Industrial Robotic CNC Machining using PowerMILL Robot
Brian Ringley – Design Technology Platform Specialist, Woods Bagot

PE4996-L  Industrial robotic automation is becoming the standard for manufacturing sophistication and efficiency in many industries. PowerMILL Robot combines the CAM control that users have come to know and love from Delcam with intuitive robotic-arm configuration and motion control. This course will begin by using PowerMILL machine tool configurations to define a basic 6-axis robotic arm from an articulated CAD model. We will then move into part setup for both a static situation and a dynamic situation using an external workpiece positioner for our seventh axis. We will create machining approaches on the fly by moving the robot joints separately or in tandem and then creating workplanes from our tool center point. Finally, we will use PowerMILL finishing toolpaths with the robotic arm and learn how to manage and monitor robotic joint motion using PowerMILL Robot simulation tools.

Learning Objectives
At the end of this class, you will be able to:

- Learn how to define your robotic arm using template libraries and custom machine tool configurations
- Learn how to position the part relative to the robot in a static position and on an external axis
- Learn how to create workplanes on the fly from robotic arm tool center-point positions
- Learn how to manage and monitor robotic joint motion using robot simulation tools

About the Speaker
Brian Ringley is the design technology platform specialist at Woods Bagot, where he specializes in project-specific design technology integration, design computation methodology, and Building Information Modeling (BIM). Prior to working at Woods Bagot, he was the Fuse Lab technology coordinator for the Department of Architectural Technology at the New York City College of Technology (CUNY), and he was the digital fabrication coordinator for the College of Design, Architecture, Art, and Planning at the University of Cincinnati. He has worked in the architectural offices of Kohn Pederson Fox Associates, Dellekamp Arquitectos, and R&Sie(n), and he has taught fabrication seminars and design studios at the University of Cincinnati, the City University of New York, and Pratt Institute. He has also led design-build workshops at Rensselaer Polytechnic Institute and plusFARM, and he has sat on juries as an invited critic at Columbia Graduate School of Architecture, Planning, and Preservation and Cornell University College of Architecture, Art, and Planning. He is a regular contributor to Designalyze.com and AEC-APPS.com.

email: brian.ringley@woodsbagot.com
Twitter: @brianringley
## Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining a Robotic Arm</td>
<td>04</td>
</tr>
<tr>
<td>Getting Started with PowerMILL Robot</td>
<td>04</td>
</tr>
<tr>
<td>Plugin Windows</td>
<td></td>
</tr>
<tr>
<td>Loading a Robot</td>
<td></td>
</tr>
<tr>
<td>Defining a Custom Machine Tool</td>
<td>06</td>
</tr>
<tr>
<td>Download CAD</td>
<td></td>
</tr>
<tr>
<td>File Conversion Using Delcam Exchange</td>
<td></td>
</tr>
<tr>
<td>Editing the Machine Tool File</td>
<td></td>
</tr>
<tr>
<td>Robotic Configuration</td>
<td>11</td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td>Robotic Arm Toolpathing</td>
<td>14</td>
</tr>
<tr>
<td>Loading a Robot into an Existing Program</td>
<td></td>
</tr>
<tr>
<td>Defining Custom Stock</td>
<td></td>
</tr>
<tr>
<td>Reach/Approach Limitations</td>
<td></td>
</tr>
<tr>
<td>Robotic Roughing</td>
<td>17</td>
</tr>
<tr>
<td>Limiting a Custom Block</td>
<td></td>
</tr>
<tr>
<td>Starting a Robot Simulation</td>
<td></td>
</tr>
<tr>
<td>Fixed Tool Axis</td>
<td></td>
</tr>
<tr>
<td>Dynamic Tool Axis</td>
<td></td>
</tr>
<tr>
<td>Aligning Workplanes to Tool Axis</td>
<td></td>
</tr>
<tr>
<td>Working with External Axes</td>
<td>24</td>
</tr>
<tr>
<td>Toolpathing with External Positioner</td>
<td>24</td>
</tr>
<tr>
<td>Positioning the Part onto the Positioner</td>
<td></td>
</tr>
<tr>
<td>Roughing with External Positioner</td>
<td></td>
</tr>
<tr>
<td>Flat Finishing</td>
<td></td>
</tr>
</tbody>
</table>
Embedded Pattern Finishing

