

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

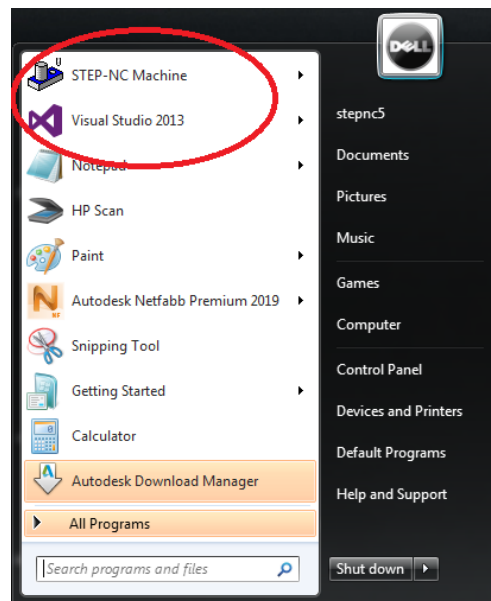
Contributor: Efrain Rodriguez

STEP-NC LaDPRER

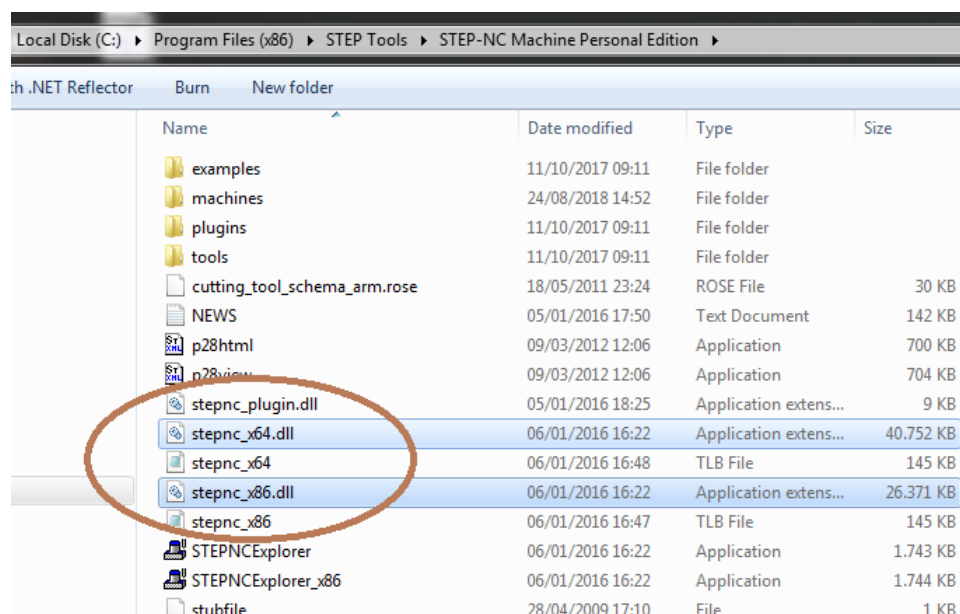
2018

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.

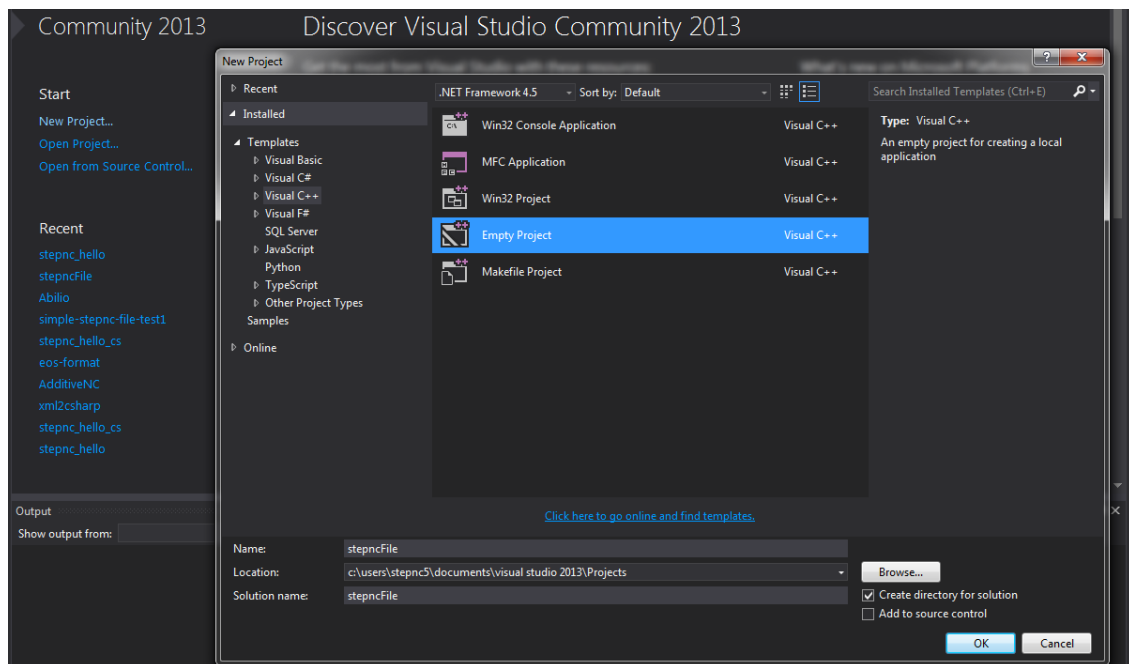
You need **Visual Studio 2013** installed on your **Windows** operating system. You also need the **STEP-NC Machine** software containing the **stepnc.dll** to create the STEP-NC data.



