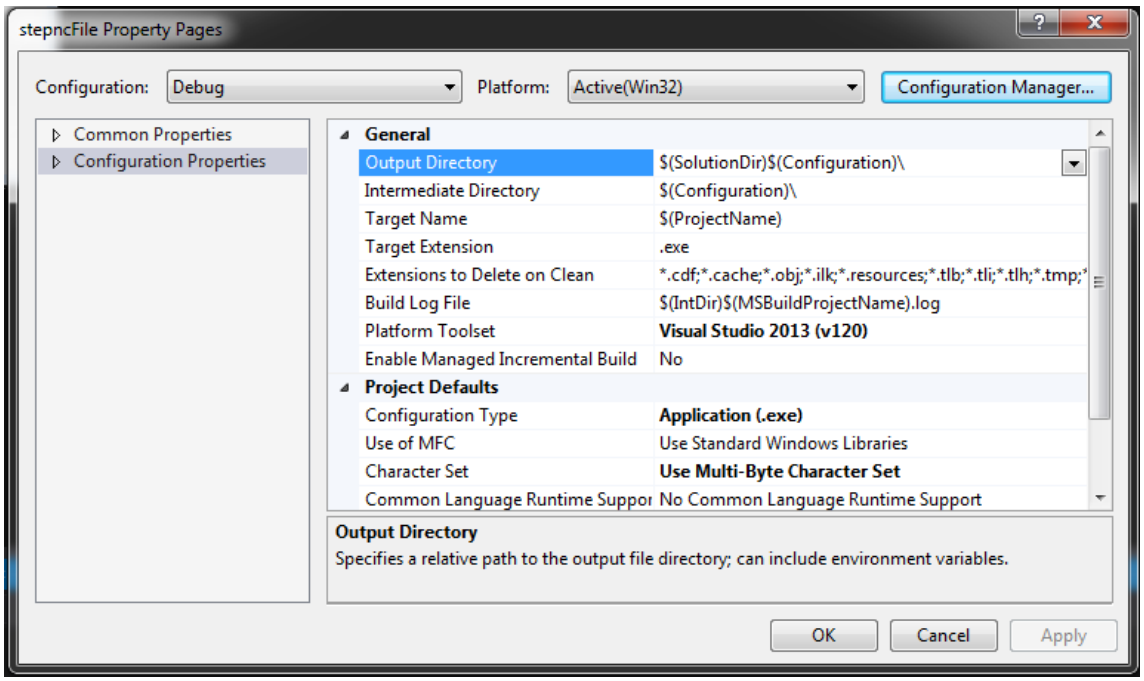
You will need to create a new C++ **Empty Project** by starting Visual Studio. Give your project a name; for example, “stepncFile”.



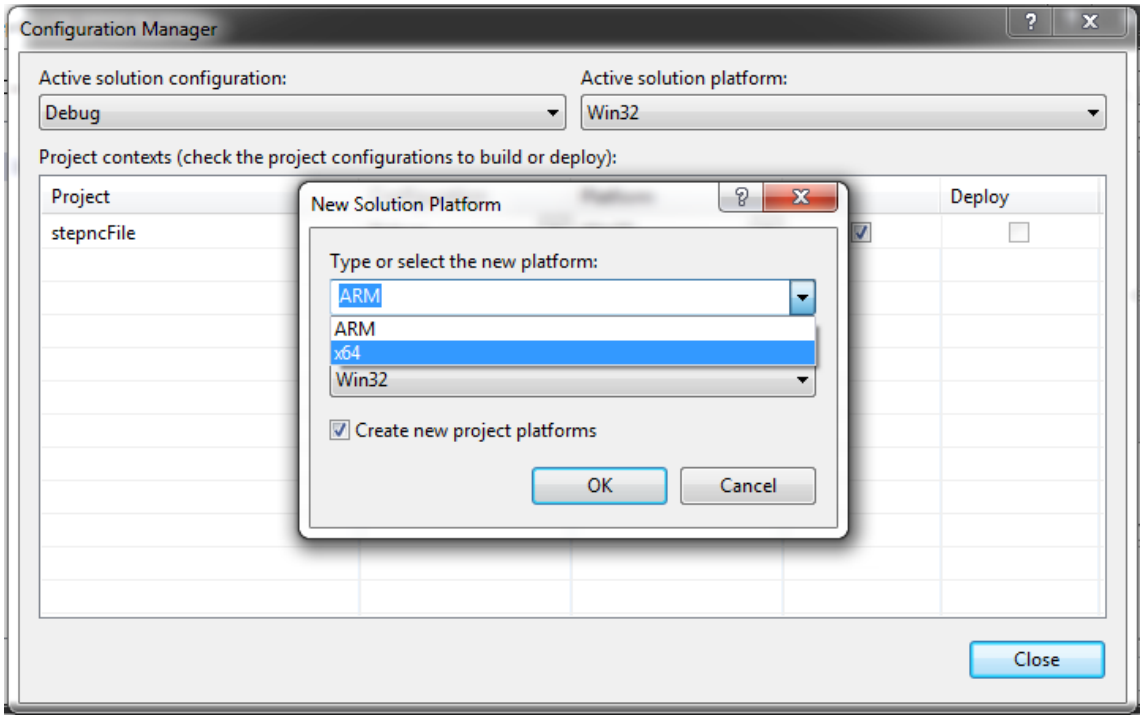
Then add a new **Windows Form** item and give it the name you want; for example, “MyForm”.