Outputting Robot Toolpaths as NC Programs  …  …  …  …  …  29

Recording Simulations

Adding Toolpaths to NC Program
Defining a Robotic Arm

Getting Started with PowerMILL Robot

Plugin Windows
To pull up additional windows for PowerMILL plug-ins, right-click anywhere within the top tool bar area and select **Vertical Plugin Window (VPW)** and **Horizontal Plugin Window (HPW)** from the menu. The only plugin features showing should be those for **PowerMILL Robot**.
Loading a Robot

To load a robot or pre-made robotic configuration, go to the VPW and expand the Robot Library drawer to reveal PowerMILL’s default robot library, organized by manufacturer. In this course we’ll be using ABB robots so expand the ABB drawer to reveal a single robot, the IRB 6640.

Expanding the robot model reveals a series of robotic configurations, including the robot by itself or the robot in combination with assets such as tracks, external positioners, and working envelopes. Some configurations even allow you to place the part on a robot to be milled by an additional robot or a stationary spindle, as opposed to a stationary part milled by the robot. This is achieved by locating the default workplane at the tool center point (TCP) rather than the robot base.

Double-click on IRB6640 + LinearTrack to load the configuration. You will see that these configurations load as Machine Tools in PowerMILL Explorer (PE). Double-click an item again to unload it, or double-click a different robotic configuration to swap it out.
Defining a Custom Machine Tool

Download CAD

To create your particular robotic setup you will likely need to define your own custom machine tool. The first step in this process is obtaining a 3D model of your robot. This model will need to be separated into its individual components. For example, a typical 6 axis robotic arm is made up of 7 components: 6 joints and a base.

It is very likely that between your PowerMILL reseller and robot manufacturer you will be provided with a 3D model of your robot. We will be defining the ABB IRB 140. To download the model, I can simply visit the CAD downloads page for the IRB 140 at http://new.abb.com/products/robotics/industrial-robots/irb-140/irb-140-cad. If you are given the option for the “complete” file versus the “joints” or “joints-links” files, it’s best to go with the latter as each joint model will need to be separately converted into a Delcam Machining Triangles (*.dmt), the file paths of which will be referenced into the Machine Tool XML file. I will download the M2004 IRB 140 IGES joint files, also available in the course resources folder.
File Conversion Using Delcam Exchange

Launch **Delcam Exchange** and use the **File > Import** option to bring in the first file, **IRB140_-_M2004C_BASE.IGS**. On the left of the screen there will be a **Tree View**. In this view, right click on **Level 0** and select **Select Item** to select the entirety of the base geometry. Then select **File > Export**. In the **Export File Details** set **File Format** to **DMT** and click on the **Browse** button. In the **Output File** window browse to  `C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140\IRB 140`, set **File name** to **Base.dmt**, and hit **Save**. This library directory is linked directly to the **Robot Library** within PowerMILL’s **VPW**.

*Note: If you can’t write to the specified directory due to user permissions, make your own directory and we can add it to the library later using the **Add a robot** button in the **VPW’s Robot Library** drawer.*

As IGES faces from multiple robot components could mix within Level 0 if we import all IGES files into a single Exchange session, it is recommended to open a new Exchange session each for each file conversion. Repeat this process for the 6 remaining IGES files and name them using the following convention:

- **LINK1.IGS** as **AB.dmt**
- **LINK2.IGS** as **BC.dmt**
- **LINK3.IGS** as **CD.dmt**
- **LINK4.IGS** as **DE.dmt**
- **LINK5.IGS** as **EF.dmt**
- **LINK6.IGS** as **FG.dmt**
Editing the Machine Tool File