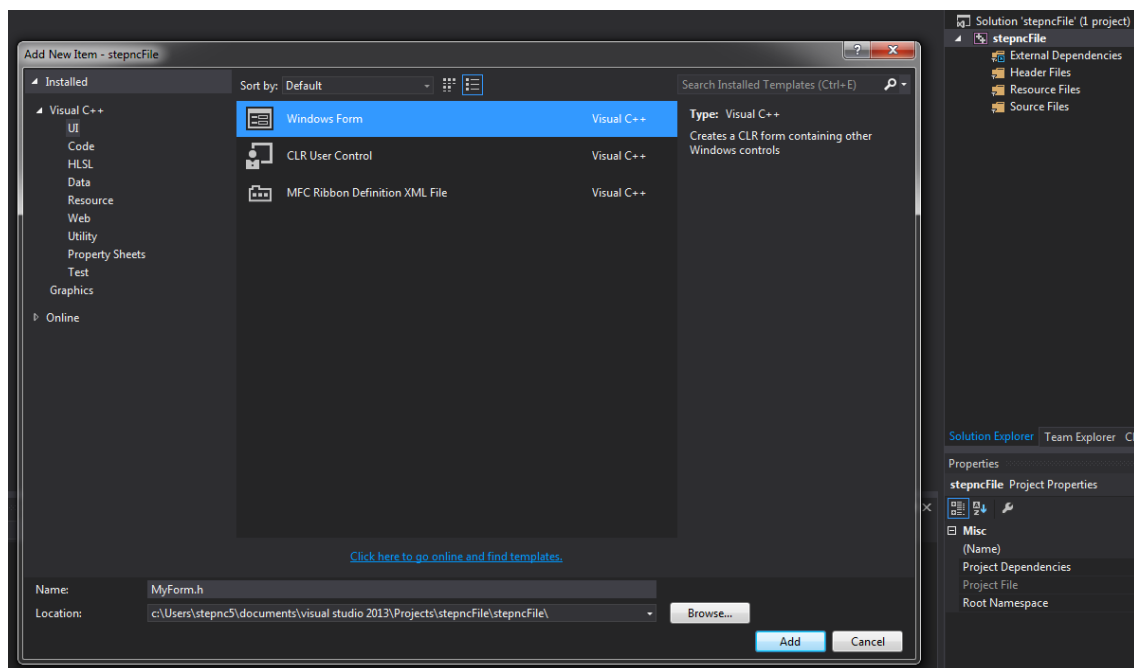
Make sure that the **stepnc_x64.dll** and/or **stepnc_x86.dll** are installed in the STEP-NC Machine directory, usually: **C:\Program Files (x86)\STEP Tools\STEP-NC Machine Personal Edition**.



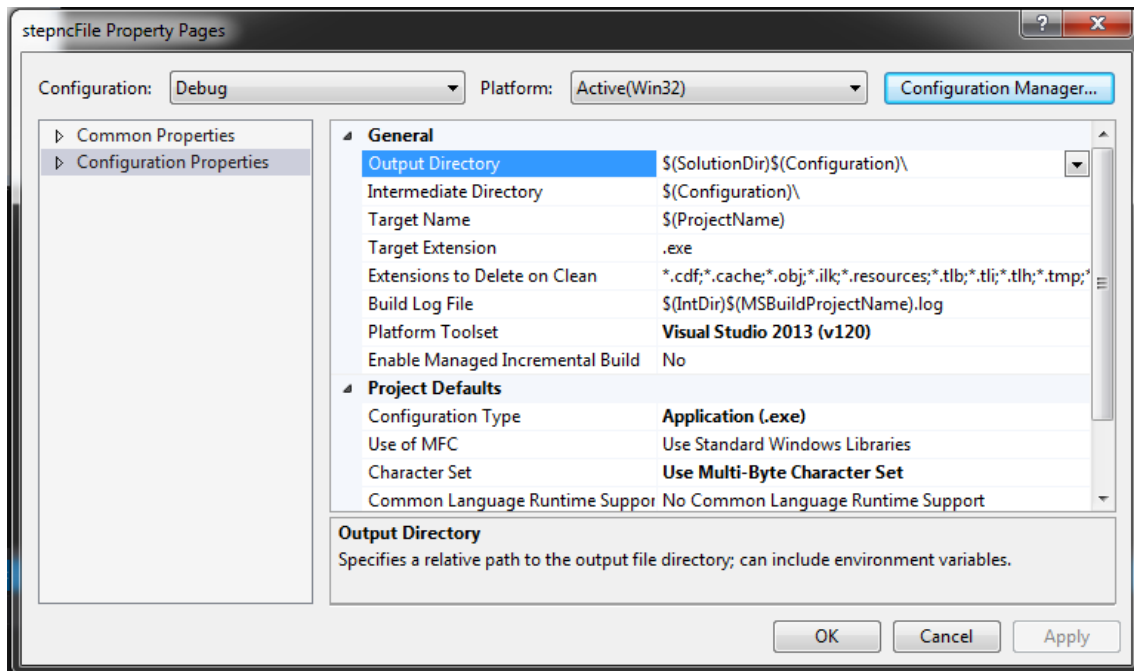
You will need to create a new C++ **Empty Project** by starting Visual Studio. Give your project a name; for example, “stepncFile”.



