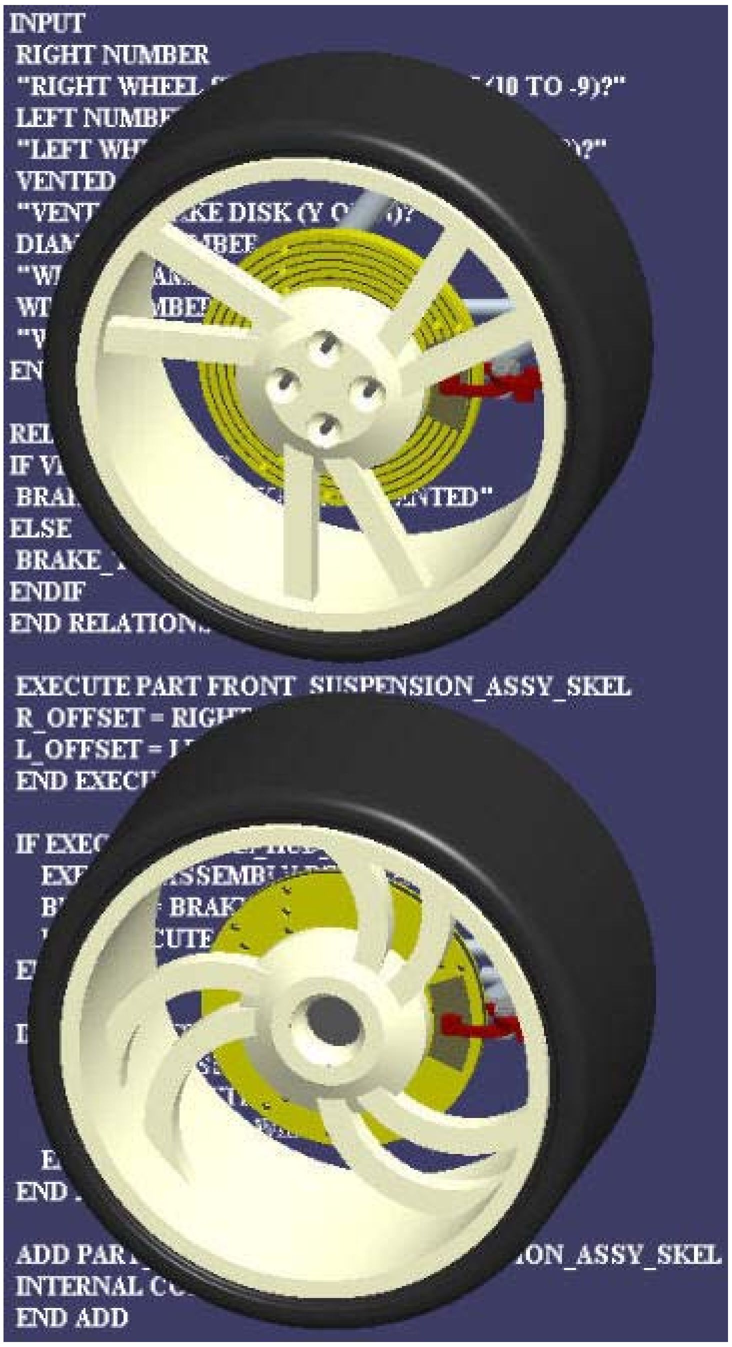
Welcome to the Pro/ENGINEER Pro/Program Hands-on Workshop



Tutorial Overview

THE GOAL: The goal of this tutorial is to give a brief, hands-on experience using PTC's built-in automation tool, Pro/Program

THE CONTEXT: You'll be working with a simplified automobile front suspension assembly.

THE FLOW: The flow of this tutorial goes something like this:

- **You'll begin by automating some changes to the wheel rim and capturing those changes in family table iterations.**
- **You'll then automate the wheel assembly to choose an appropriate wheel rim and size the tire.**
- **Once that's done, you'll investigate how Pro/Engineer can automatically substitute completely independent components in and out-of an assembly automatically.**
- **Finally, from a top-level assembly perspective you'll put it all together and automate the entire assembly.**

Hopefully you'll enjoy your experience with this workshop and take away from it how using Pro/Engineer for assembly automation just makes sense.

Pro/ENGINEER really is simple to use, but it's also a powerhouse of automation capability.

Before you get started

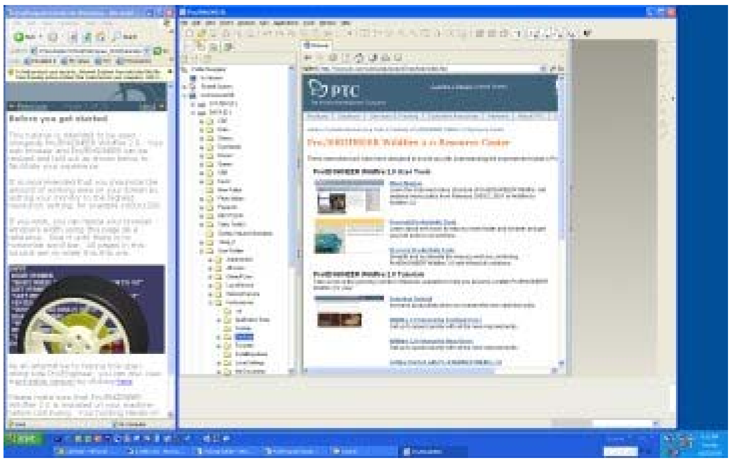
The Pro/ENGINEER model files for this tutorial can be found in a sub-folder called "source_files" or by clicking here.