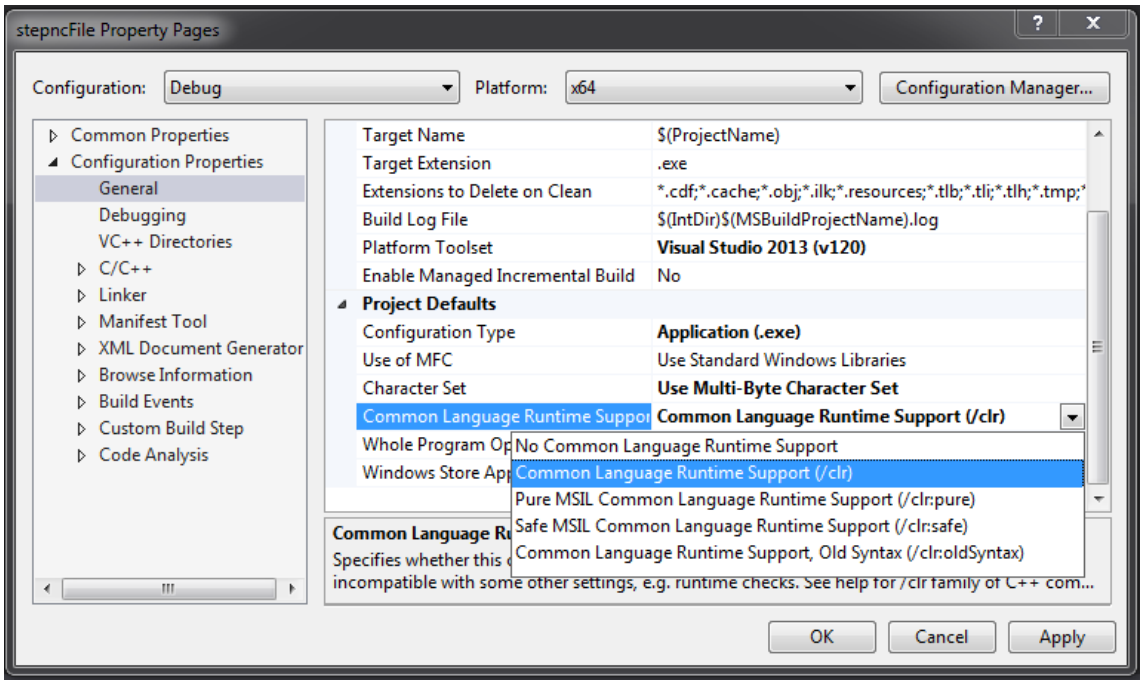
By right clicking on the project, select **Properties** from the menu that will appear to open the project **Property Pages**. Then click on **Configuration Manager**.



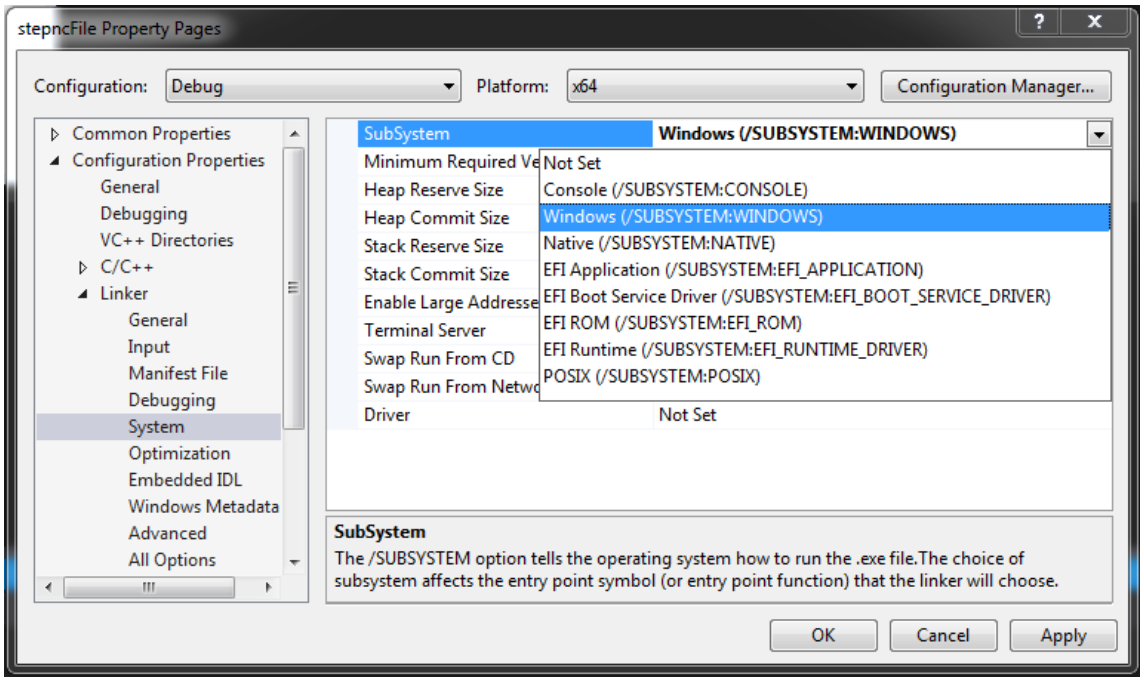
Select **New** from the **Active solution platform** menu and choose the **x64** platform. Press on **OK** and close the **Configuration Manager** box.