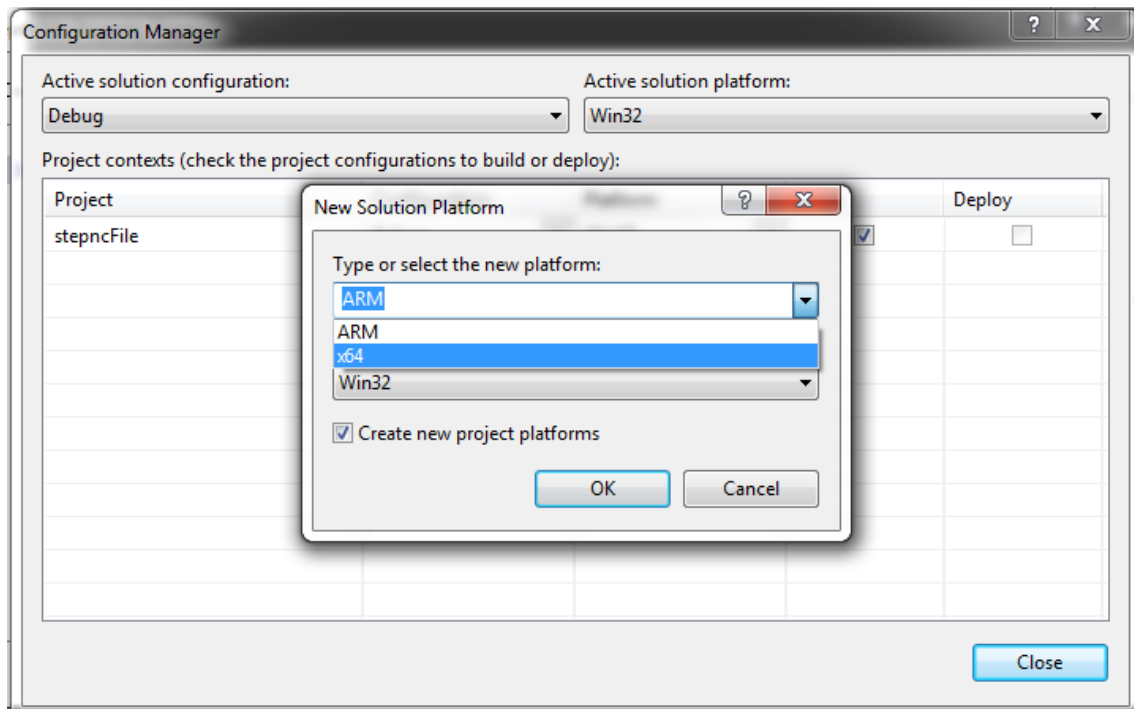
Then add a new **Windows Form** item and give it the name you want; for example, “MyForm”.



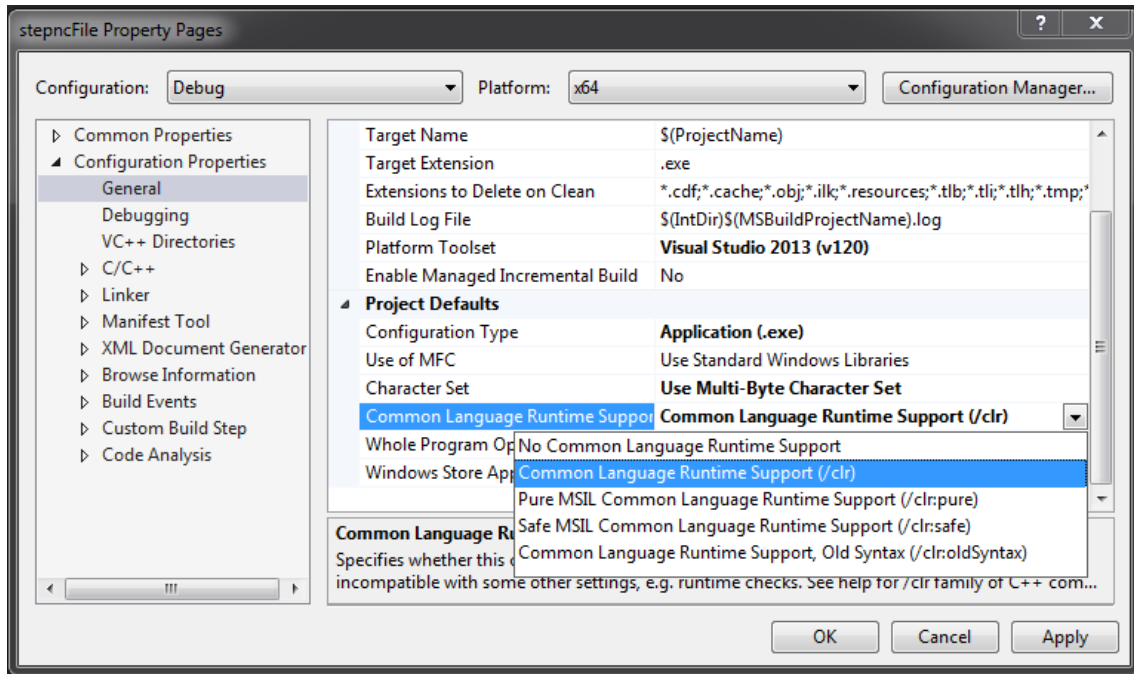
By right clicking on the project, select **Properties** from the menu that will appear to open the project **Property Pages**. Then click on **Configuration Manager**.