Extract the Pro/E models to a folder and either start Pro/E in that folder or simply set the working directory to it after starting ProE (there are no special config files required for this tutorial).

This tutorial is intended to be used alongside Pro/ENGINEER Wildfire 2.0. Your web browser and Pro/ENGINEER can be resized and laid out as shown below to facilitate your experience.

It is recommended that you maximize the amount of working area on your screen by setting your monitor to the highest resolution setting, for example 1600x1200.

If you wish, you can resize your browser window's width using this page as a reference. Size it until there is no horizontal scroll bar. All pages in this tutorial are no wider this this one.



As an alternative to having this open along-side Pro/Engineer, you can also view a printable version by clicking here.

Please make sure that Pro/ENGINEER Wildfire 2.0 is installed on your machine before continuing. Your hosting Hands-on Workshop Application Engineer will have this set up for you.

Using this Tutorial

- Click **Next** or **Previous** at the top or at the bottom of each page to proceed through the tutorial.
- Carefully read all of the content on each page and perform ALL the given steps before proceeding to the next.
  - In several cases, you may have to 'scroll down' on the tutorial page before continuing.
- You can navigate back to the starting page by using the **Home** link found at the bottom of each page.

You will see various icons throughout the tutorial:

Information is provided for some of the functionality.

Tips are provided along the way.

Notes are provided as additional information.

Small Images:
There may be times when the picture illustration needs to be larger than the column format of this tutorial. In those cases you'll notice a button just beneath the image. Click on the button to open the full-size picture.

Conventions:
There are several conventions used when working with Pro/ENGINEER Wildfire 2.0:

- The "picks and clicks" are shown in **Bold**.

- Text that you enter is shown in ***Bold Italics***.
- Icons and their names are shown inline with the text.
- You can adjust the font size of this tutorial using either of two methods:
  - By clicking **View > Text Size** from the Internet Explorer window.
  - By pressing CTRL and rolling the mouse wheel.

Table of Contents

Click **NEXT** to begin the tutorial or you can select any of the links below to go directly to that section.

About Pro/Program

About Pro/Program

What is a Pro/Program?

Each model in Pro/ENGINEER contains a listing of major design steps and parameters that can be edited to work as a program. A each and every regenerate command executes this program.

What isn't a Pro/Program?

It isn't a true programming language

It isn't a way to make one single model generate your entire product catalog

There are 4 basic areas of a Pro/Program listing

• INPUT - This space holds any parameters that the programmer wishes changed upon regeneration

• RELATIONS - This space holds all part or assembly level relations and can be edited via the Pro/Program editor

EXECUTE STATEMENTS - This section is only valid for assembly Pro/Programs and it allows an assembly to pass down parameter values to its sub-components.

• BODY - This is the bulk of a Pro/Program listing. It contains a list of all the features, their state (active or suppressed), and key parameters.
-These entries can ONLY be created using the Pro/ENGINEER GUI interface (or Pro/TOOLKIT)
-However one can add conditional logic and parameter substitution to the existing features
-Deleting a feature entry in the program WILL delete it from the model tree as well

• MASS PROPERTIES - When setup, this section forces a mass property recalculation each time the geometry changes.

Task 1 Add Pro/Program control to a wheel rim part

During this task you will:

- Investigate the various parts of a Pro/Program
- Create input statements to query the user for values
- Add IF-ELSE logic to the part's regeneration list
- Create conditional input statements
- Instantiate family table entries based on the Pro/Program input values

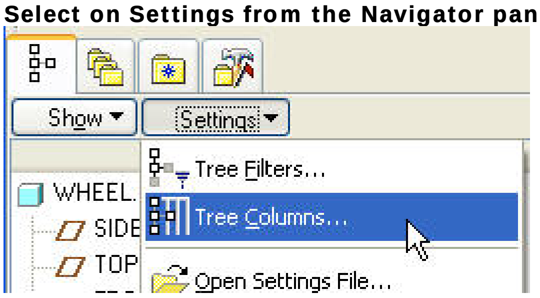
Task 1-1 Open and investigate the Wheel part model

Open **WHEEL.PRT** part

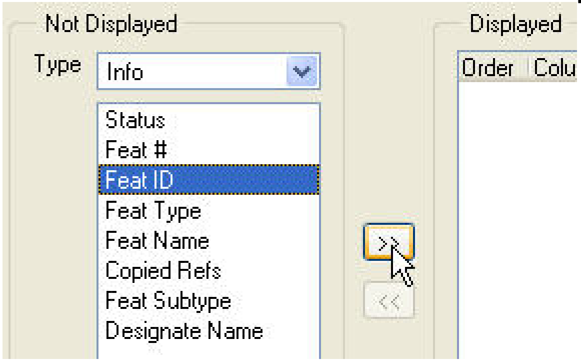
Turn off all the Datum displays (Plane, Axis, Point and Coordinate System) using

Zoom in on the center hub

Select on Settings from the Navigator panel and choose Tree Columns...