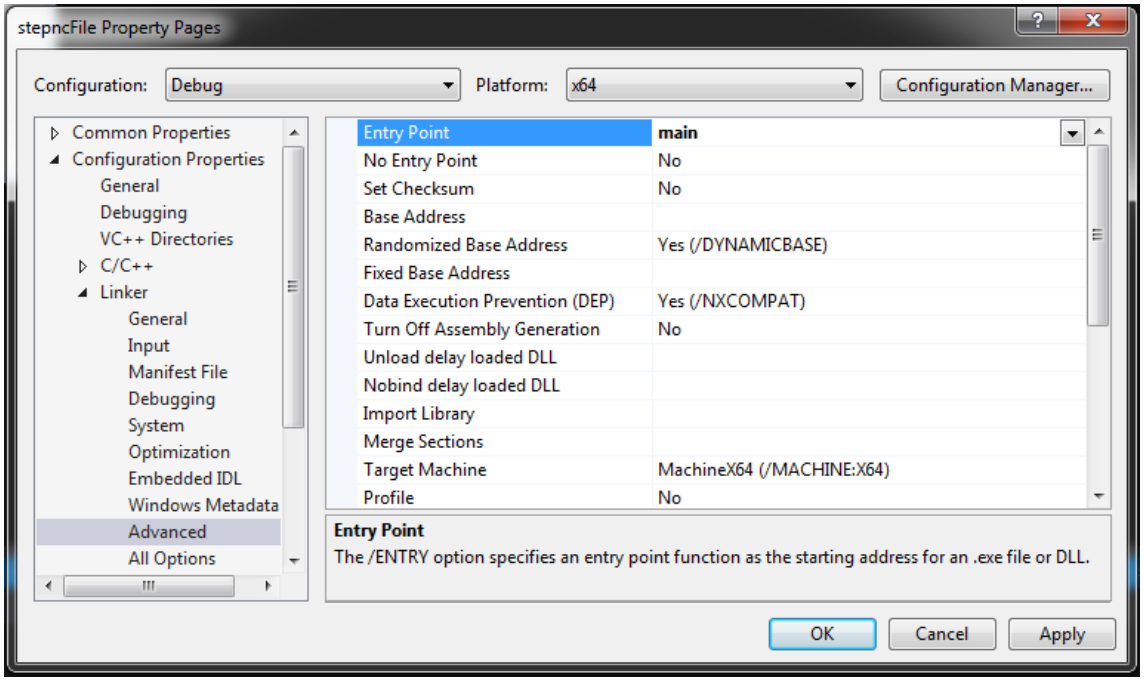
Continuing in the **Property Pages**, in the **Configuration Properties** drop-down list, go to **General** and select **Common Language Runtime Support (/clr)** from the menu shown in the figure.



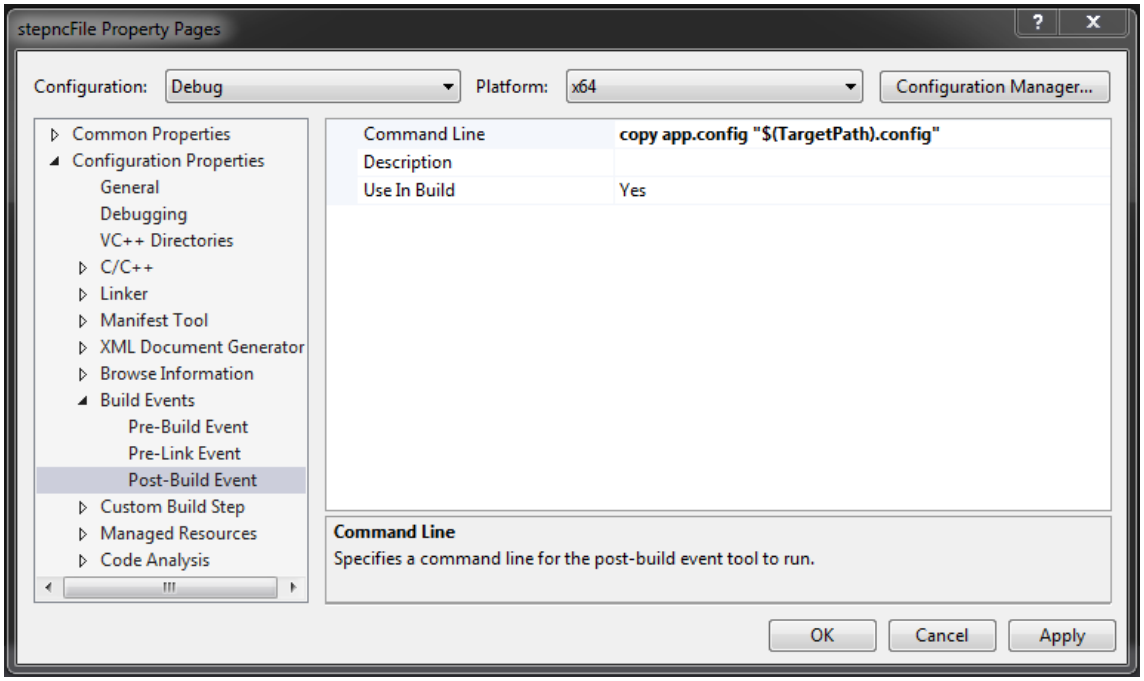
In the path **Linker->System->SubSystem** select **Windows (/SUBSYSTEM:WINDOWS)** for execution environment of Windows (GUI) application.