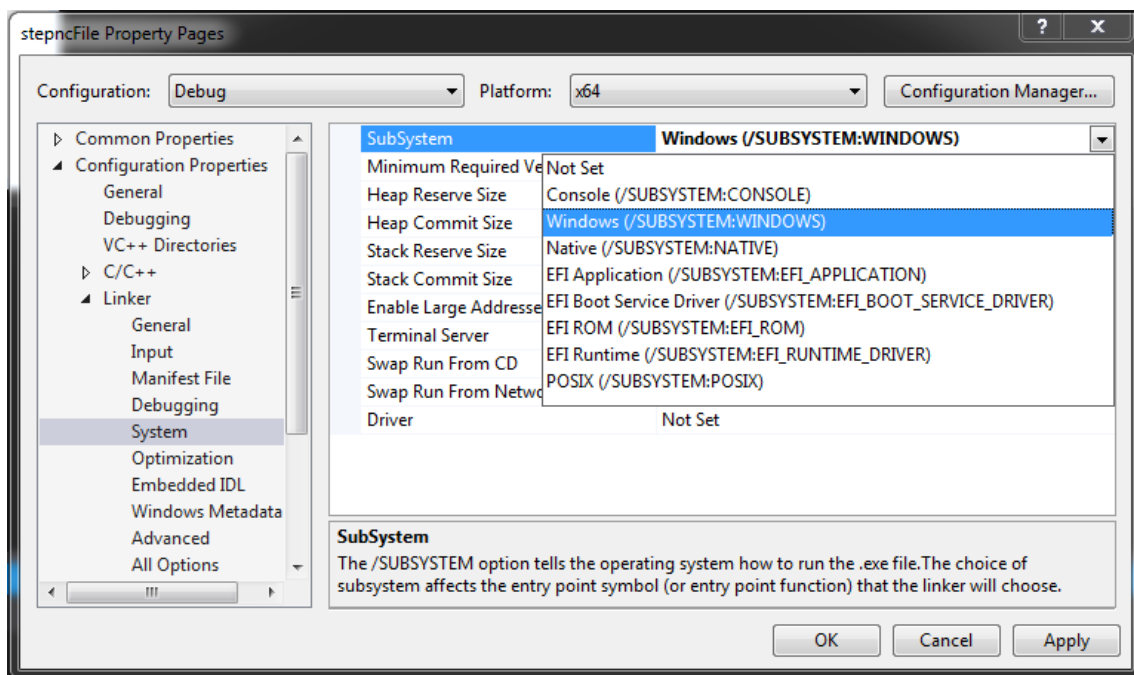
Select **New** from the **Active solution platform** menu and choose the **x64** platform. Press on **OK** and close the **Configuration Manager** box.



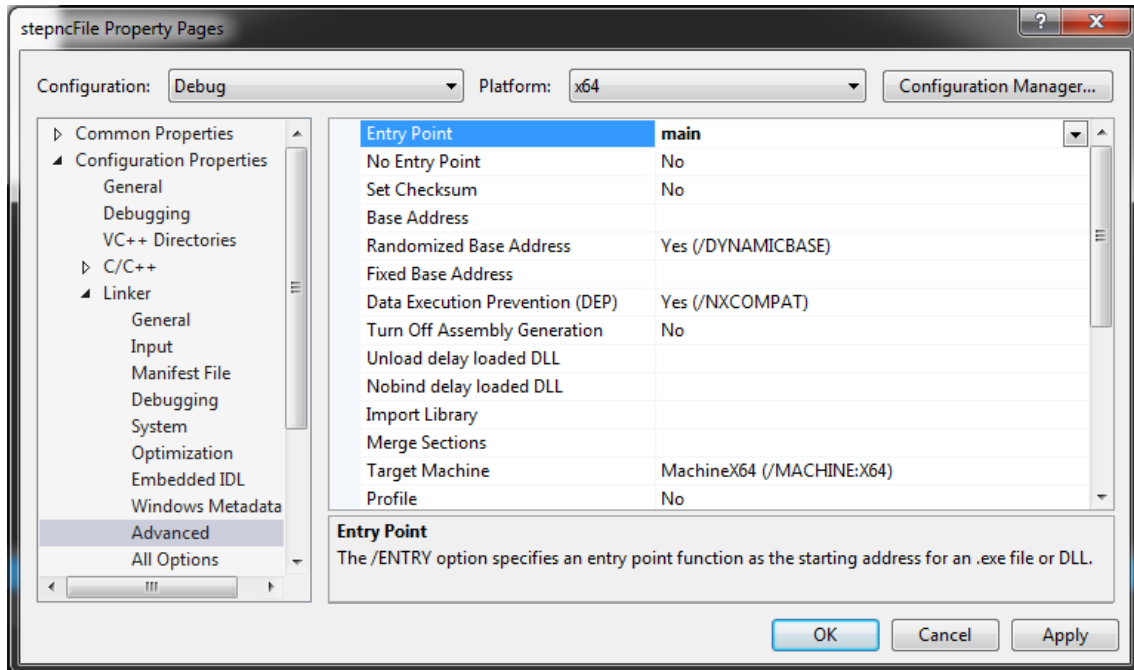
Continuing in the **Property Pages**, in the **Configuration Properties** drop-down list, go to **General** and select **Common Language Runtime Support (/clr)** from the menu shown in the figure.



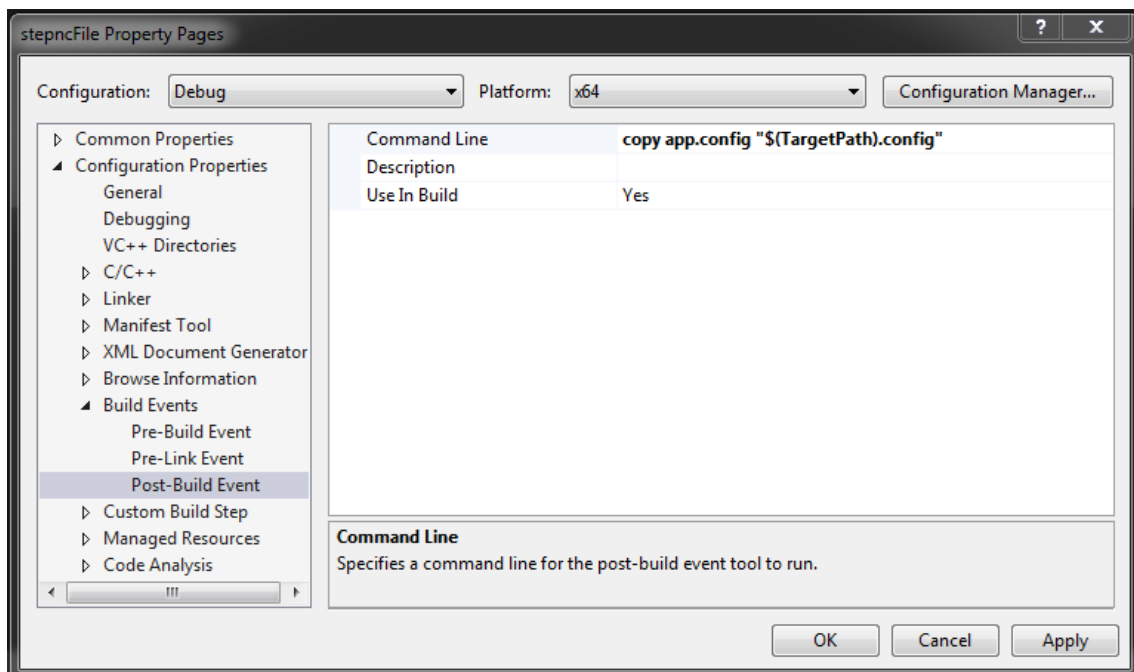
In the path **Linker->System->SubSystem** select **Windows (/SUBSYSTEM:WINDOWS)** for execution environment of Windows (GUI) application.