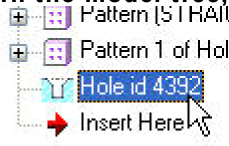


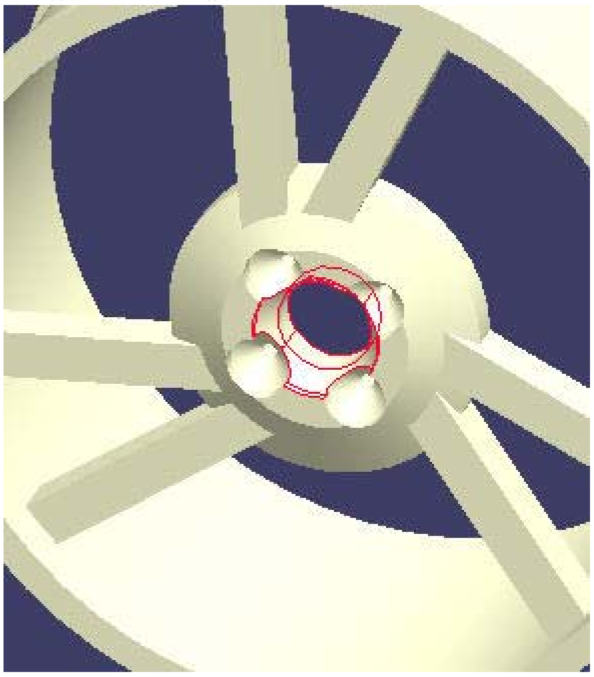
Select and add Feature ID to the columns displayed using the >> button



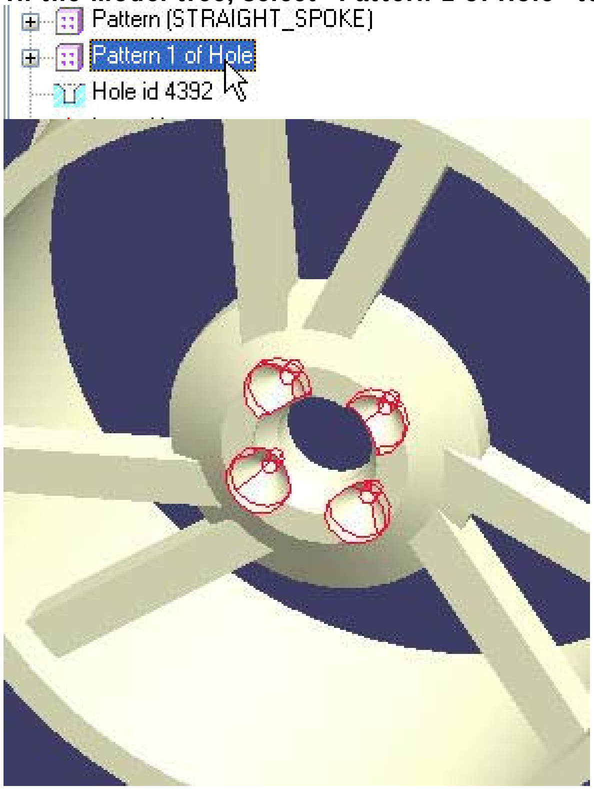
Unlike the Feature Number, Feature ID's are always unique and never change during the life of a Pro/Engineer model. This makes them a handy item to search for when editing the Pro/Program.

In the model tree, select "Hole id 4392" to highlight the single, spindle mounting hole feature

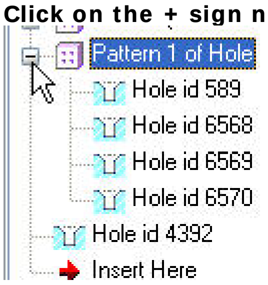




In the model tree, select "Pattern 1 of Hole" to highlight the 4 lug mounting hole features



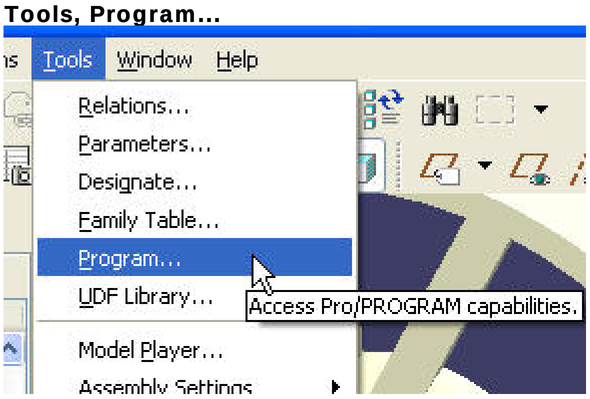
Click on the + sign next to the pattern to expand it



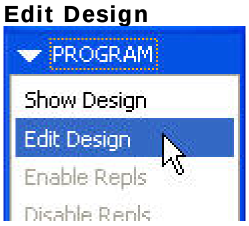
We only want either the lug hole pattern or the spindle hole active at any given moment, so let's modify the Pro/E part program to accomplish this.

Task 1-2 Pro/Program INPUT statements

Tools, Program...