Delcam’s **Machine Tool File (MTD)** is a custom **Extensible Markup Language (XML)** schema authored by Delcam to allow simple manipulation of machine joint configurations. In its most basic format, it looks something like the image below, where each subsequent linked joint is a nested `<machine_part>` tag within an overall `<machine>` tag. Non-linked joints, such as the base, sit side-by-side within the machine tag. It’s recommended to use an advanced text editor such as **Notepad++** which color codes the various aspects of the XML for easier editing. (See [IRB140_basic01.xml](#) in the course resources.)

```xml
<machine xmlns="x-schema:PowermillMachineTool" POST="SimPost.*">
    <machine_part NAME="RobotBase">
    </machine_part>
    <machine_part NAME="RobotAxis1">
        <machine_part NAME="RobotAxis2">
            <machine_part NAME="RobotAxis3">
                <machine_part NAME="RobotAxis4">
                    <machine_part NAME="RobotAxis5">
                        <machine_part NAME="RobotAxis6">
                            <machine_part NAME="head" />
                        </machine_part>
                    </machine_part>
                </machine_part>
            </machine_part>
        </machine_part>
    </machine_part>
</machine>
```

Define the CAD geometry by inserting both a `<path/>` tag to specify file location and a `<rgb/>` tag to specify color within a `<triangle_file>` tag, and the place this entire thing within a `<model_list>` tag which also serves to control model opacity. Each `<machine_part>` tag should contain a `<model_list>` tag which defines the one or more **DMT** elements making up that part. (See [IRB140_basic02.xml](#).)

```xml
<machine_part NAME="RobotBase">
    <model_list OPACITY="100">
        <triangle_file>
            <path FILE="C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140\IRB 140\Base.dmt" />
            <rgb R="0" G="0" B="0" />
        </triangle_file>
    </model_list>
</machine_part>
```
While immovable parts such as the base can be defined solely with a `<model_list>` tag, elements which have motion also require an `<axis>` tag. Within this `<axis>` tag are two other tags – one called `<control_info/>` to define things such as the range of motion of the joint in degrees, and another called `<simple_rotary/>` to define the rotational base and vector. There are other motion type tags but the simple rotary type is sufficient to define the motion of a 6 axis robot. (See IRB140_basic03.xml.)

Note: Joint rotational limits can be found on a robot's technical data sheet, typically available through the manufacturer's website. Rotational base points, if not supplied within the technical data, must be measured from the CAD model using a CAD modeler such as Delcam PowerSHAPE or Inventor.

```
<machine_part NAME="RobotAxis1">
  <axis>
    <control_info ADDRESS="A" MIN="-360" MAX="360" HOME="0" VALUE="0" PRIORITY="MEDIUM"/>
    <simple_rotary X="0" Y="0" Z="0" I="0" J="0" K="1"/>
  </axis>
  <model_list OPACITY="100">
    <triangle_file>
      <path FILE="C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140\IRB 140\AB.dmt"/>
      <rgb R="0" G="0" B="0"/>
    </triangle_file>
  </model_list>
</machine_part>
```

We still require a spindle as our end effector so we have something into which we can place our end mills. Place the file SpindleOnHead.dmt from, available as a course resource, into the library folder with the rest of the robot geometry. Then add the corresponding triangle file definition to the RobotAxis6’s model list. The spindle can be included within this list instead of being placed within its own machine part because it does not move independently of axis 6. (See IRB140_basic04.xml.)

```
<model_list OPACITY="100">
  <triangle_file>
    <path FILE="C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140\IRB 140\SpindleOnHead.dmt"/>
    <rgb R="100" G="100" B="100"/>
  </triangle_file>
  <triangle_file>
    <path FILE="C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140\IRB 140\FG.dmt"/>
    <rgb R="0" G="0" B="0"/>
  </triangle_file>
</model_list>
```
We're almost done with the file – we just need to add some finishing touches into the XML:

- The `<head_attach_point/>` tag defines where the end mill attaches to the end of the spindle or whatever element is defined as the terminus of the **Machine Tool**.
- The `<table_attach_point/>` tag defines where the robot base attaches to the table – we're not using a table so the robot attaches to the default workplane at the global origin.
- The `<collision_exclusion_list>` tag and its interior tags define which combinations of elements should be excluded from collision detection during robotic program simulation.
- Even though we are not using one, it’s necessary to include a machine part for the table. Because we’re not defining geometry we can just close the tag. You’ll notice we did the same thing for the head if you look through the code thus far. (See IRB140_basic05.xml.)

```xml
<head_attach_point PART="head" X="575" Y="0" Z="600" I="0" J="0" K="1" U="1" V="0" W="0" />
<table_attach_point PART="table" X="0" Y="0" Z="0" I="0" J="0" K="1" U="1" V="0" W="0" />
<collision_exclusion_list>
  <collision_group>
    <collision_part NAME="RobotBase"/>
    <collision_part NAME="RobotAxis1"/>
  </collision_group>
</collision_exclusion_list>
<machine_part NAME="table"/>
```