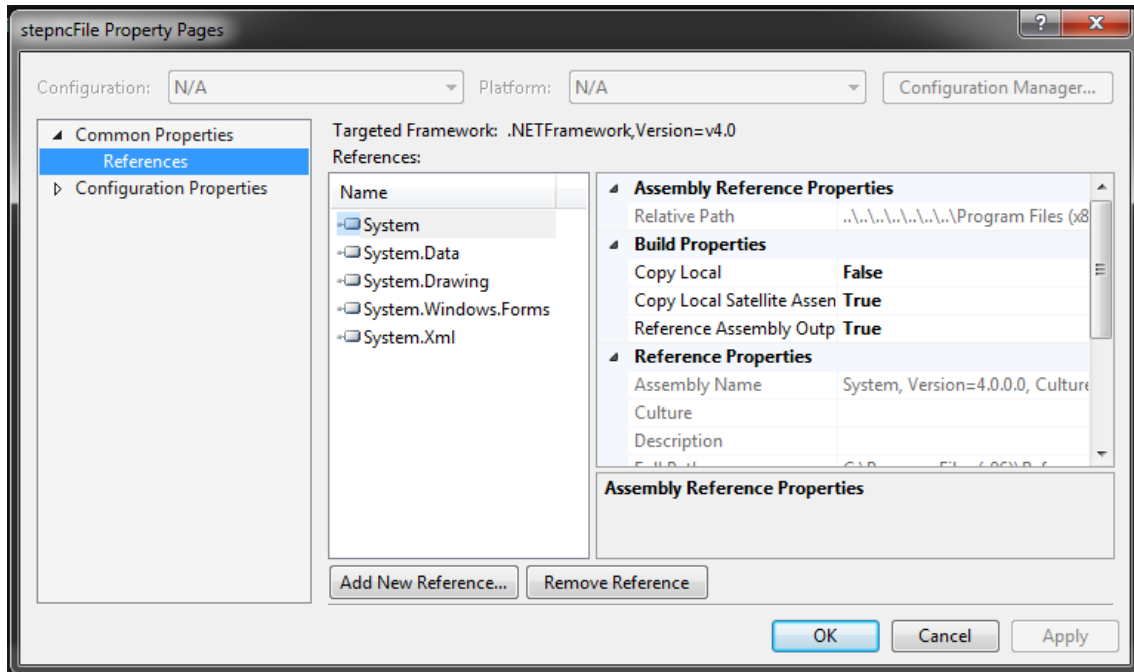
Define the starting address of the project as “main” function in **Linker->Advanced->Entry Point**.



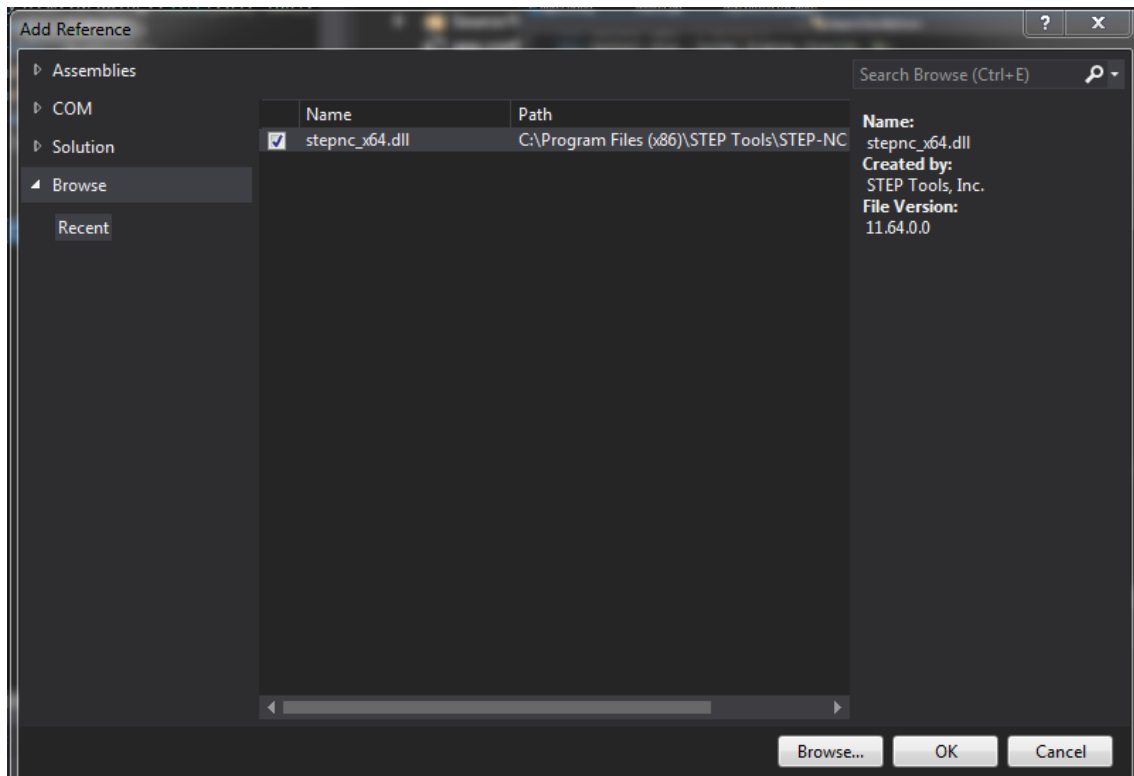
Specify a post-build events for the **Build Events** page by typing the sentence **copy app.config “\$(TargetPath).config”** in the **Command Line** edit box.