Edit Design



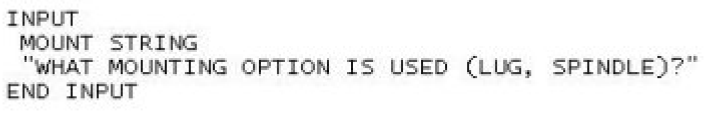
We'll begin by creating an INPUT command to capture the user's wish for the mounting type; either lug or spindle.

Find the INPUT and END INPUT line in the Pro/Program.

Type the following lines between the INPUT and END INPUT lines:

MOUNT STRING
"WHAT MOUNTING OPTION IS USED (LUG, SPINDLE)?"

It should look something like this (Note: Indentation doesn't matter)



```
INPUT
 MOUNT STRING
 "WHAT MOUNTING OPTION IS USED (LUG, SPINDLE)?"
END INPUT
```

This can be read as, 'Ask the user to input the value for the parameter MOUNT, which is a string, and display the message, "What mounting option is used (lug, spindle)?" when asking.'

Task 1-3 Pro/Program IF-ELSE statements in the body

Now, we need to use the value of MOUNT to make some logic decisions. We only want the lug features to be regenerated when MOUNT equals "LUG" and the spindle feature to be regenerated only when MOUNT equals "SPINDLE".

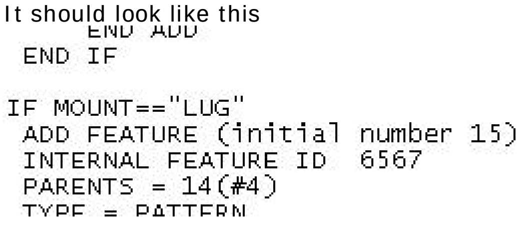
This is accomplished using IF-ELSE logic in the body (where the features are listed) of the Pro/Program.

Find Feature ID 6567 (the lug hole pattern) in the Pro/Program listing (use the editor's find tool if necessary)

Add the following line before the ADD FEATURE (initial number 15) entry

IF MOUNT=="LUG"

It should look like this



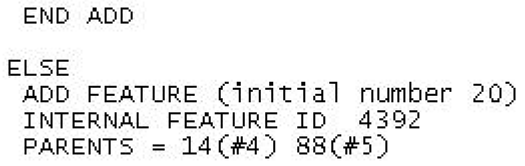
```
      END ADD
 END IF

IF MOUNT=="LUG"
 ADD FEATURE (initial number 15)
 INTERNAL FEATURE ID  6567
 PARENTS = 14(#4)
 TYPE = PATTERN
```

Find Feature ID 4392 (the spindle feature) in the Pro/Program listing

Add an "*ELSE*" just before the feature entry.

It should look like this:

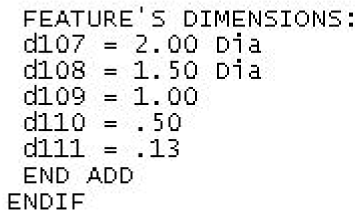


```
 END ADD

ELSE
 ADD FEATURE (initial number 20)
 INTERNAL FEATURE ID  4392
 PARENTS = 14(#4) 88(#5)
```

Finally, add an "ENDIF" after the spindle feature entry (just before the MASSPROP line).

It should look like this:

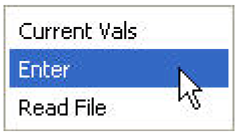


```
 FEATURE'S DIMENSIONS:
 d107 = 2.00 Dia
 d108 = 1.50 Dia
 d109 = 1.00
 d110 = .50
 d111 = .13
 END ADD
ENDIF
```

Finally, exit the editor program using

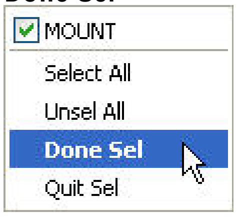
Yes **to save the changes**

Yes **to incorporate your changes into the model**

Select "Enter" 

to have Pro/E ask you for the input values

Check the box next to MOUNT (or use Select All),
Done Sel



Type *SPINDLE* when prompted (case is important) and press Enter

Notice how the lug holes disappear.

Now, Regenerate the model

Choose Enter

Check MOUNT, Done Sel from the menus

Type *LUG* when prompted.

Notice how the spindle feature goes away and the lug holes return.

Task 1-4 Pro/Program Input IF-ELSE statements

If a lug mounting is choosen, Pro/E should also ask how many holes are in the pattern.

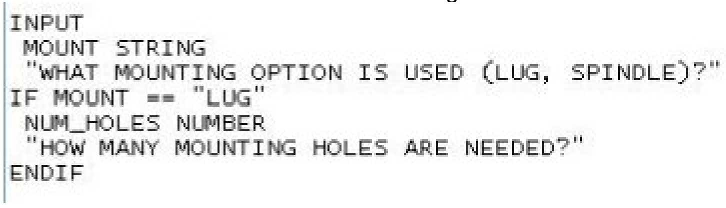
IF-ELSE statements can also be used within the INPUT block to provide conditional input of parameter values.