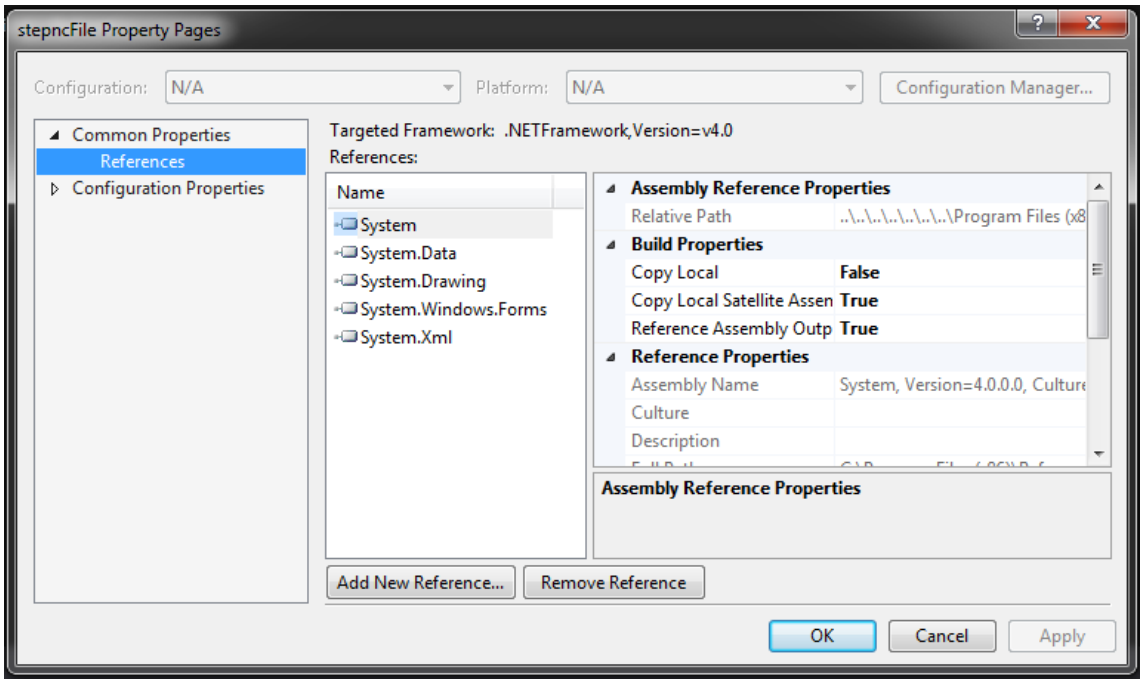
Define the starting address of the project as “main” function in **Linker->Advanced->Entry Point**.



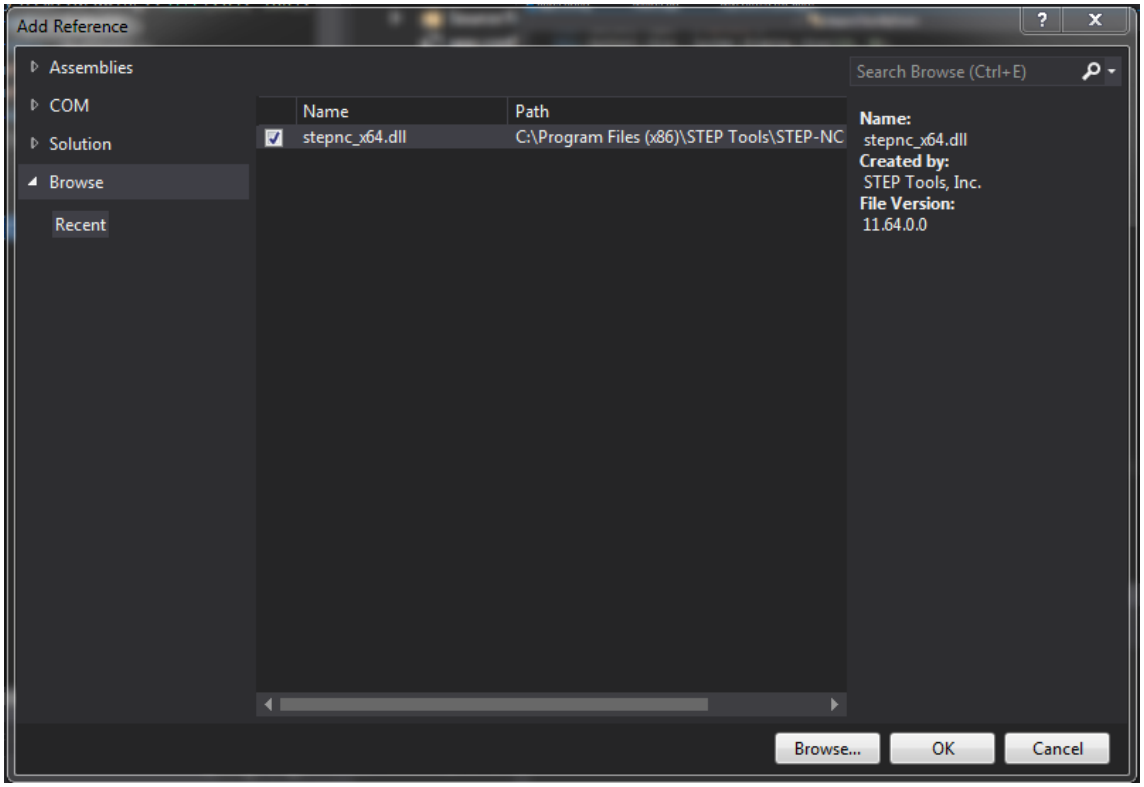
Specify a post-build events for the **Build Events** page by typing the sentence **copy app.config “($TargetPath).config”** in the **Command Line** edit box.