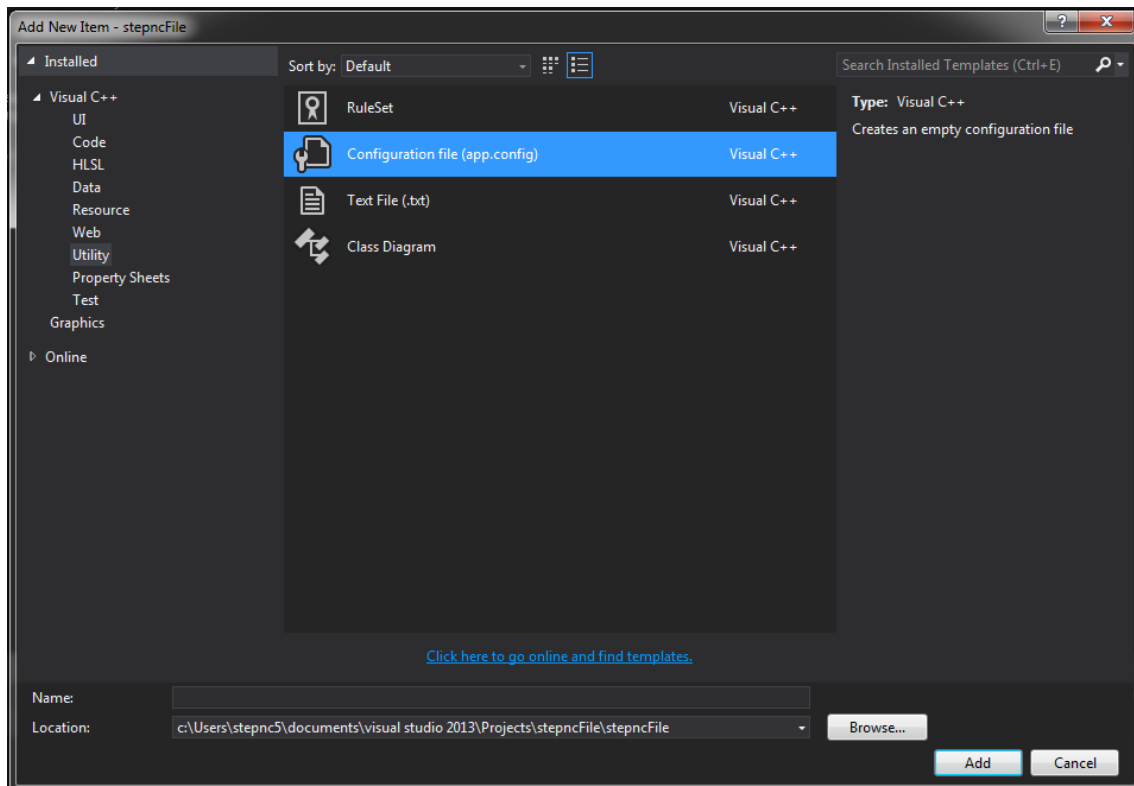
An important step is to add the **stepnc.dll** as within the reference from the project to use its functions in the building the STEP-NC data. You can add that stepnc.dll accessing by **Common Properties->References** and clicking on **Add New Reference**.



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:

```
app.config  MyForm.h  MyForm.cpp
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
</configuration>
```