Enter the Pro/Program menu using Tools, Program..., Edit Design

Just above the "END INPUT" line, insert the following

```
IF MOUNT == "LUG"
  NUM_HOLES NUMBER
  "HOW MANY MOUNTING HOLES ARE NEEDED?"
ENDIF
```

The INPUT block should look something like this now:



```
INPUT
 MOUNT STRING
 "WHAT MOUNTING OPTION IS USED (LUG, SPINDLE)?"
IF MOUNT == "LUG"
 NUM_HOLES NUMBER
 "HOW MANY MOUNTING HOLES ARE NEEDED?"
ENDIF
```

The parameter NUM_HOLES drives the pattern of holes using the pre-defined relation, "P147=NUM_HOLES" located in the RELATIONS section of the program. We'll be creating relations during Task 2.

Exit the editor and incorporate the changes as before.

Select all values to input

Enter *LUG* when asked for the mounting type.

Enter *5* when asked for the number of holes.

Now, Regenerate the model again and enter *SPINDLE* this time

Notice how you're not asked for the number of holes.

Task 1-5 Pro/Program and Family Table Instances

There's already logic designed to capture some wheel spoke options, so let's finish the input options to capture the parameters for that too.

Edit the Pro/Program

Type or copy and paste the following lines into the INPUT block <u>above</u> the existing MOUNT lines

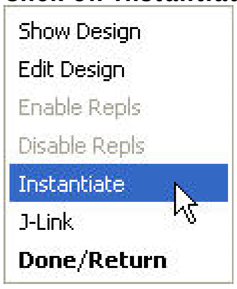
```
RIM_DIA NUMBER
"WHAT IS THE RIM DIAMETER?"
RIM_WIDTH NUMBER
"WHAT IS THE RIM WIDTH?"
STRAIGHT YES_NO
"DO YOU WANT THE STRAIGHT SPOKES INCLUDED (Y,N)?"
IF STRAIGHT==YES
    SIDE STRING
    "DO YOU WANT RIGHT OR LEFT HANDED SPOKES (L,R)"?"
END IF
R_CURVED YES_NO
"DO YOU WANT THE RIGHT HAND CURVED SPOKES INCLUDED (Y,N)?"
L_CURVED YES_NO
"DO YOU WANT THE LEFT HAND CURVED SPOKES INCLUDED (Y,N)?"
IF L_CURVED==YES | R_CURVED==YES
    CURVE_RAD NUMBER
    "WHAT IS THE RADIUS OF THE CURVED SPOKES?"
END IF
```

Exit the editor, incorporate the changes, select all parameters to input, and Enter the following values:

RIM_DIA = *12*
RIM_WIDTH = *6*
STRAIGHT = *Y*
SIDE = *R*
R_CURVED = *N*
L_CURVED = *N*
MOUNT = *LUG*
NUM_HOLES = *4*

These input values can be captured in a family table simply by "instantiating" them.

Click on Instantiate in the Program menu



Enter *RIM1* when prompted for the name

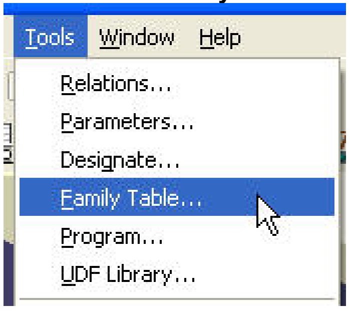
Regenerate the model

Select all parameters to input and Enter the following values:

RIM_DIA = *12*
RIM_WIDTH = 8
STRAIGHT = N
R_CURVED = Y
L_CURVED = *N*
CURVE_RAD = *3.5*
MOUNT = SPINDLE

Instantiate this set of parameters and enter RIM2 when prompted

Review the family table using Tools, Family Table...



Notice how all the input parameters are now columns in the table.

Exit the family table editor using

Window, Close

Task 2 Add Assembly Pro/Program control to the Wheel Rim Assembly

During this task you will:

- Learn an alternate syntax for INPUT statements
- Create a relation to choose a family table instance by matching parameter values
- Use an EXECUTE statement to push parameter values into lower level objects

Task 2-1 Wheel Assembly Pro/Program Input and Relations

If the WHEEL.prt window isn't closed yet, do so now using Window, Close

Open TIRE.PRT model

Tools, Program...

Show Design

Make note of the two input lines in this part model. Three things stand out:
- There are no quoted questions for the input lines.
- There are two INPUT parameters: TIRE_WIDTH and RIM_SIZE
- The values of the parameters are shown (i.e. RIM_SIZE NUMBER = 12.000000)

The quoted question is optional and if the parameters are self-descriptive the question is often unnecessary.

TIRE_WIDTH and RIM_SIZE are the parameter names that are "local" to the tire.prt model. A little later we'll be using a special command called an EXECUTE statement to change both of these, as needed, from the wheel_assy model. For now, just note the names of the parameters.

The current parameter values being shown in the input section is simply a function of that the Show Design tool. If you wish, you can use the Edit Design option instead.

Close **the information window**

Window, Close

Open WHEEL_ASSY.ASM model

Edit the Pro/Program design

Add INPUT entries as follows:

WHEEL_DIAMETER NUMBER
WHEEL_WIDTH NUMBER

Add a RELATION (between the RELATIONS and END RELATIONS lines) as follows:

You may not want to copy and paste this relation, type it instead. Sometimes the quote characters don't come across correctly for some reason -- it might be the backslashes.

***WHEEL_INSTANCE= \\
lookup_inst("wheel.prt",0, \\
"RIM_DIA",WHEEL_DIAMETER, \\
"RIM_WIDTH",WHEEL_WIDTH)***

This "lookup_inst" function is a very powerful feature of Pro/Program with assemblies.