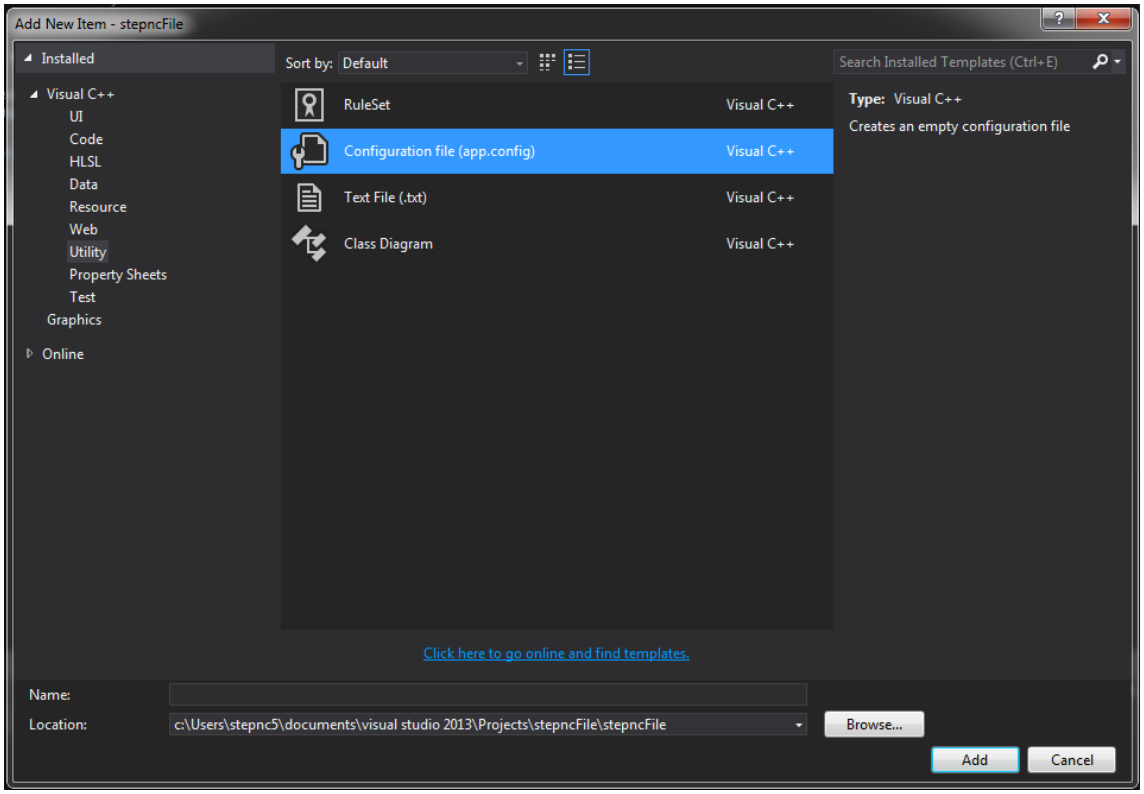
An important step is to add the **stepnc.dll** as within the reference from the project to use its functions in the building the STEP-NC data. You can add that stepnc.dll accessing by **Common Properties->References** and clicking on **Add New Reference.**



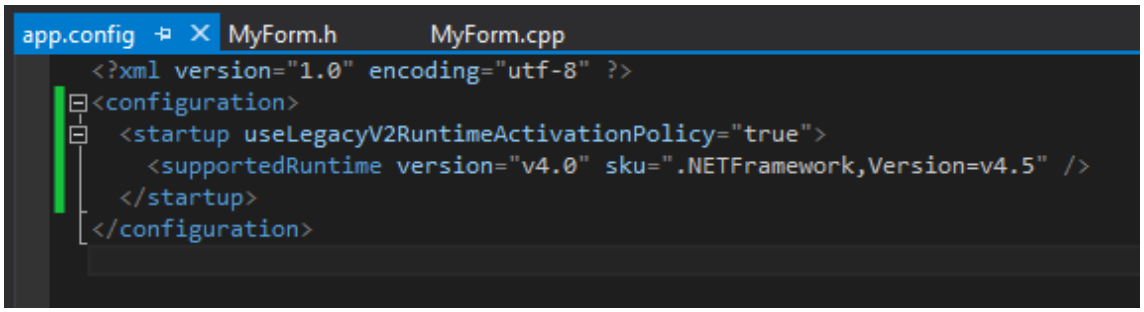
The **Browser** will allow you to access the **stepnc.dll** through from the aforementioned STEP-NC Machine directory.