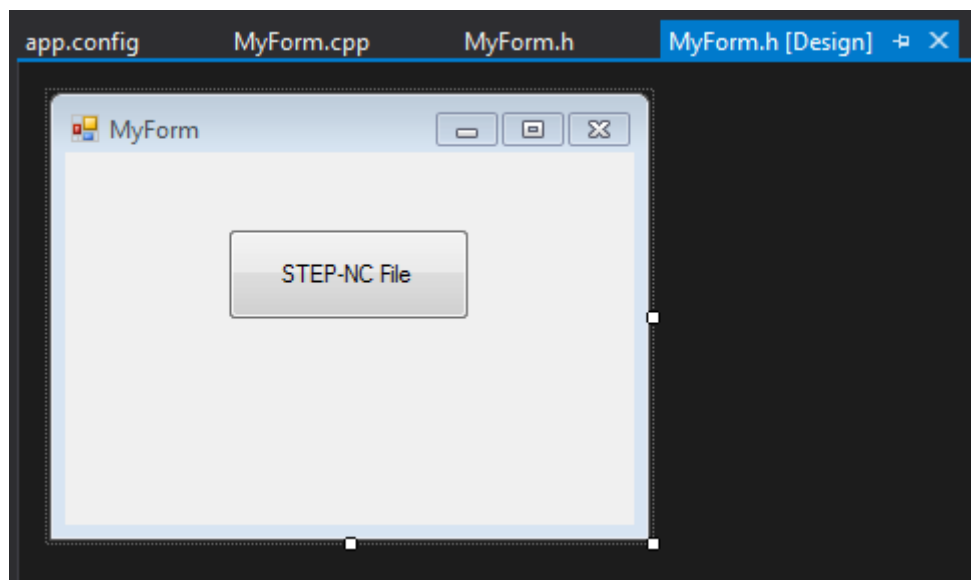

In the **MyForm.cpp** file of the project place the following code necessary:

```
MyForm.cpp  stepncFile  (Global Scope)
#include "MyForm.h"

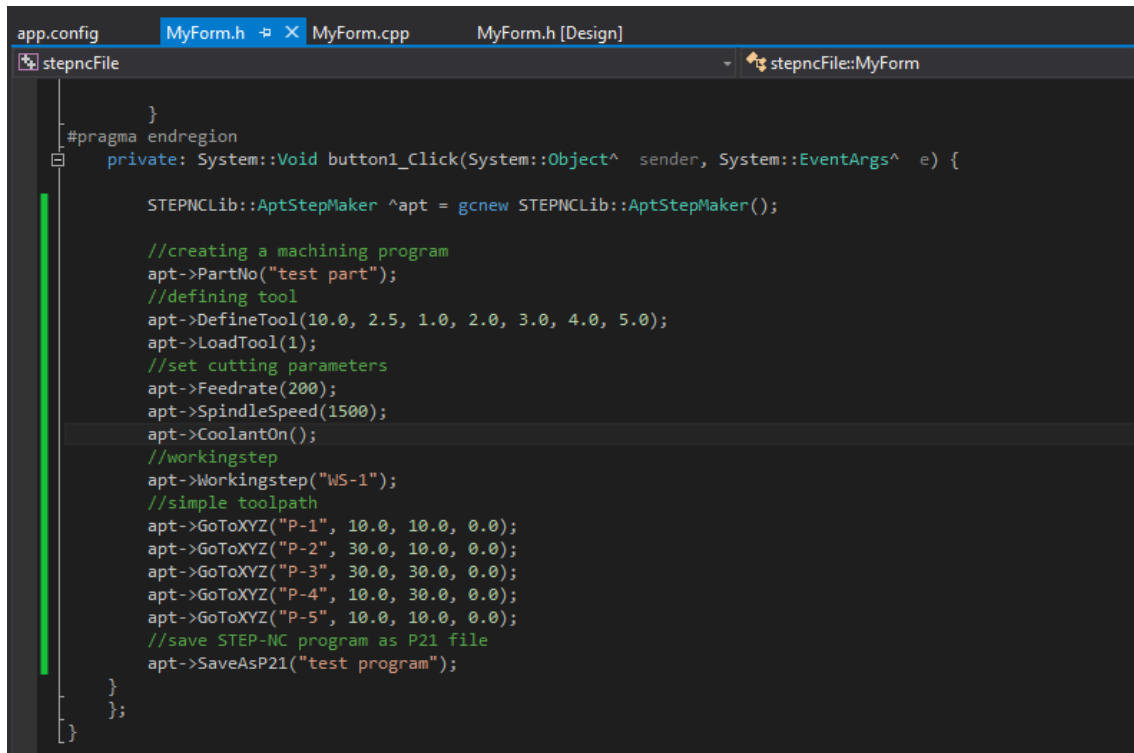
using namespace System;
using namespace System::Windows::Forms;

[STAThreadAttribute]
void main(array<String^>^args)
{
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew stepncFile::MyForm);
}
```

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.



```
    }  
    #pragma endregion  
    private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {  
  
        STEPNClib::AptStepMaker ^apt = gcnew STEPNClib::AptStepMaker();  
  
        //creating a machining program  
        apt->PartNo("test part");  
        //defining tool  
        apt->DefineTool(10.0, 2.5, 1.0, 2.0, 3.0, 4.0, 5.0);  
        apt->LoadTool(1);  
        //set cutting parameters  
        apt->Feedrate(200);  
        apt->SpindleSpeed(1500);  
        apt->CoolantOn();  
        //workingstep  
        apt->Workingstep("WS-1");  
        //simple toolpath  
        apt->GoToXYZ("P-1", 10.0, 10.0, 0.0);  
        apt->GoToXYZ("P-2", 30.0, 10.0, 0.0);  
        apt->GoToXYZ("P-3", 30.0, 30.0, 0.0);  
        apt->GoToXYZ("P-4", 10.0, 30.0, 0.0);  
        apt->GoToXYZ("P-5", 10.0, 10.0, 0.0);  
        //save STEP-NC program as P21 file  
        apt->SaveAsP21("test program");  
    }  
};
```

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.