It allows one to have Pro/E automatically lookup a family table instance that matches a series of parameter values. It can be instructed to match exactly or to grab the next lower or next higher close match.

In this case, we're looking at the wheel.prt family table we added instances to in the previous task. We're asking it to find an exact match (the 0 entry) where the RIM_DIA parameter (in wheel.prt) is equal to the WHEEL_DIAMETER value we just added to the INPUT block and the RIM_WIDTH parameter is equal to the value of WHEEL_WIDTH also just added.

The new parameter WHEEL_INSTANCE is a string parameter which will receive the name of the instance once a match is found.

The back slash character, "\" can be used to continue an entry on the following line. Note, Pro/Program has a limit of 80 characters per line so use "\" when necessary.

Task 2-2 Wheel Assembly Pro/Program Execute and Body Edit

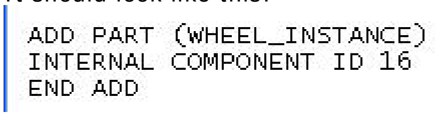
Let's first use that WHEEL_INSTANCE parameter we just set using the lookup_inst function.

Scroll to the bottom of the Pro/Program listing

Look for the line that reads, "ADD PART WHEEL"

Replace "WHEEL" with "(WHEEL_INSTANCE)"

It should look like this:



```
ADD PART (WHEEL_INSTANCE)
INTERNAL COMPONENT ID 16
END ADD
```

By replacing any fixed part or assembly callout in the body of a Pro/Program [i.e. ADD PART WHEEL] with a string parameter enclosed in parentheses [i.e. ADD PART (WHEEL_INSTANCE)], Pro/E will then assemble the object named in the parameter instead.

Of course, this means that the object must be able to be automatically assembled without assistance. This can be accomplished either by substituting a family table instance or by using an interchange assembly. We'll cover the interchange assembly in the next task.

Now, scroll back up to the RELATIONS block

Immediately after the END RELATIONS line, type the following:

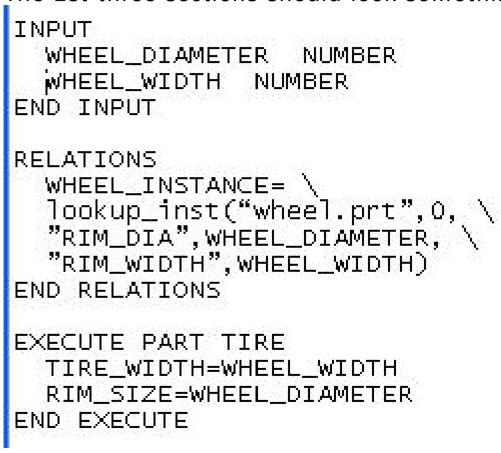
```
EXECUTE PART TIRE
  TIRE_WIDTH=WHEEL_WIDTH
  RIM_SIZE=WHEEL_DIAMETER
END EXECUTE
```

Note that the EXECUTE statement assigns to the TIRE.PRT's local parameters, TIRE_WIDTH and RIM_SIZE, the values of the assembly parameters, WHEEL_WIDTH and WHEEL_DIAMETER respectively.

Because of this syntax, one can use whatever parameter name suits the situation without needing to keep everything consistant.

Another advantage is the ability to apply equations when setting the value. As an example: If the tire.prt model was built using millimeters but the wheel assembly was in inches, one could use TIRE_WIDTH=WHEEL_WIDTH*25.4 to correctly size the tire for the rim.

The 1st three sections should look something like this:



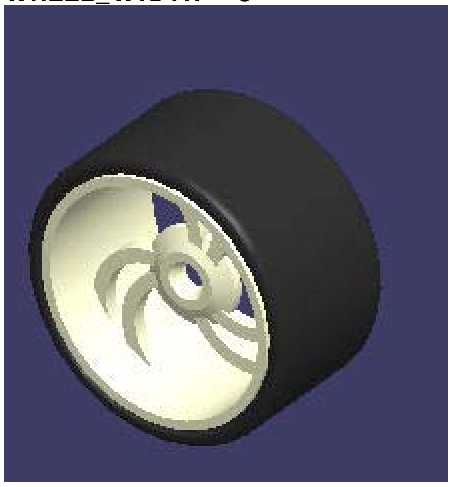
```
INPUT
  WHEEL_DIAMETER  NUMBER
  WHEEL_WIDTH  NUMBER
END INPUT

RELATIONS
  WHEEL_INSTANCE= \
  lookup_inst("wheel.prt",0, \
  "RIM_DIA",WHEEL_DIAMETER, \
  "RIM_WIDTH",WHEEL_WIDTH)
END RELATIONS

EXECUTE PART TIRE
  TIRE_WIDTH=WHEEL_WIDTH
  RIM_SIZE=WHEEL_DIAMETER
END EXECUTE
```

Exit the editor, incorporate the changes, and Enter new values as follows:

WHEEL_DIAMETER = *12*
WHEEL_WIDTH = *8*



Regenerate

Now try these values:

WHEEL_DIAMETER = *12*
WHEEL_WIDTH = *6*

Notice how the wheel rim changes based on the family table match while the tire model is simply flexed based on the values given it. This is also evident in the model tree -- TIRE.PRT stays the same, but WHEEL.PRT becomes either RIM1 or RIM2.