Rename IRB140_basic05.xml to IRB140.mtd and place in C:\Program Files (x86)\Delcam\Delcam PowerMILL Robot PlugIn 2015 (8.0.09)\Library\Robots\ABB\IRB 140. Now you can import the robot into PowerMILL by double-clicking on the IRB140 object which should now have appeared within the **Robot Library**.

The completed progress file **myRobot01** is available within the course resources.
Robotic Configuration

Configuration

When loading an unconfigured robot you will receive a message that a new configuration file has been generated. This file is a *.RobConfig file and lives next to the Machine Tool Definition (*.mtd) file in the robot model’s root library folder.

A Configuration window will appear where you can make changes to the configuration. Navigating with the menu on the left-hand side, select Robot:

1. Name the robot myIRB140.
2. Set Robot manufacturer to ABB.
3. Set Robot kinematic to “The robot holds the end effector | The part is on the table.”
Select **Dimensions** and verify that the critical dimensions shown match those listed in your robot technical data or those measured from your actual robot.

Select **MTD informations** where you can review the head and table attach positions (XYZ), vectors (IJK), and orientations (UVW) as well as see where your MTD file is located.
Select *Workplanes* where you can override default values for *Robot calibration position* with joint rotational values for each axis including any external axes (default is all 0’s), the *Robot world workplane* (manufacturer’s default is robot base center), and *Robot 6th axis workplane* (flange) (manufacturer’s default is pointing straight outward from the center of the flange’s front face). You can also set *User defined workplanes* for custom locations and orientations for things like tool changing and jigs for welder tip maintenance. You will notice that these workplanes will not appear in PE as they are considered part of the robot and not their own document objects.

*Simulation* options are for setting up tool change sequence and *Tool database* options are for specifying the source for tool data. Make sure that the *Postprocessor* is set to ABB and hit *Save*. (You will likely need to work with your PowerMILL reseller and/or robot manufacturer to develop the Postprocessor that works best for your particular robotic model and setup.) Reload the robot from the library to complete the configuration. The *Configuration* window can be reached anytime by expanding the *Robot Cell* drawer in the VPW and selecting the *Robot configuration* button.
Robotic Arm Toolpathing

Loading a Robot into an Existing Program

Defining Custom Stock

Open the sample file partForRobot. You may notice the unusual block shape, this is a custom block defined by an STL (triangle) file. Materials like RenShape and other high density polyurethane tooling boards are extremely expensive, so this is a representation of a custom glue-up of blocks to efficiently limit the location of the stock to the general massing of the sculptural part being milled.

Custom blocks like this are defined within the Block form by setting Defined by to Triangles and selecting the Load block from file button, then browsing to the appropriate triangle file (as in a file format which supports triangulated mesh faces).
Reach/Approach Limitations

Robotic arms are helpful for many reasons, one of the more obvious being their abilities to approach a part from multiple vantages, and while the rigidity falls short of a high-end 5-axis machine, the repeatability and extra axis for reach, as well as the ability to quickly additional, external axes, make the robot a real asset for milling complex forms without the need to break them into multiple parts, each potentially with multiple machining positions.

Within this 3 axis program we have the following major issues:

1. **Collisions**, primarily the result of the part being too tall for a standard 3 axis machining envelope.
2. **Reach issues**, primarily the result of the many undercuts from a single 3 axis approach.
3. **Flat area detection failure** because we cannot change our axis of approach with this method.