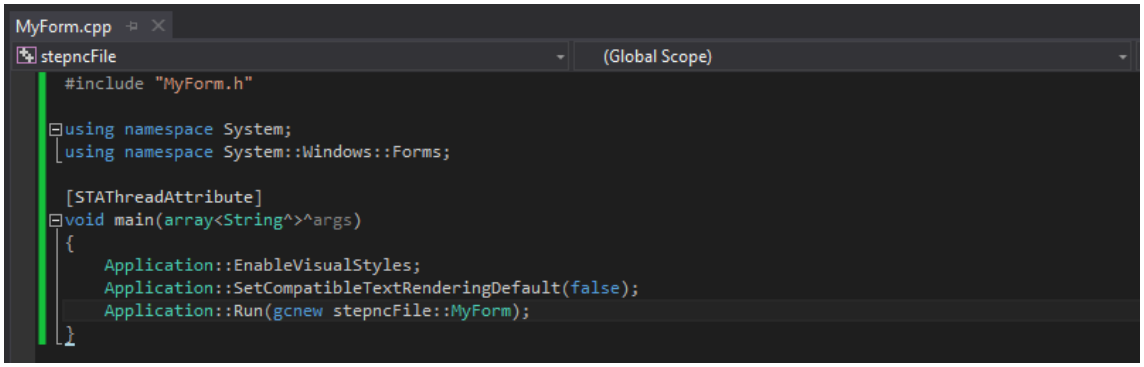
You will also need to create an **.xml configuration file** for the project. You can do it by adding a new item from C++ **Utility** menu, well as shown in the figure.



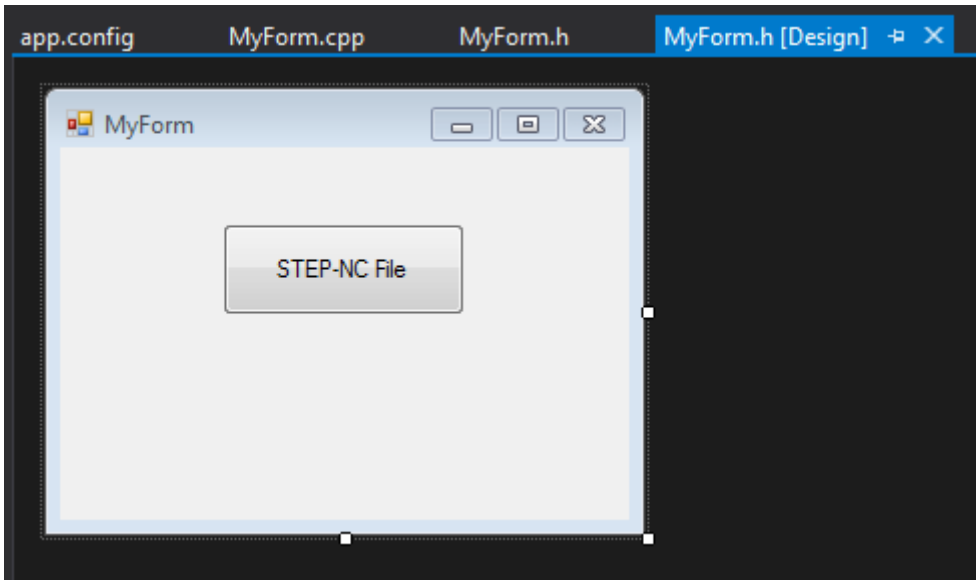
In the created **app.config** file type the following necessary statements:



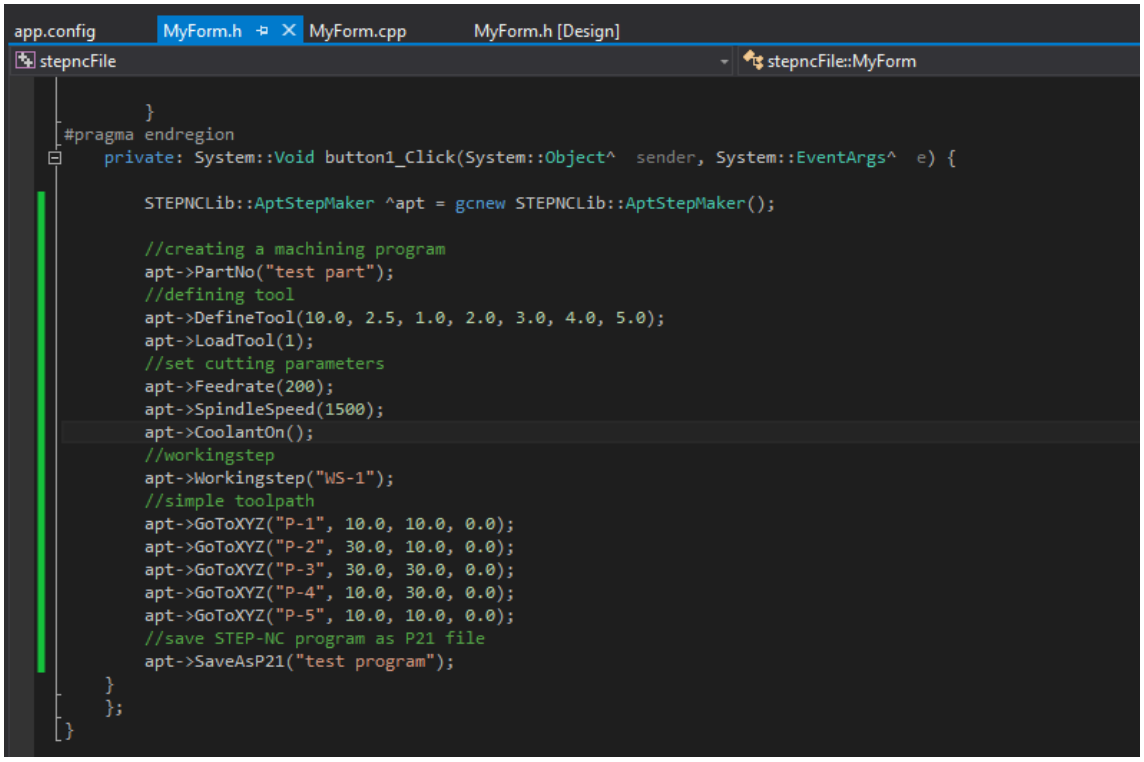
In the **MyForm.cpp** file of the project place the following code necessary:



Add a button object into main form as seen in the figure.