Window, Close

Task 3 Add Assembly Pro/Program control to the Wheel Hub Assembly

During this task you will:

- Investigate an interchange assembly
- Add Pro/Program logic to the assembly to swap interchangeable components

Task 3-1 Investigate Brake Disk Interchange Assembly

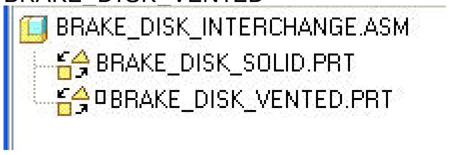
We'll begin by investigating the functional interchange assembly for the Brake Disk parts.

An interchange assembly is a special type of Pro/ENGINEER assembly that sets up the conditions to allow an automatic replacing of one component for another in another, regular assembly. The components are typically seperate, unrelated models.

Two types of interchange capability is supported
  • Simplification interchange - The location of the new component relative to the old is based solely on the relative position of the objects within the interchange assembly.
  • Functional interchange - The required references from one object (relative to the assembly its used in) are mapped to equivalent geometry on the other objects. This allows a completely functional interchange to occur.

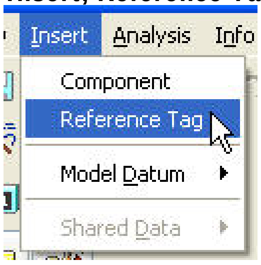
Open BRAKE_DISK_INTERCHANGE.ASM

Notice from the model tree that there are two disk brakes in this assembly, BRAKE_DISK_SOLID and BRAKE_DISK_VENTED

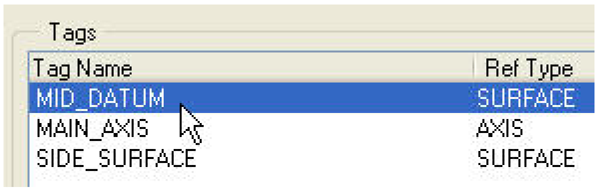

BRAKE_DISK_INTERCHANGE.ASM
BRAKE_DISK_SOLID.PRT
BRAKE_DISK_VENTED.PRT

Since this is a functional interchange, let's look at the reference mapping between the two brake disk parts

Insert, Reference Tag



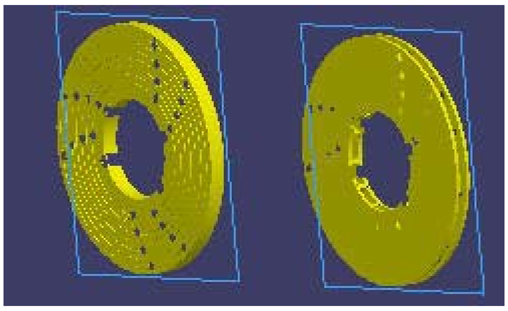
Click through the three named references: MID_DATUM, MAIN_AXIS and SIDE_SURFACE

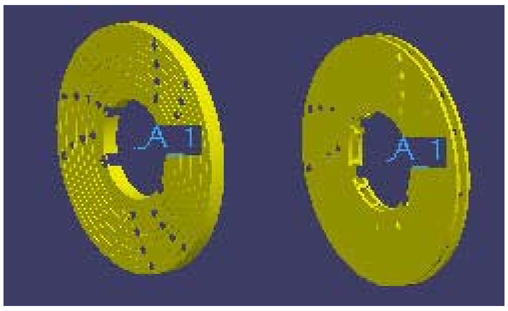


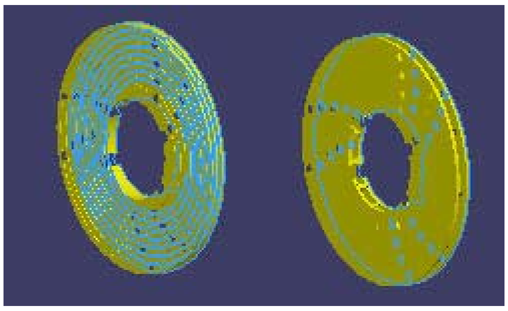
Tags

Tag Name	Ref Type
MID_DATUM	SURFACE
MAIN_AXIS	AXIS
SIDE_SURFACE	SURFACE

Notice the highlighted, tagged references.







Cancel **The Reference Tag dialog box**

Yes **to confirm**

Window, Close

Task 3-2 Edit the Wheel Hub Assembly's Pro/Program

Open RF_WHEEL_HUB.ASM

Edit the Pro/Program

Add an INPUT statement to ask for the name of the brake disk part as follows:

BRAKES STRING
"WHICH DISKS DO YOU WANT (BRAKE_DISK_SOLID, BRAKE_DISK_VENTED)?"

Now, add an EXECUTE statement (after the RELATIONS block) that simply activates the interchange assembly as follows:

EXECUTE ASSEMBLY BRAKE_DISK_INTERCHANGE
END EXECUTE

Find the existing ADD statement in the body that assembles BRAKE_DISK_SOLID

Change BRAKE_DISK_SOLID to read (BRAKES)

It should look like this:

```
ADD PART (BRAKES)
INTERNAL COMPONENT ID 19
PARENTS = 16(#4) 18(#5)
END ADD
```

Exit the editor, incorporate the changes and enter BRAKE_DISK_VENTED for the parameter BRAKES when prompted.