Repositioning the Part Relative to the Robot

Load the **IRB140** into the program. You may receive a collision as the robot could pop into the session right on top of your part. Regardless of whether or not there's a collision, you will likely want to reposition the part from its current position, optimized for a 3 axis setup, to a new position relative to the robot from which it can reached by the robot's end-of-arm tooling.
To move the part without losing any toolpath data, expand the Robot Cell drawer in the VPW and select the Part positioning button. Here you can select a pre-existing workplane for part repositioning, or enter in translation and rotation values per axis. In the Part positioning form:

1. Set Workplane name to partRelToRobot (this name will be appended with the transformation information and is optional).
2. Set X to 250.
3. Set Y to -200.
4. Set Z to 100.
5. Leave the rotation fields blank
6. Select Set new origin to generate the new workplane.

Note: Creating multiple part positioning workplanes with incremental rotations is one way to manually rotate the part position if no external axis is present.
Robotic Roughing

Limiting a Custom Block

Programming an efficient robotic toolpath that is entirely reachable and free of both self and part collisions is a bit of an art, so I recommend limiting the toolpath to something we’re positive is approachable and then expanding incrementally from there. Currently we have no external axis so we know we may as well limit the toolpath to the portion of the part facing the robot. Because we’re using a custom block (and this is really only an issue with an area clearance toolpath) we can’t limit the toolpath by changing the block extents, so we have to use an alternative approach.

Right-click on the 010_rough toolpath under Toolpaths in PE, then select Settings. Hit the Clone button to preserve the original toolpath and rename the duplicate toolpath 010_rough_robot. In the Model Area Clearance toolpath form go to Limit and then drop-down the Boundary type list. Select a User-defined boundary and in the boundary form select the Sketch Boundary button. Select the View from top (Z) icon to get an overhead view and use the Rectangle tool from the sketch toolbar to draw a boundary that only encompasses the near side of the part relative to the robot. Then click on the green checkmark button to Accept changes.

This will limit the toolpath in the X and Y axes. To limit the Z axes, stay in Limit and set the Z limit maximum to 500 and the minimum to 300. Calculate the toolpath.
Starting a Robot Simulation
To start a robot simulation, expand the Robot Control drawer within the VPW and select the Attach tool to start button. This will position the robot and its TCP at the start point for the toolpath with the initial orientation matching that with which the toolpath was generated. You will also notice that the rotation value for your 6 axes has updated. Hitting the Home button will return the robot to its home position. Hit the play to run the simulation. If no errors appear, try dropping the Z limit minimum to 100. At this point, you should be receiving a collision error where the spindle is colliding into the part.

Note: Portions of the robot involved in the collision will turn red.
Fixed Tool Axis

If our tool approach is fixed vertically, as it is currently, collisions between the part and the spindle are inevitable. Let’s adjust our tool orientation. To do this, Recycle the toolpath and rename it 010_rough_robot_upper. Go to Tool Axis and select Fixed direction from the Tool axis drop-down menu, and define Direction as:

1. \( I = -0.8 \)
2. \( J = 0.0 \)
3. \( K = 1.0 \)

This IJK vector will give our tool a 36 degree tilt from Z up, enough to avoid hitting the part with the spindle, but not so much as to cause joint configuration or reach issues. This vector will appear as a red arrow in PowerMILL. You may receive an alert that end mills are not permitted for multi-axis work, so remember to go to Tool and set your tool to the 20mm_ballMill ball nosed tool.

Go to Limit and relax the Z limit Minimum to 60 so that the toolpath gets as much of the part as possible without entering the wider base portion, for which we will use a different strategy. Hit Calculate.
Dynamic Tool Axis

The base portion of our part is difficult to rough because we have to contend with the undercuts which require a negative IJK vector orientation, while at the same time needed to machine the “toes” area where we will need a positive IJK vector orientation to avoid the robot scrunching up on itself and experiencing self-collisions between its joints. Therefore we will need our tool axis to dynamically respond to its position relative to the part. We will do this using a curve such that the robot will take its orientation as the vector from the nearest point on the curve to the toolpath point.

To create this curve, right-click on Patterns within PE, then select Create Pattern. Rename this pattern toolOrient. Right-click on toolOrient and select Curve Editor…. This will open the curve editing interface. Within the top view, select the Bezier curve button from the freeform curve fly-out toolbar and sketch a curve approximately like the one shown below. Hit the green checkmark to Accept changes.