In the **MyForm.h** file, just inside the definition of the button object, you will begin to develop your STEP-NC machining part program. Below is a code template to generate a trivial STEP-NC machining program file using methods of stepnc.dll that are based on the APT CL file standard.



Firstly, you define an object *apt* belonging to *AptStepMaker* class included in the *STEPNCLib* library of stepnc.dll.

Subsequently, create a STEP-NC program session through the *PartNo* method with the program name as input parameter.

DefineTool will allow you to define a tool, where the first input parameter refers to tool diameter. That tool can be load by using *LoadTool* specifying its number.

You can set cutting parameters such as feedrate, spindle speed and coolan activation.

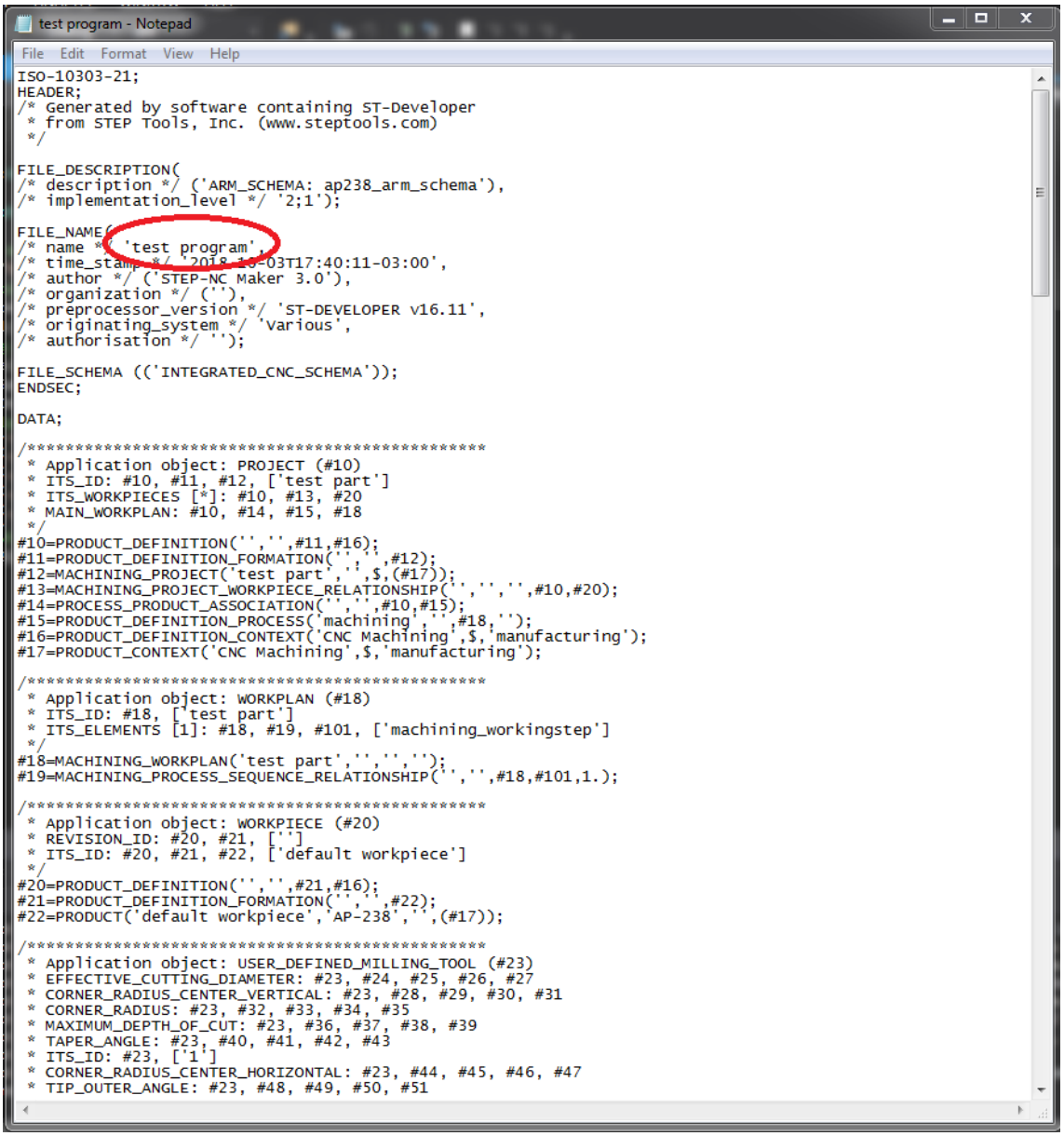
Workingstep method will allow you create a workingstep within the machining program passing its label as input parameter.