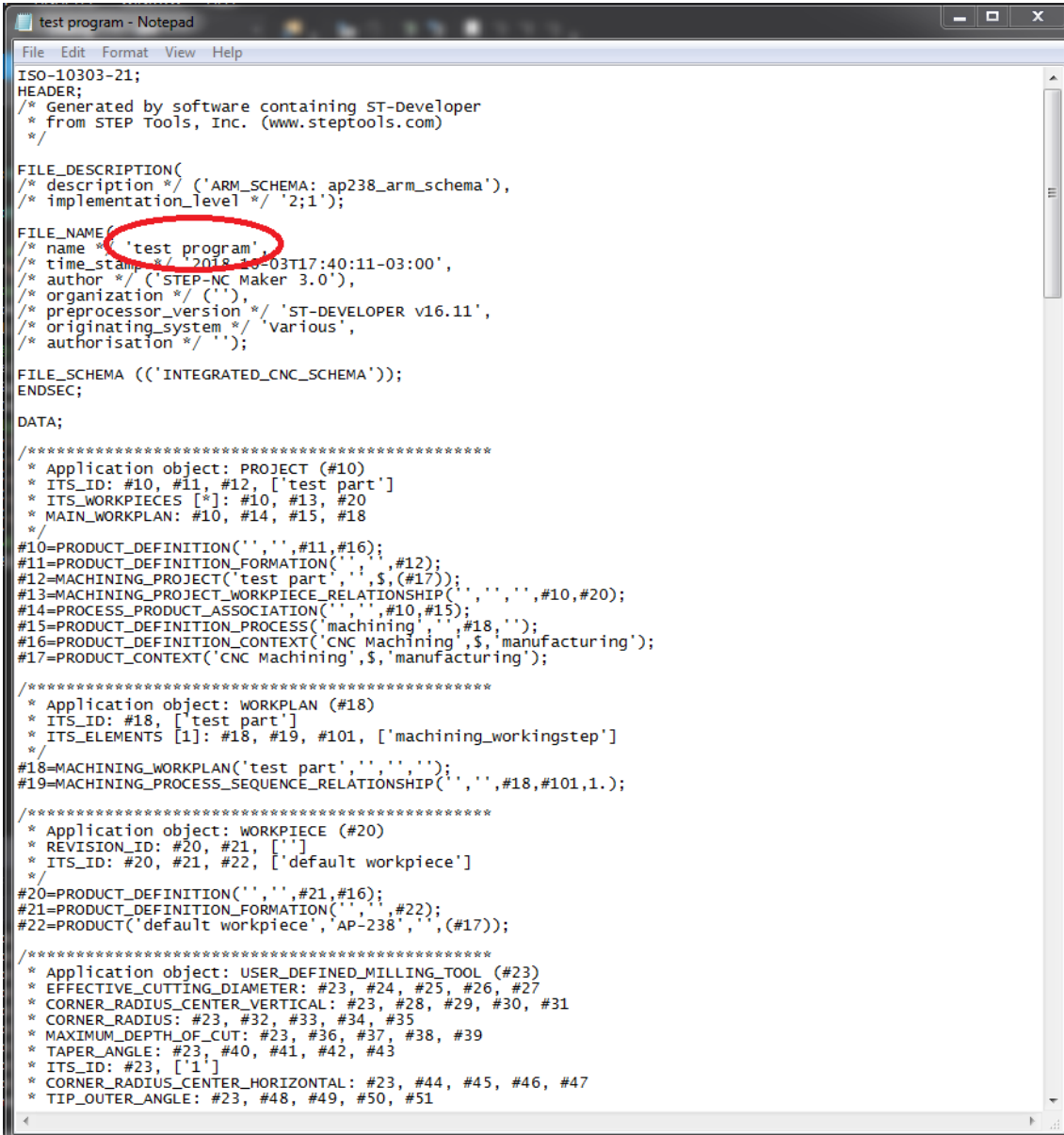
Workingstep method will allow you create a workingstep within the machining program passing its label as input parameter.

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!



```
test program - Notepad
File Edit Format View Help
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com)
 */

FILE_DESCRIPTION(
/* description */ ('ARM_SCHEMA: ap238_arm_schema'),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'test program',
/* time_stamp */ '2018-10-03T17:40:11-03:00',
/* author */ ('STEP-NC Maker 3.0'),
/* organization */ (''),
/* preprocessor_version */ 'ST-DEVELOPER v16.11',
/* originating_system */ 'Various',
/* authorisation */ (''));

FILE_SCHEMA (('INTEGRATED_CNC_SCHEMA'));
ENDSEC;

DATA;

/* *****
 * Application object: PROJECT (#10)
 * ITS_ID: #10, #11, #12, ['test part']
 * ITS_WORKPIECES [*]: #10, #13, #20
 * MAIN_WORKPLAN: #10, #14, #15, #18
 */
#10=PRODUCT_DEFINITION('', '', #11, #16);
#11=PRODUCT_DEFINITION_FORMATION('', '#12');
#12=MACHINING_PROJECT('test part', '#17');
#13=MACHINING_PROJECT_WORKPIECE_RELATIONSHIP('', '#10, #20');
#14=PROCESS_PRODUCT_ASSOCIATION('', '#10, #15');
#15=PRODUCT_DEFINITION_PROCESS('machining', '#18');
#16=PRODUCT_DEFINITION_CONTEXT('CNC Machining', '$, manufacturing');
#17=PRODUCT_CONTEXT('CNC Machining', '$, manufacturing');

/* *****
 * Application object: WORKPLAN (#18)
 * ITS_ID: #18, ['test part']
 * ITS_ELEMENTS [1]: #18, #19, #101, ['machining_workingstep']
 */
#18=MACHINING_WORKPLAN('test part', '#19');
#19=MACHINING_PROCESS_SEQUENCE_RELATIONSHIP('', '#18, #101, 1.);

/* *****
 * Application object: WORKPIECE (#20)
 * REVISION_ID: #20, #21, ['']
 * ITS_ID: #20, #21, #22, ['default workpiece']
 */
#20=PRODUCT_DEFINITION('', '#21, #16);
#21=PRODUCT_DEFINITION_FORMATION('', '#22);
#22=PRODUCT('default workpiece', 'AP-238', '#17));

/* *****
 * Application object: USER_DEFINED_MILLING_TOOL (#23)
 * EFFECTIVE_CUTTING_DIAMETER: #23, #24, #25, #26, #27
 * CORNER_RADIUS_CENTER_VERTICAL: #23, #28, #29, #30, #31
 * CORNER_RADIUS: #23, #32, #33, #34, #35
 * MAXIMUM_DEPTH_OF_CUT: #23, #36, #37, #38, #39
 * TAPER_ANGLE: #23, #40, #41, #42, #43
 * ITS_ID: #23, [1]
 * CORNER_RADIUS_CENTER_HORIZONTAL: #23, #44, #45, #46, #47
 * TIP_OUTER_ANGLE: #23, #48, #49, #50, #51
```

```

#1=MACHINING_PROPERTY_REPRESENTATION('',milling,#10,#10);
#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',#83,');
#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',#86);
#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.);
#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);
#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,#125,#126,#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.