![Bezier curve](image)

Move the curve to the proper Z height by right-clicking the curve and selecting Edit > Transform…. In the Pattern Transform interface select Move geometry and then select the Position form at the bottom of the screen. Type 350 into Z then hit Apply and Accept. Hit the green checkmark to Accept changes.
Now clone the toolpath **010_rough_robot_upper** and rename it **015_rough_robot_lower**. Reference the curve we just sketched by going to **Tool axis** and setting the method to **From curve**. A Pattern drop-down list will appear from which you can select your **toolOrient** curve. **Limit** the toolpath to a **Maximum** of **100** and hit **Calculate** and simulate the toolpath.

Depending on your setup, you may notice a number of wacky things happening, including joint self-collisions, extraneous ramping, and reach issues.

![Diagram of a robot arm with toolpath curves]

**Aligning Workplanes to Tool Axis**

Much of this is resulting from our vertical safe and start heights, which are set to clear the top of the part. Because this distance is so far from where we’re actually trying to machine our flyover moves are causing issues for the robot. Instead of approaching this toolpath from the Z up orientation let’s approach it from 45 degrees off the tool axis. We can do this quickly by aligning a workplane with a tool axis:

1. Go to **Robot Control** and set the robot to the **Home** position.
2. Jog axis **5** to **-45**.
3. Go to **Robot Cell** and select the **Create workplane aligned with robot tool** button.
4. A new workplane called **NcToolWkp** will appear in **PE**.
Recycle the toolpath and set the **Workplane** to **NcToolWkp**. You may notice your block shifting as it will automatically align with the toolpath workplane unless otherwise instructed. With the toolpath form open, open the **Block** form and switch the **Coordinate System** from **Active Workplane** to **Named Workplane**. Then specify the workplane that resulted from our initial part repositioning for the robot.

*Note: Global Transform will also work but specifying a Named Workplane is arguably more reliable.*

Because of the new workplane orientation we will no longer need to limit Z to constrain the toolpath to the lower portion of the part, as the boundary is being projected obliquely and this coincidentally aligns with our desired machining area. Also, we can now reduce the **Rapid Move Heights** for **Safe Z** and **Start Z** to **200** and **175**, respectively.
Hit Calculate and simulate both `010_rough_robot_upper` and `015_rough_robot_lower` using ViewMill's Dynamic Stock Preview.

The progress file `myRobot02` is available as a course resource.
Working with External Axes

Toolpathing with External Positioner

Positioning the Part onto the Positioner
Go to the Robot Library and load the IRB6640 + Positioner. When the robot loads, position the part onto the positioner by going to Robot Cell and selecting Part positioning. In the Part positioning form set:

1. X to -335
2. Y to -200
3. Z to 0 (Actually, set Z to 20 to represent a spoil board between the part and the positioner.)

Roughing with External Positioner
Recycle either one of the rough toolpaths (we’ll only need one from here on out). Remove any limits and set Tool Axis to Towards Point. Set:

1. X to 425
2. Y to 200
3. Z to -100 (to keep the tool from leveling flat with the positioner bed at Z0)
Rename the toolpath to **010_rough_robot** and hit **Calculate**.

Prior to simulating the toolpath you’ll want to activate the external positioner by right-clicking on **E2** in the **Robot Control** panel and setting it to **HIGH**. **E2** represents the rotary motion of the positioner’s bed, whereas **E1** represents the rotation of the positioner’s arm. Run the simulation.

**Flat Finishing**

With multi-axis capabilities we should be able to detect flat areas no matter their plane provided we have a corresponding workplane. To make a workplane for the flat plane capping our part, right-click on **Workplanes** in **PE** and select **Create and Orient Workplane > Workplane Aligned to Geometry**… and then place the workplane onto the flat plane capping the part. Rename the workplane **flat**.