With *GoToXYZ* you can create toolpath specifying the XYZ Cartesian point.

Finally, the STEP-NC data file in P21 format can be generated through the *SaveAsP21* method with the file name as input parameter.

The file is saved in your project directory. A piece of the STEP-NC program generated is shown in the figures.

Nice!



```
#77=ACTION_PROPERTY_REPRESENTATION('','milling',#76,#78);
#78=REPRESENTATION('constant',(#79),#57);
#79=DESCRIPTIVE_REPRESENTATION_ITEM('constant','coolant on');

/*************************************************
 * Application object: MACHINING_WORKINGSTEP (#80)
 * ITS_ID: #80, ['WS-1 WS 1']
 * ITS_OPERATION: #80, #81, #124
 * ITS_FEATURE: #80, #82, #83, #84, #85, #107
 */
#80=MACHINING_WORKINGSTEP('WS-1 WS 1','machining','','');
#81=MACHINING_OPERATION_RELATIONSHIP('','machining',#80,#124);
#82=MACHINING_FEATURE_RELATIONSHIP('','machining',#80,#83);
#83=MACHINING_FEATURE_PROCESS('','machining','','');
#84=PROPERTY_PROCESS('machining','machining',#83,'');
#85=PROCESS_PROPERTY_ASSOCIATION('','machining',#84,#107);

/*************************************************
 * Application object: MILLING_TECHNOLOGY (#86)
 * SPINDLE: #86, #87, #88, #89, #90
 * FEEDRATE: #86, #91, #92, #93, #94
 */
#86=MACHINING_TECHNOLOGY('','milling','','');
#87=ACTION_PROPERTY('spindle','milling',#86);
#88=ACTION_PROPERTY_REPRESENTATION('rotational speed','milling',#87,#89);
#89=MACHINING_SPINDLE_SPEED_REPRESENTATION('spindle speed',(#90),#57);
#90=MEASURE_REPRESENTATION_ITEM('rotational speed',NUMERIC_MEASURE(1500.),
#95);
#91=ACTION_PROPERTY('feedrate','milling',#86);
#92=ACTION_PROPERTY_REPRESENTATION('feed speed','milling',#91,#93);
#93=MACHINING_FEED_SPEED_REPRESENTATION('feed speed',(#94),#57);
#94=MEASURE_REPRESENTATION_ITEM('feed speed',NUMERIC_MEASURE(200.),#104);
#95=DERIVED_UNIT((#96,#99));
#96=DERIVED_UNIT_ELEMENT(#97,1.);
#97=CONTEXT_DEPENDENT_UNIT(#98,'revolution');
#98=DIMENSIONAL_EXPONENTS(0.,0.,0.,0.,0.,0.,0.);
#99=DERIVED_UNIT_ELEMENT(#100,-1.);
#100=(
CONVERSION_BASED_UNIT('minute',#102)
NAMED_UNIT(#101)
TIME_UNIT()
);
```

```
/*************************************************
 * Application object: FREEFORM_OPERATION (#124)
 * ITS_TECHNOLOGY: #124, #125, #86
 * ITS_TOOLPATH [1]: #124, #126, #128
 * ITS_ID: #124, ['P-1']
 * ITS_TOOL: #124, #23
 * ITS_MACHINE_FUNCTIONS: #124, #127, #115
 */
#124=FREEFORM_MILLING_OPERATION('P-1','','','');
#125=MACHINING_TECHNOLOGY_RELATIONSHIP('','',#124,#86);
#126=MACHINING_TOOLPATH_SEQUENCE_RELATIONSHIP('','',#124,#128,1.);
#127=MACHINING_FUNCTIONS_RELATIONSHIP('','',#124,#115);

/*************************************************
 * Application object: CUTTER_LOCATION_TRAJECTORY (#128)
 * BASICCURVE: #128, #129, #130, #131, #132
 * ITS_ID: #128, ['P-1 WS 1 TP 1']
 * ITS_TECHNOLOGY: #128, #133, #86
 */
#128=MACHINING_TOOLPATH('P-1 WS 1 TP 1','cutter location trajectory','',
'');
#129=ACTION_PROPERTY('basic curve','cutter location trajectory',#128);
#130=ACTION_PROPERTY_REPRESENTATION('','cutter location trajectory',#129,
#131);
#131=REPRESENTATION('',(#132),#57);
#132=POLYLINE('',(#134,#135,#136,#137,#138));
#133=MACHINING_TECHNOLOGY_RELATIONSHIP('','cutter location trajectory',
#128,#86);
#134=CARTESIAN_POINT('',(10.,10.,0.));
#135=CARTESIAN_POINT('',(30.,10.,0.));
#136=CARTESIAN_POINT('',(30.,30.,0.));
#137=CARTESIAN_POINT('',(10.,30.,0.));
#138=CARTESIAN_POINT('',(10.,10.,0.));

/*************************************************
 * END OF APPLICATION OBJECT DESCRIPTIONS
 */
#139=NAME_ATTRIBUTE('revolution/minute',#95);
#140=NAME_ATTRIBUTE('inch/minute',#104);
ENDSEC;
END-ISO-10303-21;
```

Simulation in STEP-NC Machine software was successful. The generated toolpaths are associated to the WS-1 workingstep instantiated created in the C++ code.

Note that this program can be enriched with more machining data by using the classes and methods of the STEPNCLib library or directly from the STEP-NC Machine environment.