Notice the replaced component.

Window, Close

Task 4 Top Level Assembly Pro/Program

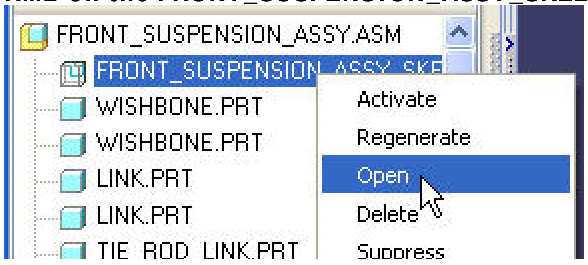
During this task you will:

- Investigate one way a skeleton model can be used with Pro/Program
- Investigate the behavior of lower-level Pro/Program INPUT statements when regenerating the top-level
- Edit the top-level assembly Pro/Program to drive the components

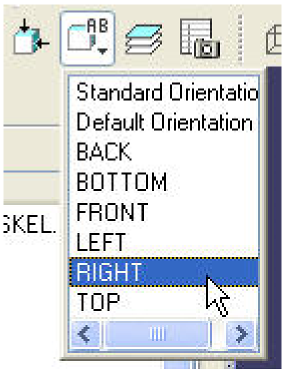
Task 4-1 Investigate the Skeleton Model

Open FRONT_SUSPENSION_ASSY.ASM

RMB on the FRONT_SUSPENSION_ASSY_SKEL.PRT line in the model tree and choose OPEN



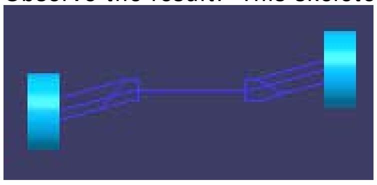
Reorient to the RIGHT view



Regenerate

Enter *-6* for R_OFFSET
Enter *2* for L_OFFSET

Observe the result. This skeleton model drives the location of the linkages and wheels for the top-level assembly.



Window, Close

Task 4-2 Top-Level Pro/Program Work

Let's see what happens when we regenerate a top-level assembly when sub-assemblies and parts have Pro/Program INPUT statements.

Regenerate

Notice in the message window that the first GET INPUT is requesting parameter values for the FRONT_SUSPENSION_ASSY_SKEL

Select Current Vals to use the existing values

The next request is from the RF_WHEEL_HUB_ASSY Pro/Program

Select Current Vals to use the existing values

Finally WHEEL_ASSY makes its request.

Select Current Vals

To drive all these requests from a single top-level location, we'll need to push the appropriate parameter values into the sub-assemblies and parts using EXECUTE statements

Specifically:
- FRONT_SUSPENSION_ASSY_SKEL needs R_OFFSET and L_OFFSET,
- RF_WHEEL_HUB_ASSY needs BRAKES, and
- WHEEL_ASSY needs WHEEL_DIAMETER and WHEEL_WIDTH

Edit the Pro/Program

Create the following INPUT statements

RIGHT NUMBER
"RIGHT WHEEL SUSPENSION OFFSET (10 TO -9)?"
LEFT NUMBER
"LEFT WHEEL SUSPENSION OFFSET (10 TO -9)?"
VENTED YES_NO
"VENTED BRAKE DISK (Y OR N)?"
DIAMETER NUMBER
"WHEEL DIAMETER?"
WIDTH NUMBER
"WHEEL WIDTH?"

Create the following RELATIONS

```
IF VENTED == YES
 BRAKE_TYPE="BRAKE_DISK_VENTED"
ELSE
 BRAKE_TYPE="BRAKE_DISK_SOLID"
ENDIF
```

Finally, create the following EXECUTE statements

```
EXECUTE PART FRONT_SUSPENSION_ASSY_SKEL
 R_OFFSET = RIGHT
 L_OFFSET = LEFT
END EXECUTE

EXECUTE ASSEMBLY RF_WHEEL_HUB_ASSY
 BRAKES = BRAKE_TYPE
END EXECUTE

EXECUTE ASSEMBLY WHEEL_ASSY
 WHEEL_DIAMETER = DIAMETER
 WHEEL_WIDTH = WIDTH
END EXECUTE
```

Exit the editor, incorporate the changes and enter these values: 0, 0, YES, 12, 6.

Remember, go back to WHEEL.PRT and instantiate more family table instances if desired.

Instantiate some top-level family table instances as well if desired.

Summary

Tutorial Summary

Hopefully you've enjoyed this workshop experience. However, there's more to be done with Pro/Program: Interact statements, mass property calculations, use in user defined features, etc.

Coupled with capabilities in the Advanced Assembly Extension such as the interchange assemblies you saw, Pro/Program becomes a powerful tool to help automate your design needs.

Further help can be found in the Pro/Engineer help system. Choose "Model Analysis" for the functional area and "Pro/PROGRAM" for the module.

To get started using Advanced Assembly Extension, schedule a demonstration or learn more, please contact your PTC sales representative or your local reseller.

You can contact PTC through our website at www.ptc.com