Flat area detection can only occur in quadrant I of a workplane so we will need to right-click on the workplane **flat** and select **Workplane Editor**…. Within the **Workplane Editor** interface, use the position form to move the workplane in the negative X and Y directions. Make sure you are using 20mm_endMill tool and **Tool Axis** is set to **Vertical**, then **Calculate** and simulate the toolpath.
Embedded Pattern Finishing

Instead of a typical surface finish done from multiple vantages via multiple workplanes, and then configuring the robotic motion accordingly, we’re going to drive the motion with a custom curve pattern extracted from the geometry to get the effect of a PowerMILL Surface toolpath without the limitation of needing to have one toolpath per surface.

1. Right-click on Models and select Import Model… to import roboPattern.igs.
2. Right-click on Patterns and select Create Pattern, then rename it roboPattern.
3. Right-click on the model roboPattern and Select All (make sure Wireframe View is enabled).
4. Right-click on pattern roboPattern and select Insert Model.
5. Right-click on pattern roboPattern and select Edit > Embed, then embed with the default settings (embedding a pattern captures surface normal data into the curve vertices for enhanced tool-to-toolpath positioning).

Go to the Toolpath Strategies button and then click on Finishing strategies. From the resulting list, select Embedded Pattern finishing. In the Embedded Pattern Finishing toolpath form go to Embedded pattern finishing and set the Drive curve to roboPattern. Hit OK to close the form.
Now we’re going to create another curve from which our toolpath axis will be oriented. Right-click on Patterns, then select Create Pattern. Name it toolAxis. Right-click on pattern toolAxis and select Curve Editor…. In the Curve Editor interface select the Circle tool and set the Radius to 500. Center the circle roughly on center with the flat capping surface of the part (I like to snap geometry to workplanes), then select the green checkmark to Accept changes.

Select the newly created pattern and right-click, selecting Edit > Transform… and use the Move Geometry tool with the Position form to position the pattern at roughly Z = 150 in the active workplane.
Go back to the **Embedded Pattern Finishing** toolpath form and go to **Tool Axis**. Set it to From Curve and then set the **Pattern** to **toolAxis**. **Calculate** the toolpath. We will need both axes of the external positioner to drive this toolpath properly, so go to **Robot Control** and set both **E1** and **E2** to **HIGH**. Simulate the toolpath. There are some rough patches to clean up, but the part is mostly finished from this single toolpath!

The progress file **myExternalAxis** is available as a course resource.

*Note: **Robot Control** settings can be saved to the toolpath by going to the **Tools** tab and selecting the **Save simulation settings to toolpath** button.*
Outputting Robot Toolpaths as NC Programs

Recording Simulations
You may have noticed that the robot toolpath simulation button doubles as a **Record** button. When a toolpath simulation looks good and you’re ready to export is as an NC program or as part of an NC program, save the recording as a **RobSim** file with the Save button at the top of the **Robot Control** panel. Maintaining the same name as was used for the toolpath will avoid confusion.

You can now play the analyzed simulation in the **Horizontal Plugin Window (HPW)** which will be the final check for errors and give you a toolpath ready for output.

*Note: Turning on **Trace axis limits** in the **Robot Control** panel will allow you to see the maximum range traversed per joint in a robot toolpath.*

Once you’ve played the simulation you can scrub through the toolpath move-by-move and a new option to home the program relative to the recorded simulation (under **Robot Control**) will appear. This simulation can be re-loaded at any point in the future using the **Load simulation file** button in the HPW. If the Simulation status turns into a green checkmark, there are no collisions detected. If it remains a warning symbol, you will need to return to the VPW simulation mode to chase down remaining issues, then save a new recording to re-simulate in the HPW.
You can view collisions by right-clicking on IRB6640 + Positioner under Machine Tools in PE and selecting Collisions. Selecting a line in this list will take you to the move causing the collision. Remember to Clear this list out when revising toolpaths as old collisions will pile up and cause confusion!

Adding Toolpaths to NC Program
To add toolpaths, go to the Robot Program drawer in the VPW and right-click anywhere within the white space of the NC program tab. Any simulated toolpath or transition between toolpaths can be added to the program, as well as custom user commands pertaining to your particular robotic language, such as ABB’s RAPID. Adjust the settings with the Parameters tab so that they reflect the requirements of your robotic setup. When finished, select Write robot NC program. Now you’re ready to machine with your robot!